
User's Guide

Publication number E8170-97001
October 2000

For Safety information, Warranties, and Regulatory information, see the pages behind the index.

© Copyright Agilent Technologies, Inc. 1994-2000
All Rights Reserved

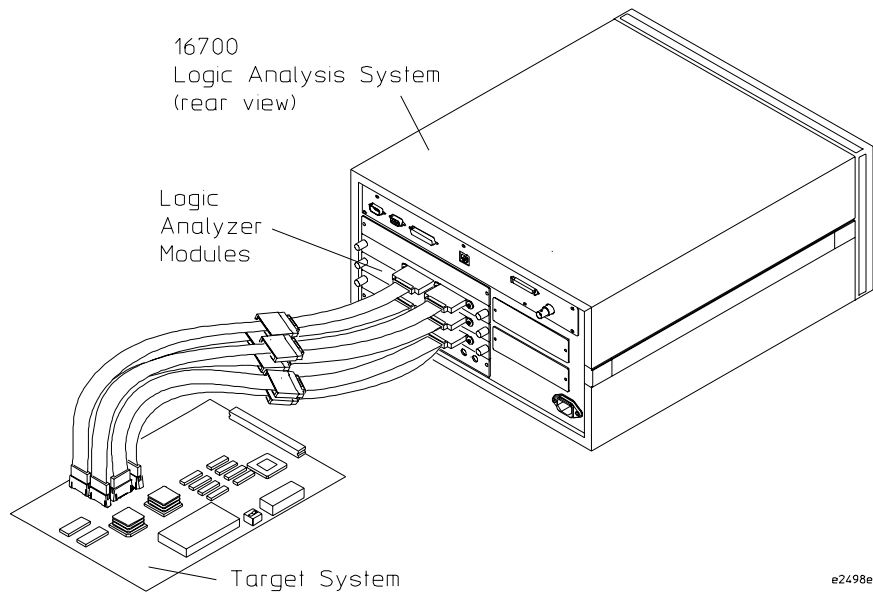
Logic Analysis Support for the Motorola MPC7400

Logic Analysis Support for the MPC7400— At a Glance

This book documents the MPC7400 inverse assembler.

Inverse Assembler Software

The Agilent Technologies E8170B inverse assembler, in conjunction with an Hewlett-Packard or Agilent Technologies logic analyzer, allows you to view MPC7400 assembly instructions that are executing in your target system. The inverse assembler can be configured to work with signals that are available for probing.



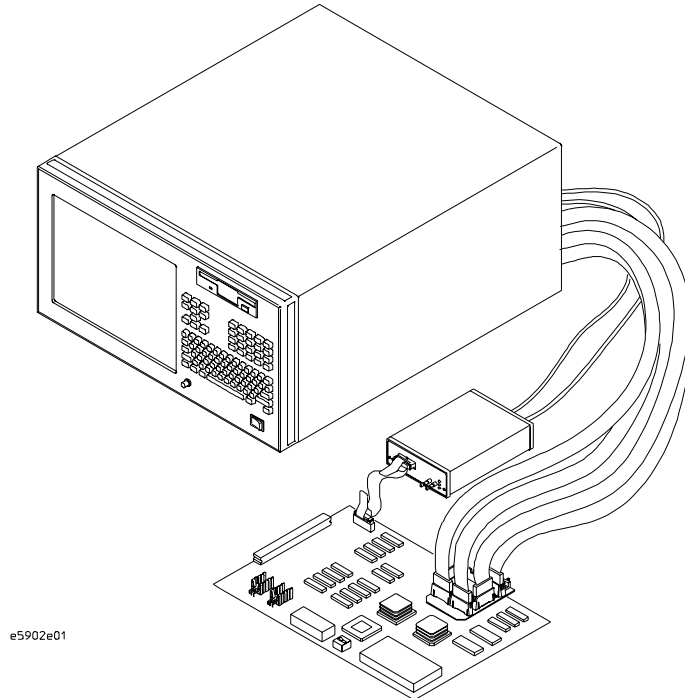
The inverse assembler has a cache-on trace reconstruction option, which makes use of the processor's branch trace mode to reconstruct the full software trace. This lets the processor run full speed without being interrupted from internal flash or cache while processor execution trace is captured.

The inverse assembler model number is Agilent Technologies E9613B Option 001 when ordered alone. It is identified as "E8170B" in the Setup Assistant.

If You Purchased an Emulation Solution

The E9513B emulation solution lets you use an Agilent Technologies logic analysis system to debug and characterize an MPC7400 target system. The emulation solution is a bundled product consisting of an inverse assembler, an emulation probe (and its cables and adapters), and the Agilent Technologies B4620B source correlation tool set. This solution is designed to be used with an HP/Agilent Technologies 16700-series logic analysis system.

Emulation Solution



Additional Equipment Included in an Emulation Solution

Emulation Module and Emulation Probe

The emulation module plugs into your logic analysis system frame, and the emulation probe connects to the emulation module and a debug port on your target system. The emulation probe and module let you use the target processor's built-in background debugging features, including run control and accessing registers and memory. A high-level source debugger can use the emulation probe/module to debug code running on the target system.

Information about using the logic analysis system with the emulation probe/module can be found in Chapter 8, “Coordinating Logic Analysis with Processor Execution,” beginning on page 117 of *this* manual.

The *Emulation for the MPC7400 User's Guide* describes setting up and using the emulation probe and emulation module.

Source Correlation Tool Set

The Agilent Technologies B4620B Source Correlation Tool Set lets you set up logic analyzer triggers based on source code, and view the source code associated with signal values captured by the logic analyzer.

Additional Information Sources

Newer editions of this manual may be available. Contact your local Agilent Technologies representative.

Application notes may be available from your local Agilent Technologies representative or on the World Wide Web at:

<http://www.agilent.com/find/logicanalyzer>

If you have an HP/Agilent 16700-series logic analysis system with an emulation probe/module, the **online help** for the Emulation Control Interface has additional information on using the emulation module. Also, see the emulation manual included with your emulation probe/module.

The **measurement examples** include valuable tips for using emulation and making analysis measurements. You can find the measurement examples under the system help in your HP/Agilent 16700-series logic analysis system.

In This Book

This book documents the following products:

Inverse Assembler Software		
Processors supported	Product ordered	Includes
MPC7400	E9613B Option 001	E8170B inverse assembler

Related equipment

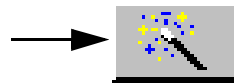
The following equipment is included in the MPC7400 emulation solution:

Processor supported	Product ordered	Includes
MPC7400	Agilent Technologies E9513B Option 001 emulation solution	E8170B inverse assembler, emulation probe, emulation module, and B4620B Source Correlation Tool Set

Tips To Save You Time

Use the Setup Assistant

Click here to connect the logic analyzer cables and automatically load the correct configuration files. See page 17.

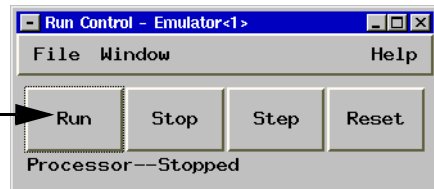


Use the appropriate Run button



Click here to start a measurement.

If your system includes an emulation probe/module, click here to run the target microprocessor.



Contents

Use the Setup Assistant 6
Use the appropriate Run button 6

1 Overview 15

Setup Checklist 16

Setup Assistant 17

Equipment Used with Inverse Assembler Software 18

Equipment supplied 18

Minimum equipment required 18

Additional equipment supported 19

B4620B Source Correlation Tool Set 19

Compatible Logic Analyzers 20

Emulation Solution 22

2 Preparing the Target System 23

Target System Requirements 24

Direct Probing with General Purpose (GP) Probes 24

Designing and using built-in connectors 24

AMP MICTOR 38 connectors 27

Design Considerations 27

Support shroud 28

Inverse assembler—signal-to-connector mapping	30
Designing a JTAG Connector into Your Target System	41

3 Setting Up the Logic Analysis System 43

Power-on/Power-off Sequence	44
To power on 16700-series logic analysis systems	44
To power off	44

Installing Logic Analyzer Modules	45
Installing the Emulation Module	45

Installing and loading software	46
What needs to be installed	46
To install the software from CD-ROM	47
To list software packages which are installed	48

4 Connecting the Logic Analyzer to the Target System 49

To connect the high-density termination cables to the target system	51
To connect to the 16715/16/17/18/19/50/51/52A logic analyzer (one card)	52
To connect to the 16715/16/17/18/19/50/51/52A logic analyzer (two cards)	53
To connect to the 16715/16/17/18/19/50/51/52A logic analyzer (three cards)	54
To connect to the 16710/11/12A logic analyzer (one card)	55
To connect to the 16710/11/12A logic analyzer (two cards)	56

To connect to the 16550A analyzer (one card)	57
To connect to the 16550A analyzer (two cards)	58
To connect to the 16554/55/56/57 (one-card)	59
To connect to the 16554/55/56/57 (two-cards)	60
To connect to the 16554/55/56/57 (three-cards)	61

5 Configuring the Logic Analyzer 63

Configuring 16700-series Logic Analysis Systems	65
To load configuration files (and the inverse assembler) from the system hard disk	66
Logic Analyzer Configuration Files	67
Using the Format Menu	69
Bit ordering conventions	69
Loading Symbol Information	70
To view predefined MPC7400 symbols	70
To create user defined symbols	72
To load object file symbols	74
Symbol use requirements	76
An accurate bus trace	76
Direct address translation	76
An inverse assembler for trace lists	76
A symbol file	76
To display symbols	77

6 Capturing Processor Execution 79

To Set Up Logic Analyzer Triggers	81
-----------------------------------	----

To Setup Trigger Alignment and Offset for Symbols and Source Code	83
Using trigger alignment	83
To compensate for relocated code	86
To Trigger on Source Code	88
To avoid capturing library code execution	89

7 Displaying Captured MPC7400 Execution 91

To Display Captured State Data	92
To Use the Inverse Assembler	94
To use the Invasm menu	94
To load the inverse assembler	94
To set the inverse assembler preferences	94
Processor Options	95
Processor Type	95
Endian Mode	95
Aack Mode	96
Decoding Options	96
External Bus Decoding	97
Data Bus Connected	97
Simplified Mnemonic Decoding	97
Exception Decoding	100
Opcode Source	101
Specifying use of Motorola S-record executable file	101
S-Record Image Relocation	101
To use the inverse assembler filters	102
To Interpret the Inverse Assembler Output	104
Data Formats	104
Branch Instructions	104

Contents

Overfetch Marking	104
To Use Cache-On Trace Reconstruction	105
Minimizing effects of cache-on trace on system performance	106
To enable branch exception disassembly	108
Inverse Assembler Modes of Operation	109
To enable/disable the instruction cache on the MPC7400	110
To disable the cache with the emulation module:	110
To disable the cache with code:	110
To View the Source Code Associated With Captured Data	112
Inverse Assembler Generated SW_ADDR (Software Address) Label	113
Access to Source Code Files	114
Source File Search Path	114
Network Access to Source Files	114
Source File Version Control	114
To Display Captured Timing Analysis Mode Data	115

8 Coordinating Logic Analysis with Processor Execution 117

What are some of the tools I can use?	118
Which assembly-level listing should I use?	118
Which source-level listing should I use?	119
Where can I find practical examples of measurements?	119
Stopping Processor Execution on a Logic Analyzer Trigger	120
To stop on a source line trigger (Source Viewer window)	120
To stop on any trigger (Intermodule window)	122
To minimize the “skid” effect	123
To stop the analyzer and view a measurement	123

Contents

Tracing Until the Processor Halts	125
To capture a trace before the processor halts	125
Triggering the Logic Analyzer when Processor Execution Stops	126
The Emulation Module Trigger Signal	126
Group Run	127
The intermodule bus signals can still be active even without a Group Run.	127
Group Run into an emulation module does not mean that the Group Run will Run the emulation module.	127
Debuggers can cause triggers	128
To trigger the analyzer when the processor halts	129
To trigger the analyzer when the processor reaches a breakpoint	130

9 General-Purpose ASCII (GPA) Symbol File Format 133

GPA Record Format Summary	136
SECTIONS	138
FUNCTIONS	139
VARIABLES	140
SOURCE LINES	141
START ADDRESS	142
Comments	142

10 Troubleshooting the Inverse Assembler 143

Logic Analyzer Problems	145
Intermittent data errors	145
Unwanted triggers	145
No activity on activity indicators	146

Contents

No trace list display	146
Analyzer won't power up	146
Target System Problems	147
Target system will not boot up	147
Erratic trace measurements	148
Capacitive loading	148
Inverse Assembler Problems	149
No inverse assembly or incorrect inverse assembly	149
Inverse assembler will not load or run	151
Intermodule Measurement Problems	152
An event wasn't captured by one of the modules	152
Logic Analyzer Messages	153
“. . . Inverse Assembler Not Found”	153
“Measurement Initialization Error”	153
“No Configuration File Loaded”	154
“Selected File is Incompatible”	155
“Slow or Missing Clock”	155
“Time from Arm Greater Than 41.93 ms”	155
“Waiting for Trigger”	156

11 Hardware Reference 157

Operating characteristics	158
---------------------------	-----

Glossary 161

Index 167

Overview

Setup Checklist


Follow these steps to connect your equipment:

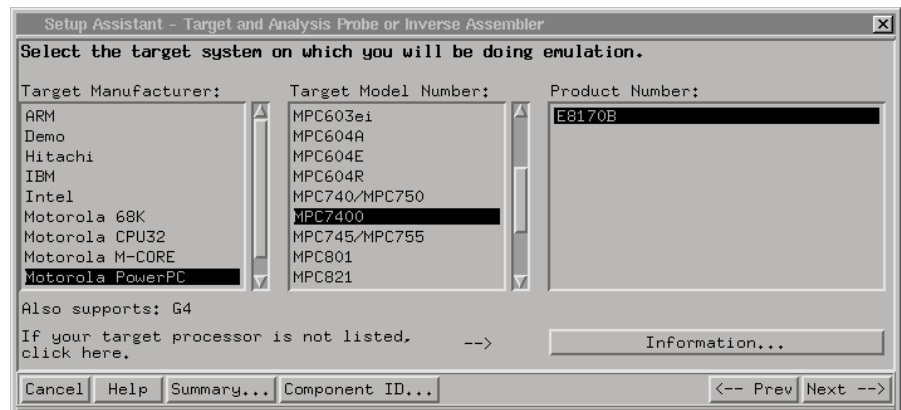
- If you need to install an emulation module in an Agilent Technologies 16700-series logic analysis system, see your emulation manual.
- Install the software. See page 46.
- This software is for use with HP/Agilent 16700-series logic analysis systems. Use the Setup Assistant to help you connect and configure your logic analysis system. See page 17.

Setup Assistant

The Setup Assistant is an online tool for connecting and configuring your logic analysis system for microprocessor and bus analysis. The Setup Assistant is available on the 16700-series logic analysis systems. You can use the Setup Assistant in place of the connection and configuration procedures provided in this manual.

This menu-driven tool will guide you through the connection procedures for connecting the logic analyzer to the target system, an emulation module, or other supported equipment.

Start the Setup Assistant by selecting  in the system window.



If you ordered this inverse assembler software with your 16700-series logic analysis system, the logic analysis system has the latest software installed, including support for this product. If you received this product after you received your logic analysis system, see the "Installing Software" chapter (page 46).

Equipment Used with Inverse Assembler Software

This section lists equipment supplied with the inverse assembler software and equipment requirements for using the inverse assembler software.

Equipment supplied

The E8170B inverse assembler software package consists of the following:

- Logic analyzer configuration files and the inverse assembler on a CD-ROM.
 - This User's Guide.
-

Minimum equipment required

For state and timing analysis of a MPC7400 target system, you need all of the following items.

- The E8170B MPC7400 Inverse Assembler.
- The appropriate connectors on the target system. Chapter 2, “Preparing the Target System,” contains information on designing the appropriate connectors into the target system. You could use General Purpose probes instead of connectors.
- One of the logic analyzers listed on page 20.

Additional equipment supported

B4620B Source Correlation Tool Set

The inverse assembler software may be used with the B4620B Source Correlation Tool Set on an 16700-series logic analysis system.

Compatible Logic Analyzers

The table below lists the logic analyzers supported by the E8170B inverse assembler software.

The E8170B requires eight logic analyzer pods (136 channels) for traditional inverse assembly (with 64-bit data). Inverse assembly can also be done with four pods (no data). See “Connecting the Logic Analyzer to the Target System” on page 49 for details.

Logic Analyzers Supported

Logic Analyzer	Channel Count	State Speed *	Timing Speed *	Memory Depth
16752A (1, 2, or 3 cards)	68/card	400 MHz	2 GHz	32 M states
16751A (1, 2, or 3 cards)	68/card	400 MHz	2 GHz	16 M states
16750A (1, 2, or 3 cards)	68/card	400 MHz	2 GHz	4 M states
16719A (2 or 3 cards)	68/card	333 MHz	2 GHz	32 M states
16718A (2 or 3 cards)	68/card	333 MHz	2 GHz	8 M states
16717A (2 or 3 cards)	68/card	333 MHz	2 GHz	2 M states
16716A (2 or 3 cards)	68/card	167 MHz	2 GHz	512 k states
16715A (2 or 3 cards)	68/card	167 MHz	667 MHz	2 M states
16712A (1 or 2 cards)	102/card	100 MHz	500 MHz	128 k states
16711A (1 or 2 cards)	102/card	100 MHz	500 MHz	32 k states
16710A (1 or 2 cards)	102/card	100 MHz	500 MHz	8 k states
16557D (1 or 2 cards)	68/card	140 MHz	500 MHz	2 M states
16556A (1, 2, or 3 cards)	68/card	100 MHz	400 MHz	1 M states

Logic Analyzer	Channel Count	State Speed *	Timing Speed *	Memory Depth
16555D/56D (1, 2, or 3 cards)	68/card	110 MHz	500/400 MHz	2 M states
16555A (1, 2, or 3 cards)	68/card	110 MHz	250 MHz	1 M states
16554A (1, 2, or 3 cards)	68/card	70 MHz	250 MHz	512 k states
16550A (1 or 2 cards)	102/card	100MHz	500 MHz	4 k states

*These descriptions are provided for identification purposes only. Actual performance may vary based on system configuration.

Emulation Solution

If you ordered an emulation solution, you received an emulation probe, emulation module, and accessories, which are described in the *Emulation for the MPC7400 User's Guide*.

The combination of an inverse assembler, an emulation module, and an Agilent Technologies 16700-series logic analysis system lets you both view MPC7400 assembly instructions that are executing on your target system and use the target processor's built-in JTAG debugging feature.

You can use a debugger or the logic analysis system's Emulation Control Interface to configure and control the target processor and to download program code. You can use the Agilent Technologies B4620B Source Correlation Tool Set to analyze high-level source code using the logic analysis system.

Preparing the Target System

Target System Requirements

The method for connecting the logic analyzer to the target system depends on the target system design. Broadly speaking, there are two possible approaches:

- Probe the target directly with GP probes
- Include logic analyzer connectors in the target system

The following sections contain information on designing or using these approaches. For signal-to-connector mappings, that show which signals must go to which logic analyzer connectors, see page 30.

Direct Probing with General Purpose (GP) Probes

If you are using General Purpose probes, connect the individual probes to the signals according to the tables beginning on page 30.

It is helpful to label the probe headers before installing the probes. You should connect the ground signal for the analyzer clock(s), and two to four signal grounds per pod.

Designing and using built-in connectors

You can design analyzer-compatible connectors into the target board, and connect the logic analyzer cables to these connectors according to the tables beginning on page 30. The primary concerns when using built-in connectors are:

- The board surface area required by the connectors
 - Ensuring that the logic analyzer connection is properly terminated
 - Ensuring that the microprocessor pins connect to the proper logic analyzer probes.
-

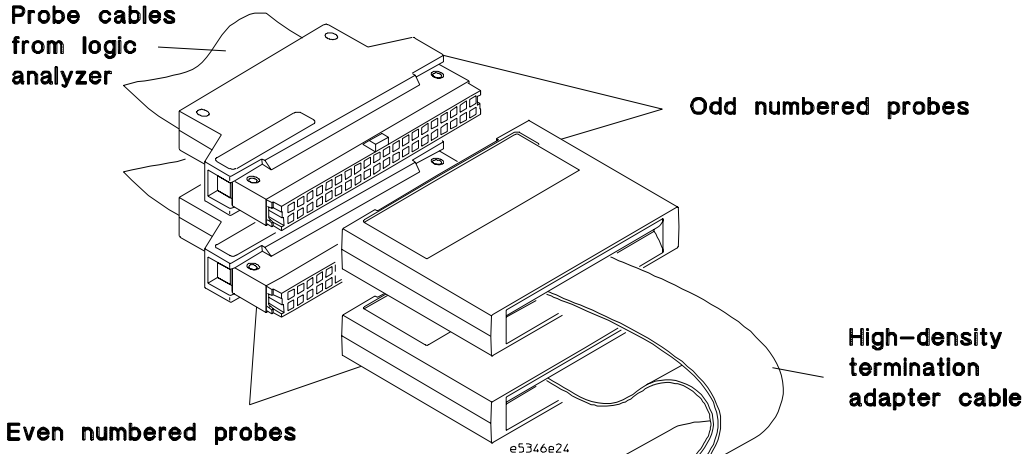
The connection scheme shown in this section uses 38-pin MICTOR connectors on the target system, and high-density termination cables to connect to the logic analyzer. Each connector and cable supports two logic analyzer pods. The part numbers for built-in connectors and cables are shown below. An illustration of the components is shown on the following page.

Part Numbers for Built-in Connectors and Cables

Part Number	Description
Agilent 1252-7431, or AMP 2-767004-2	AMP MICTOR 2 x 19 header. A minimum of four connectors (eight logic analyzer pods) are required for traditional inverse assembly; a fifth connector may be used. For “no data” inverse assembly, a minimum of two connectors (four pods) are required.
E5346-44701	Connector-support shroud
E5346A	High-density termination cable. One required for each 2x19 connector.

Chapter 2: Preparing the Target System
Designing and using built-in connectors

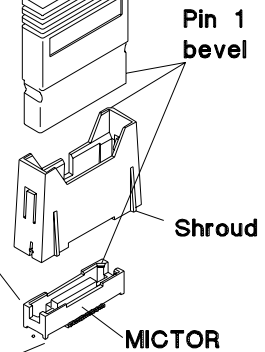
Connectors, Shroud, and High-density Termination Cables



**Top view surface mount connector
 AMP "MICTOR 38" pin assignment**

Even probe #	AC ground	Odd probe #
+5VDC 1	○	2 SCL
GND DC 3	○	4 SDA
CLK 5	○	6 CLK
D15 7	○	8 D15
D14 9	○	10 D14
D13 11	○	12 D13
D12 13	○	14 D12
D11 15	○	16 D11
D10 17	○	18 D10
D9 19	○	20 D9
D8 21	○	22 D8
D7 23	○	24 D7
D6 25	○	26 D6
D5 27	○	28 D5
D4 29	○	30 D4
D3 31	○	32 D3
D2 33	○	34 D2
D1 35	○	36 D1
D0 37	○	38 D0

**Reserved.
 Do not use.**



AMP MICTOR 38 connectors

Each MICTOR 38 connector carries 32 signals plus two clocks (CLK1 for two logic analyzer pods). The high-density termination cables are required to connect the logic analyzer cables to the connector (part number E5346A). These cables contain the required termination. One cable is required for every two logic analyzer pods.

The figure on the previous page shows the pinout for a MICTOR 38 connector. Refer to the tables following page 30 which show the microprocessor signals which should be connected to each pin. Note that the +5V pin (pin 1) is used to supply power from the logic analyzer to any active devices on an interface board. In most instances, this pin should not be used. Refer to AMP MICTOR Application Specification 114-11004 for guidelines on soldering the MICTOR connectors.

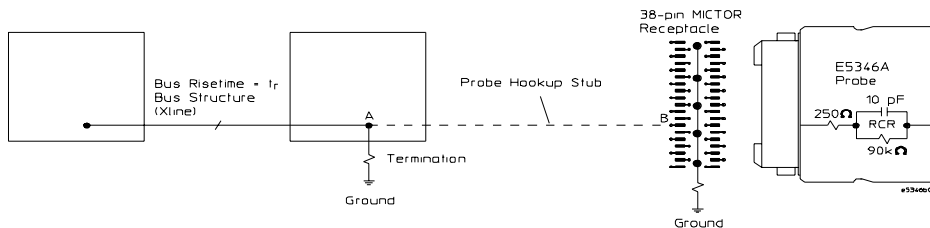
To increase the structural support for the cables, you should use cable support shrouds (part number E5346-44701) on each connector. The figures on the following page show the mechanical layouts for the shrouds and headers.

Design Considerations

The 2x19 header must be close enough to the target signal so that the stub length created is less than $\frac{1}{5}$ the t_r (bus risetime, see figure below). For PC board material, ($\epsilon_r = 4.9$) and Z_0 in the range of 50 - 80 Ω , use a propagation delay of 160 ps/inch of stub.

Each probed signal line must be able to supply a minimum of 600 mV to the probe tip and handle a minimum of 90 k Ω shunted by 10 pF. The maximum input voltage to the logic analyzer is $\pm 40V$ peak.

MICTOR 38 Connector Design Rules

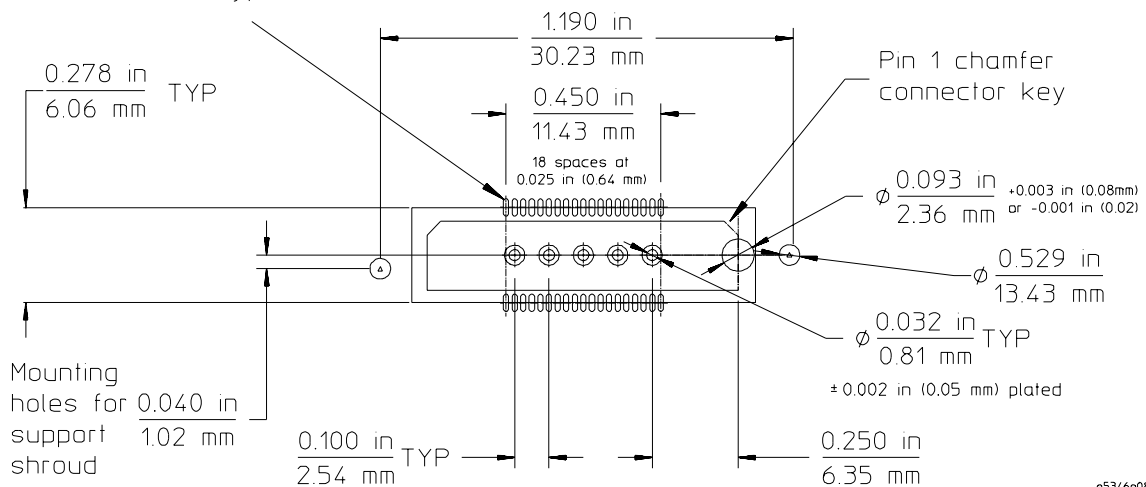


Support shroud

The support shroud (part number E5346-44701) provides additional strain relief between the connector and the high-density termination cable. The shroud requires two through-hole connections to the target board. It fits around the header, and mounts directly to the target board. The following figures show the mechanical connections for the shrouds and connectors.

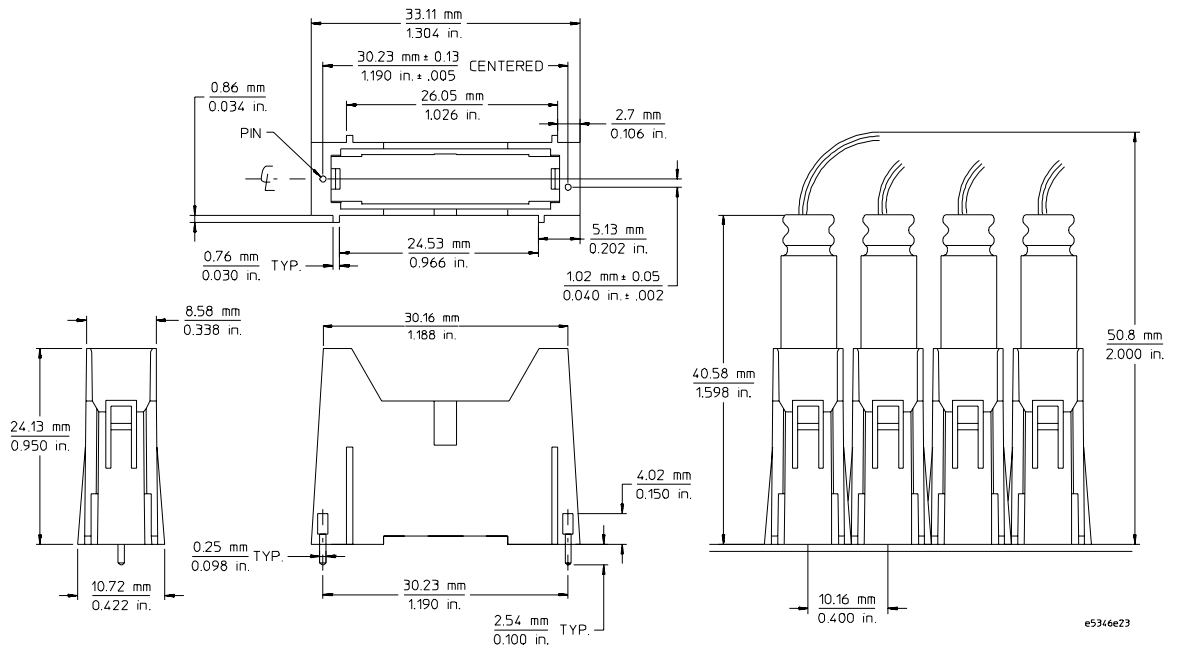
Support Shroud Mechanical Information

0.050 in X 0.017 in (1.27 mm X 0.43 mm)
 pad with 0.005 in (0.13 mm) X 45°
 corner chamfers typ 38



e5346e08

MICTOR 38 Connector Mechanical Information



Inverse assembler—signal-to-connector mapping

The following tables show the electrical signal-to-connector mapping required by the E8170B Inverse Assembler Software.

If you are using the 2x19 AMP MICTOR connectors, you must allocate the odd and even pods according to the tables in this section. (Note that the odd pods have even pin numbers, and the even pods have odd pin numbers.) The connectors and the high-density termination cables are keyed, so they will fit together only one way.

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J1 Odd			
2x19 pin	LA bit	signal name	analyzer label
6	CLK1	SYSCLK	SYSCLK
8	15	A16	ADDR
10	14	A17	ADDR
12	13	A18	ADDR
14	12	A19	ADDR
16	11	A20	ADDR
18	10	A21	ADDR
20	9	A22	ADDR
22	8	A23	ADDR
24	7	A24	ADDR
26	6	A25	ADDR
28	5	A26	ADDR
30	4	A27	ADDR
32	3	A28	ADDR
34	2	A29	ADDR
36	1	A30	ADDR
38	0	A31	ADDR

Inverse assembler—signal-to-connector mapping

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J1 Even			
2x19 pin	LA bit	signal name	analyzer label
5	CLK1	--	
7	15	A0	ADDR
9	14	A1	ADDR
11	13	A2	ADDR
13	12	A3	ADDR
15	11	A4	ADDR
17	10	A5	ADDR
19	9	A6	ADDR
21	8	A7	ADDR
23	7	A8	ADDR
25	6	A9	ADDR
27	5	A10	ADDR
29	4	A11	ADDR
31	3	A12	ADDR
33	2	A13	ADDR
35	1	A14	ADDR
37	0	A15	ADDR

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J2 Odd					
2x19 pin	LA bit	signal name	analyzer labels		
6	CLK1	QACK			QACK-
8	15	$\overline{\text{HRESET}}$	STAT		$\overline{\text{HRESET}}$ -
10	14	CKSTP_IN	STAT		CKSTP-
12	13	CKSTP_O	STAT		CHKOUT
		UT			
14	12	BR	STAT		BR-
16	11	--			
18	10	--			
20	9	WT	STAT		WT-
22	8	CI	STAT		CI-
24	7	GBL	STAT		GBL-
26	6	DBWO	STAT		DBWO-
28	5	DBG	STAT		DBG-
30	4	BG	STAT		BG-
32	3	AACK	STAT	acks	AACK-
34	2	QREQ	STAT		QREQ-
36	1	ARTRY	STAT	acks	ARTRY-
38	0	ABB	STAT		ABB-

Chapter 2: Preparing the Target System
Inverse assembler—signal-to-connector mapping

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J2 Even					
2x19 pin	LA bit	signal name	analyzer labels		
5	CLK1	--			
7	15	TSIZ0	STAT	TSIZ	
9	14	TSIZ1	STAT	TSIZ	
11	13	TSIZ2	STAT	TSIZ	
13	12	TBST	STAT	TSIZ	TBST-
15	11	TT0	STAT	TT	Atomic
17	10	TT1	STAT	TT	R/W-
19	9	TT2	STAT	TT	Invldt
21	8	TT3	STAT	TT	A Only
23	7	TT4	STAT	TT	
25	6	INT	STAT		INT-
27	5	DRTRY	STAT	acks	DRTRY
29	4	TA	STAT	acks	TA-
31	3	TEA	STAT		TEA-
33	2	SRESET-	STAT		SRESET-
35	1	TS	STAT		TS-
37	0	DBB	STAT		DBB-

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J3 Odd			
2x19 pin	LA bit	signal name	analyzer label
6	CLK1	--	
8	15	DL16	DATA_B
10	14	DL17	DATA_B
12	13	DL18	DATA_B
14	12	DL19	DATA_B
16	11	DL20	DATA_B
18	10	DL21	DATA_B
20	9	DL22	DATA_B
22	8	DL23	DATA_B
24	7	DL24	DATA_B
26	6	DL25	DATA_B
28	5	DL26	DATA_B
30	4	DL27	DATA_B
32	3	DL28	DATA_B
34	2	DL29	DATA_B
36	1	DL30	DATA_B
38	0	DL31	DATA_B

Inverse assembler—signal-to-connector mapping

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J3 Even			
2x19 pin	LA bit	signal name	analyzer label
5	CLK1	DBDIS	DBDIS-
7	15	DL0	DATA_B
9	14	DL1	DATA_B
11	13	DL2	DATA_B
13	12	DL3	DATA_B
15	11	DL4	DATA_B
17	10	DL5	DATA_B
19	9	DL6	DATA_B
21	8	DL7	DATA_B
23	7	DL8	DATA_B
25	6	DL9	DATA_B
27	5	DL10	DATA_B
29	4	DL11	DATA_B
31	3	DL12	DATA_B
33	2	DL13	DATA_B
35	1	DL14	DATA_B
37	0	DL15	DATA_B

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J4 Odd			
2x19 pin	LA bit	signal name	analyzer label
6	CLK1	--	
8	15	DH16	DATA
10	14	DH17	DATA
12	13	DH18	DATA
14	12	DH19	DATA
16	11	DH20	DATA
18	10	DH21	DATA
20	9	DH22	DATA
22	8	DH23	DATA
24	7	DH24	DATA
26	6	DH25	DATA
28	5	DH26	DATA
30	4	DH27	DATA
32	3	DH28	DATA
34	2	DH29	DATA
36	1	DH30	DATA
38	0	DH31	DATA

Inverse assembler—signal-to-connector mapping

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J4 Even			
2x19 pin	LA bit	signal name	analyzer label
5	CLK1	--	
7	15	DH0	DATA
9	14	DH1	DATA
11	13	DH2	DATA
13	12	DH3	DATA
15	11	DH4	DATA
17	10	DH5	DATA
19	9	DH6	DATA
21	8	DH7	DATA
23	7	DH8	DATA
25	6	DH9	DATA
27	5	DH10	DATA
29	4	DH11	DATA
31	3	DH12	DATA
33	2	DH13	DATA
35	1	DH14	DATA
37	0	DH15	DATA

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J5 Odd				
2x19 pin	LA bit	signal name	analyzer labels	
6	CLK1	--		
8	15	AP0	AP	
10	14	AP1	AP	
12	13	AP2	AP	
14	12	AP3	AP	
16	11	MCP		MCP-
18	10	SMI		SMI-
20	9	--		
22	8	--		
24	7	--		
26	6	--		
28	5	--		
30	4	LSSDMODE		LSSDM0
32	3	PLLCF0	PLLCFG	
34	2	PLLCF1	PLLCFG	
36	1	PLLCF2	PLLCFG	
38	0	PLLCF3	PLLCFG	

Chapter 2: Preparing the Target System
Inverse assembler—signal-to-connector mapping

E8170B Inverse Assembler for the MPC7400 Logic Analyzer Interface Signal List Connector J5 Even			
2x19 pin	LA bit	signal name	analyzer labels
5	CLK1	--	
7	15	L1TSTCLK	L1Tclk
9	14	L2TSTCLK	L2Tclk
11	13	--	
13	12	--	
15	11	--	
17	10	RSRV	RSRV-
19	9	TBEN	TBEN
21	8	TBLISYNC	TBLISY
23	7	DP0	DP
25	6	DP1	DP
27	5	DP2	DP
29	4	DP3	DP
31	3	DP4	DP
33	2	DP5	DP
35	1	DP6	DP
37	0	DP7	DP

Designing a JTAG Connector into Your Target System

For information on designing a JTAG connector into your target system, see the emulation manual supplied with your emulation probe/module.

Chapter 2: Preparing the Target System
Inverse assembler—signal-to-connector mapping

Setting Up the Logic Analysis System

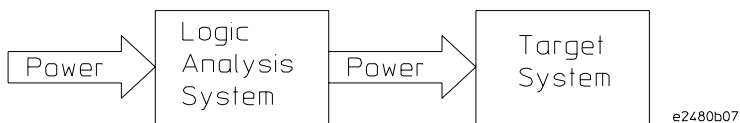
Power-on/Power-off Sequence

Listed below are the sequences for powering on and off a fully connected system. Simply stated, your target system is always the last to be powered on, and the first to be powered off.

To power on 16700-series logic analysis systems

Ensure the target system is powered off.

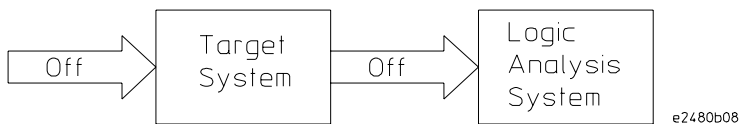
- 1 Turn on the logic analyzer. The Setup Assistant will guide you through the process of connecting and configuring the logic analyzer.
- 2 When the target system is connected to the logic analyzer, and everything is configured, turn on your target system.



To power off

Turn off power to your system in the following order:

- 1 Turn off your target system.
- 2 Turn off your logic analysis system.



Installing Logic Analyzer Modules

You should install logic analyzer, oscilloscope, or pattern generator modules in your logic analysis system before you install an emulation module (if applicable) and software.

CAUTION:

Electrostatic discharge (ESD) can damage electronic components. Use appropriate ESD equipment (grounded wrist strap, etc.) and ESD-safe procedures when you handle and install modules.

Refer to your logic analysis system's *Installation Guide* for instructions on installing logic analyzer modules.

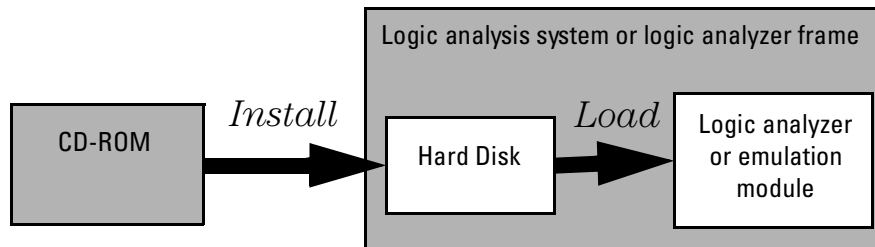
Installing the Emulation Module

If you ordered an emulation module as part of your HP/Agilent 16700-series logic analysis system, it is already installed in the system frame.

If you ordered your emulation module separately, then see the user's guide you received with your emulation module for installation instructions.

Installing and loading software

Installing the software will copy the files to the hard disk of your logic analysis system. Later, you will need to **load** some of the files into the appropriate measurement module.



What needs to be installed

If you ordered an inverse assembler or emulation solution with your logic analysis system, the software was installed at the factory.

The following files are installed when you install a processor support package from the CD-ROM:

- Logic analysis system configuration files
- Inverse assembler (automatically loaded with the configuration files)
- Personality files for the Setup Assistant
- Emulation module firmware (for emulation solutions)
- Emulation Control Interface (for emulation solutions)

The B4620B Source Correlation Tool Set is installed with the logic analysis system's operating system. A password may be required to enable the tool set. Follow the instructions on the entitlement certificate.

To install the software from CD-ROM


Installing a processor support package from a CD-ROM will take just a few minutes. If the processor support package requires an update to the Agilent Technologies 16700 operating system, installation may take approximately 15 minutes.

If the CD-ROM drive is not connected, see the instructions printed on the CD-ROM package.

- 1 Turn on the CD-ROM drive first and then turn on the logic analysis system.

If the CD-ROM and analysis system are already turned on, be sure to save any acquired data. The installation process may reboot the logic analysis system.

- 2 Insert the CD-ROM in the drive.

- 3 Select the **System Administration** icon. 

- 4 Select the **Software Install** tab.

- 5 Select **Install...**

Change the media type to “CD-ROM” if necessary.

- 6 Select **Apply**.

- 7 From the list of types of packages, double-click “PROC-SUPPORT.”

NOTE:

For touch screen systems, double select the “PROC-SUPPORT” line by quickly touching it twice.

A list of the processor support packages on the CD-ROM will be displayed.

- 8 Select on the “mpc74xx” package.

If you are unsure whether this is the correct package, select **Details** for information about the contents of the package.

- 9 Select **Install**.

The Continue dialog box will appear.

- 10 Select **Continue**.

Installing and loading software

The Software Install dialog will display “Progress: completed successfully” when the installation is complete.

- 11** If required, the system will automatically reboot. Otherwise, close the software installation windows.

The configuration files are stored in `/logic/configs/hp/mpc74xx`. The inverse assemblers are stored in `/logic/ia`.

See Also

The instructions printed on the CD-ROM package for a summary of the installation instructions.

The online help for more information on installing, licensing, and removing software.

To list software packages which are installed

- In the System Administration window, go to the **Software Install** tab and select **List...**

Connecting the Logic Analyzer to the Target System

Connecting the Logic Analyzer to the Target System

This chapter contains instructions for connecting different logic analyzers to your target system.

If you have designed connectors into the target system as described in Chapter 2, “Preparing the Target System,” use the Setup Assistant to connect and configure your system (see page 17).

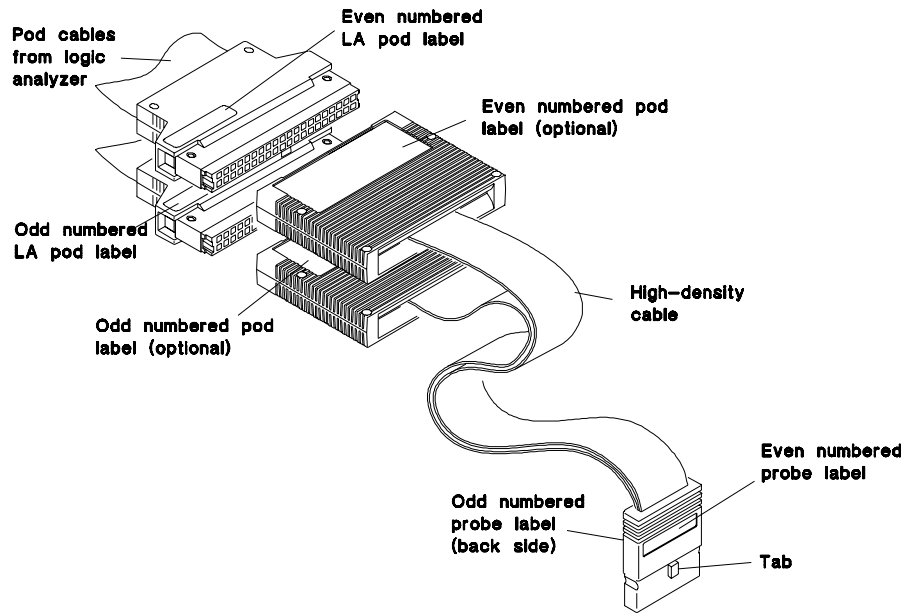
If you are not using the Setup Assistant, follow the instructions given in this chapter. This chapter covers the following tasks; the order shown here is the recommended order for performing these tasks:

- Check that the target system meets the necessary requirements (see page 24)
- Read the power on/power off sequence (see page 44)
- Connect the target system to the logic analyzer (see page 51)
- Configure the logic analyzer (see page 63)

To connect the high-density termination cables to the target system

The Agilent E5346A 2x19 high-density termination cables include labels to identify them. The labels can be attached to the cables after the cables have been connected to the target system and logic analyzer, as shown in the following illustration.

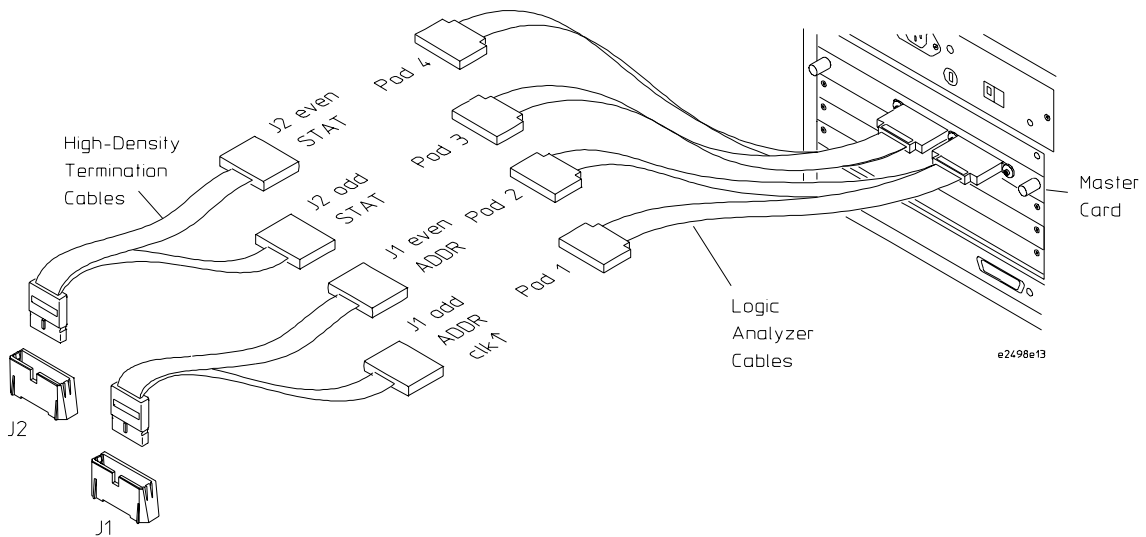
E5346A Cable Numbering



e2498e08

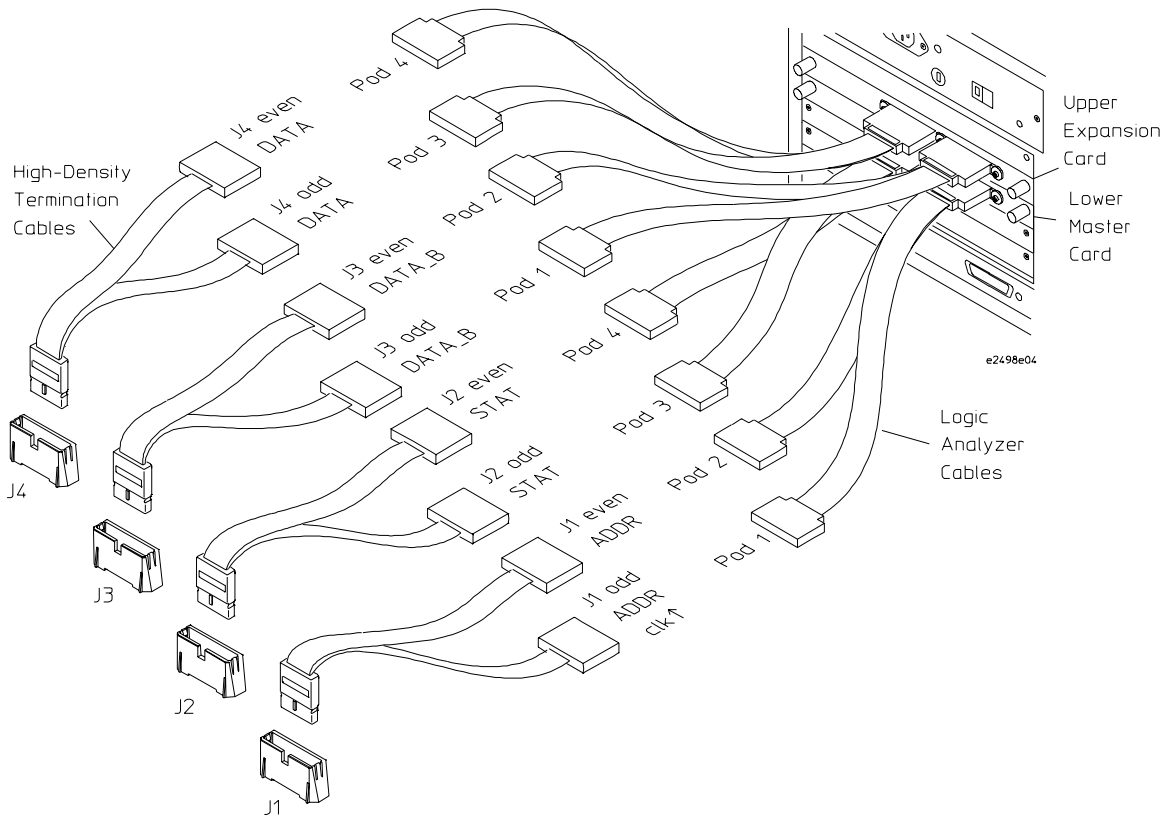
To connect to the 16715/16/17/18/19/50/51/52A logic analyzer (one card)

Use the figure below to connect the target system to the 16715/16/17/18/19/50/51/52A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



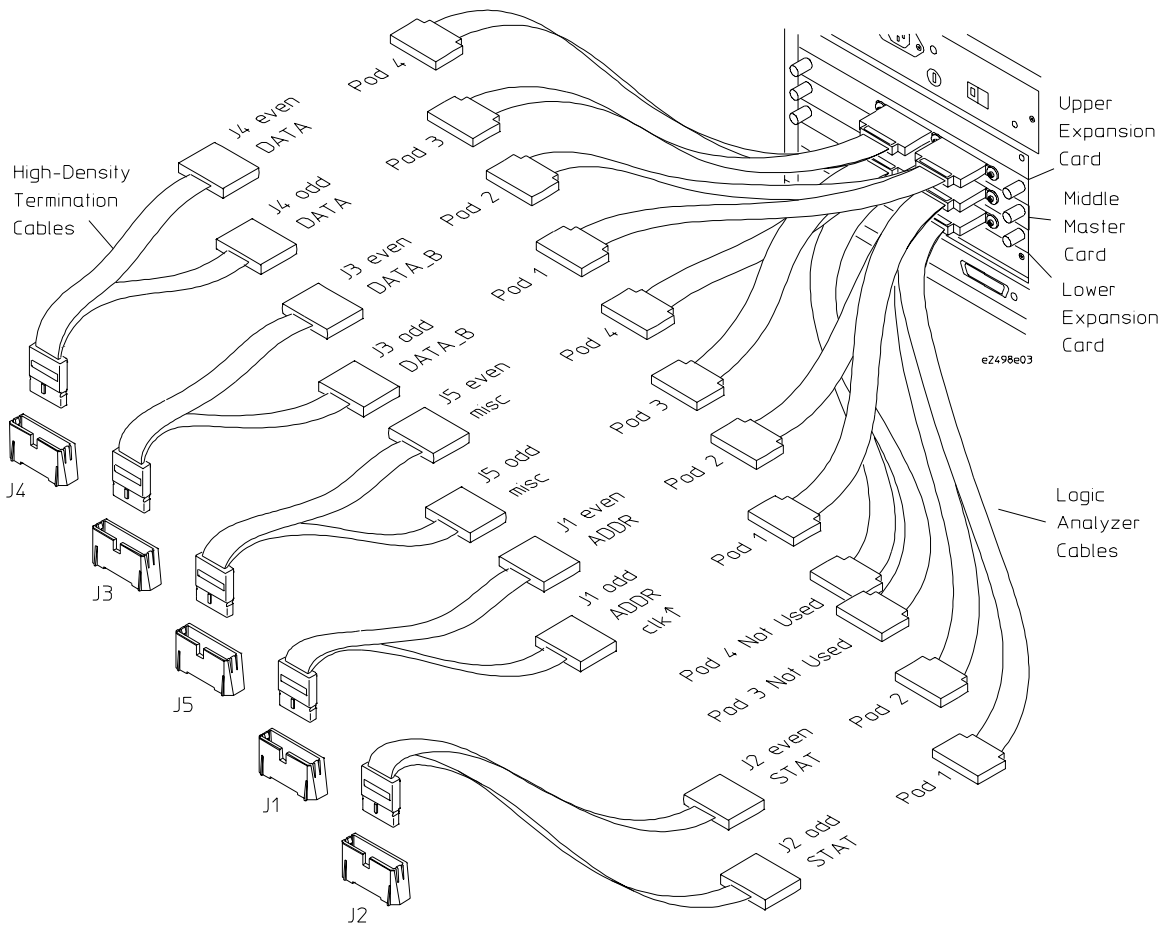
To connect to the 16715/16/17/18/19/50/51/52A logic analyzer (two cards)

Use the figure below to connect the target system to the 16715/16/17/18/19/50/51/52A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



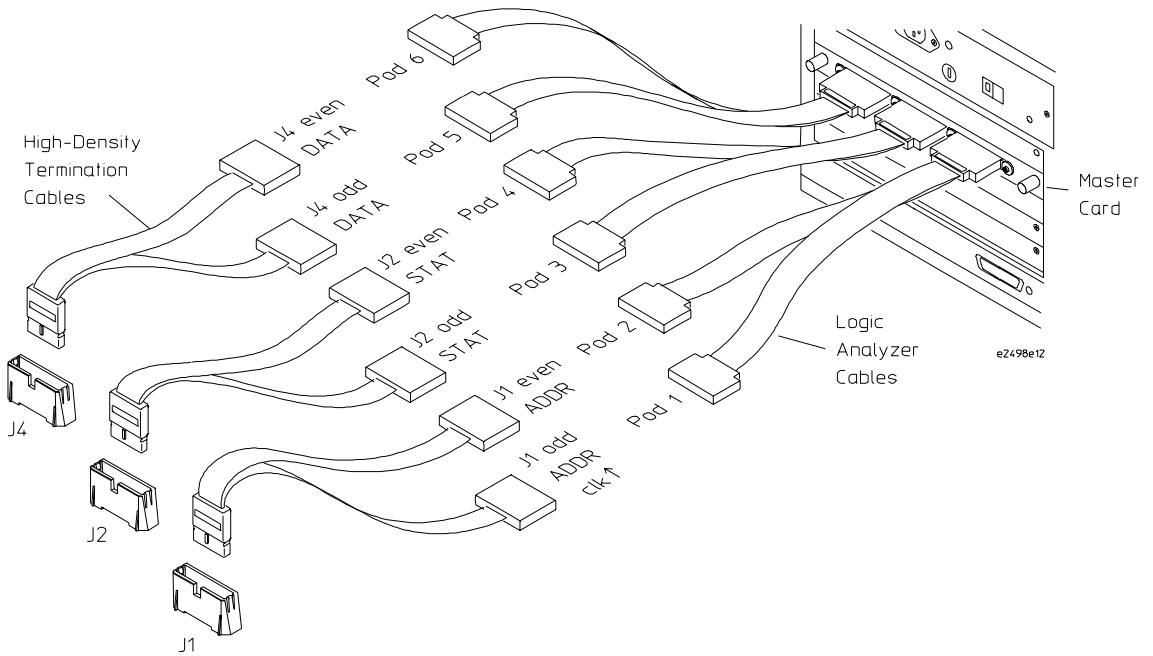
To connect to the 16715/16/17/18/19/50/51/52A logic analyzer (three cards)

Use the figure below to connect the target system to the 16715/16/17/18/19/50/51/52A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



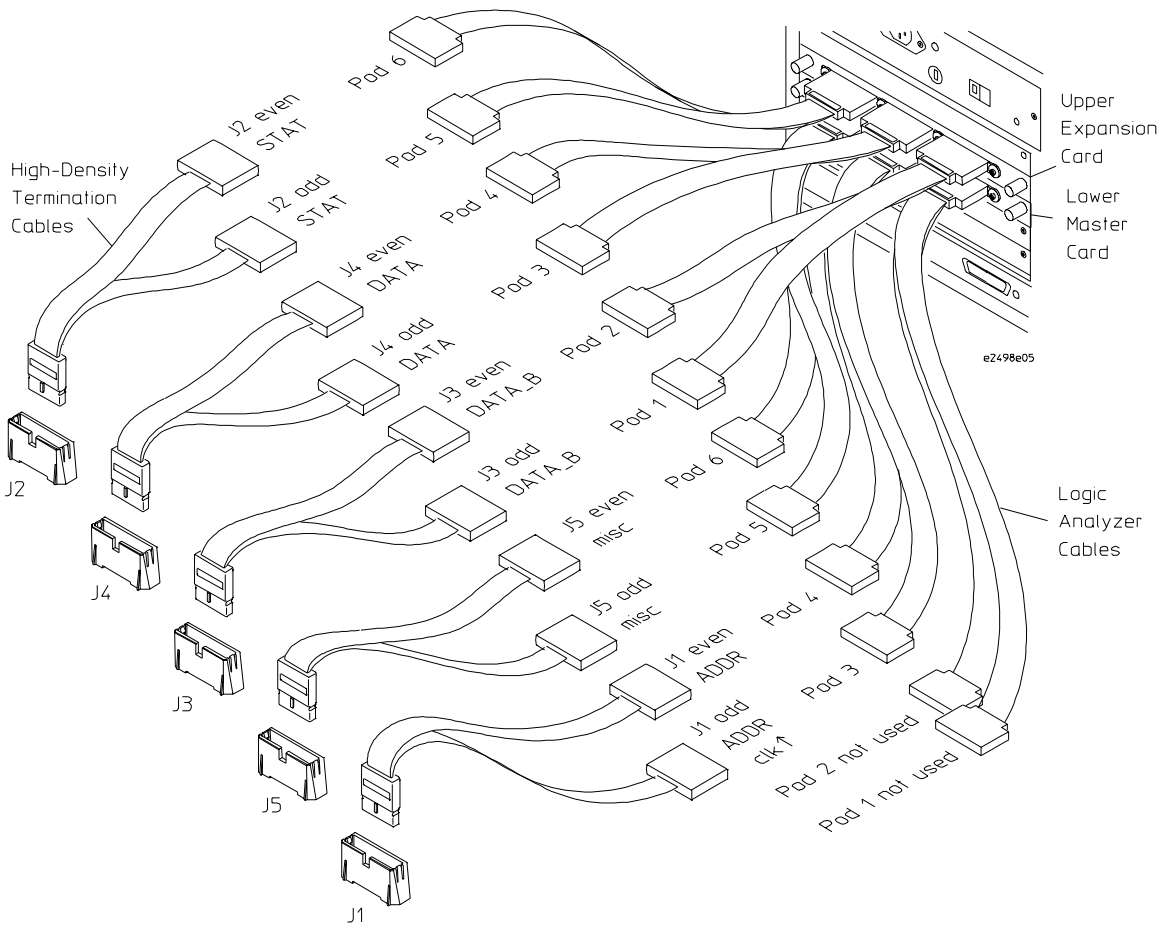
To connect to the 16710/11/12A logic analyzer (one card)

Use the figure below to connect the target system to the 16710/11/12A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



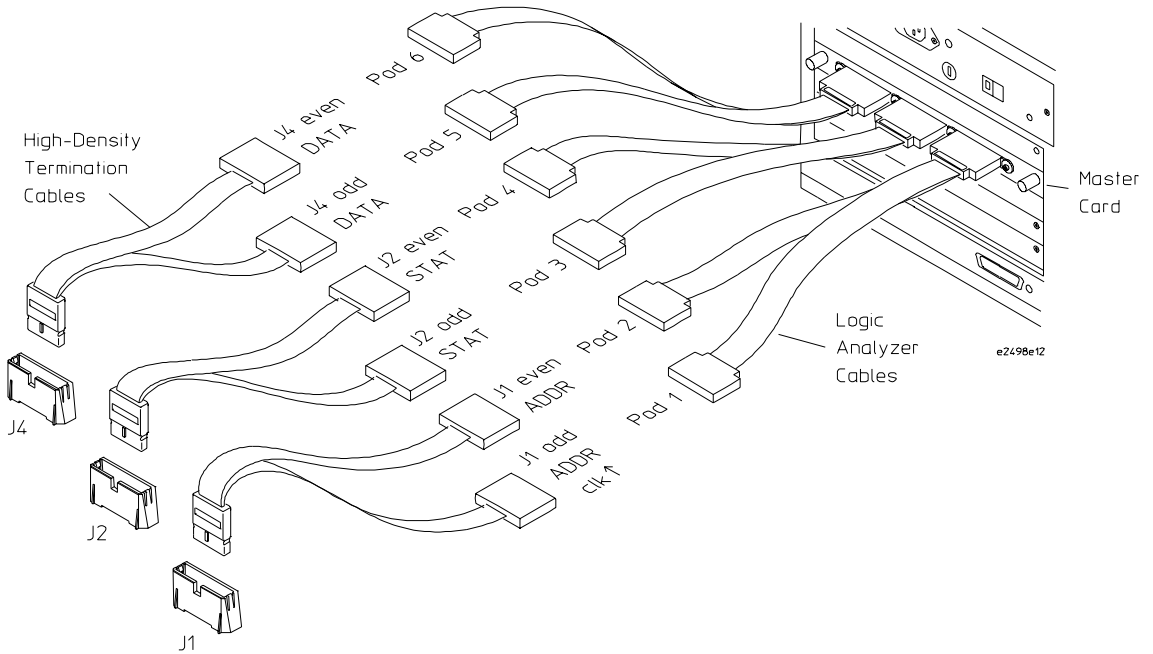
To connect to the 16710/11/12A logic analyzer (two cards)

Use the figure below to connect the target system to the 16710/11/12A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



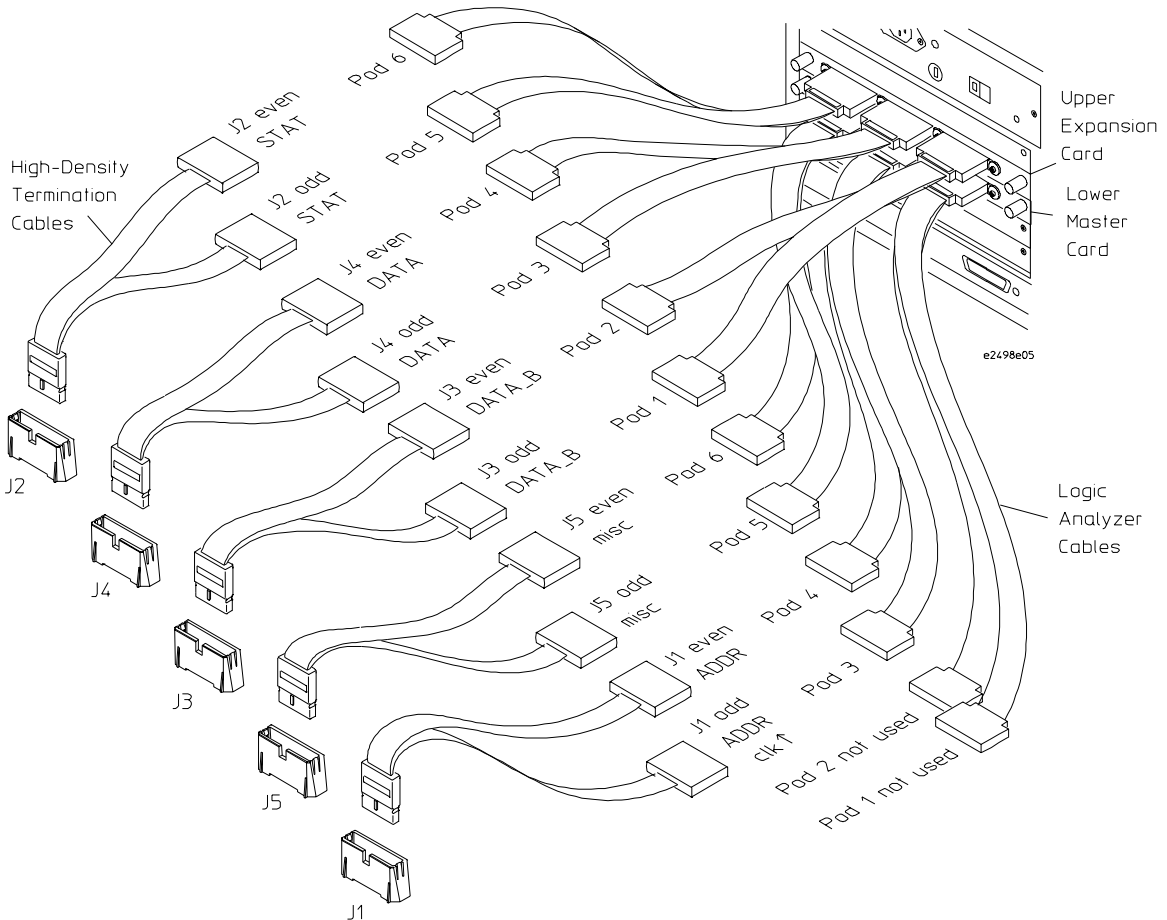
To connect to the 16550A analyzer (one card)

Use the figure below to connect the target system to the 16550A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



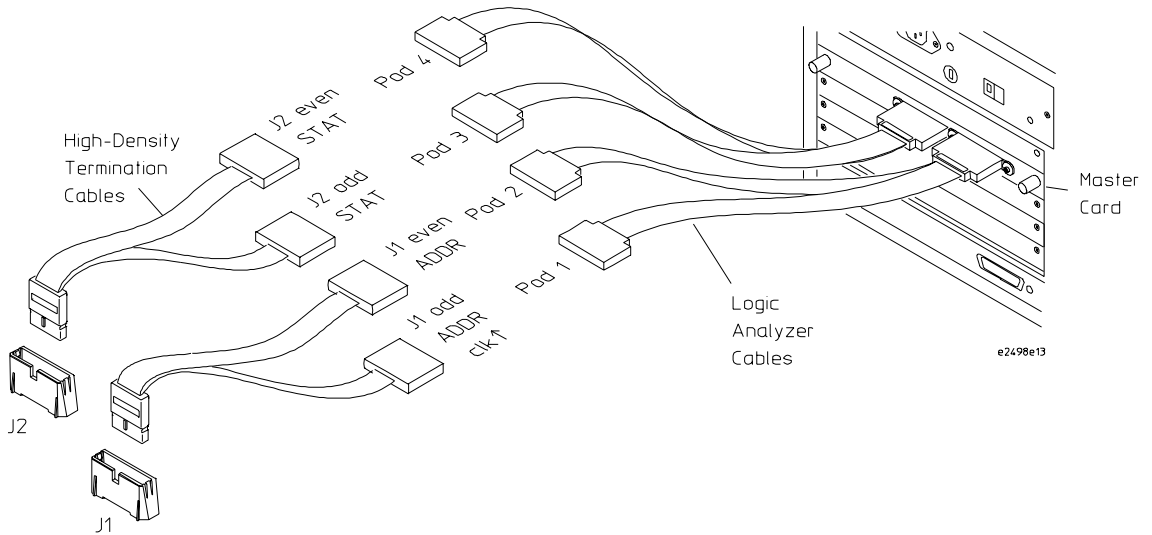
To connect to the 16550A analyzer (two cards)

Use the figure below to connect the target system to the two-card 16550A logic analyzers. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



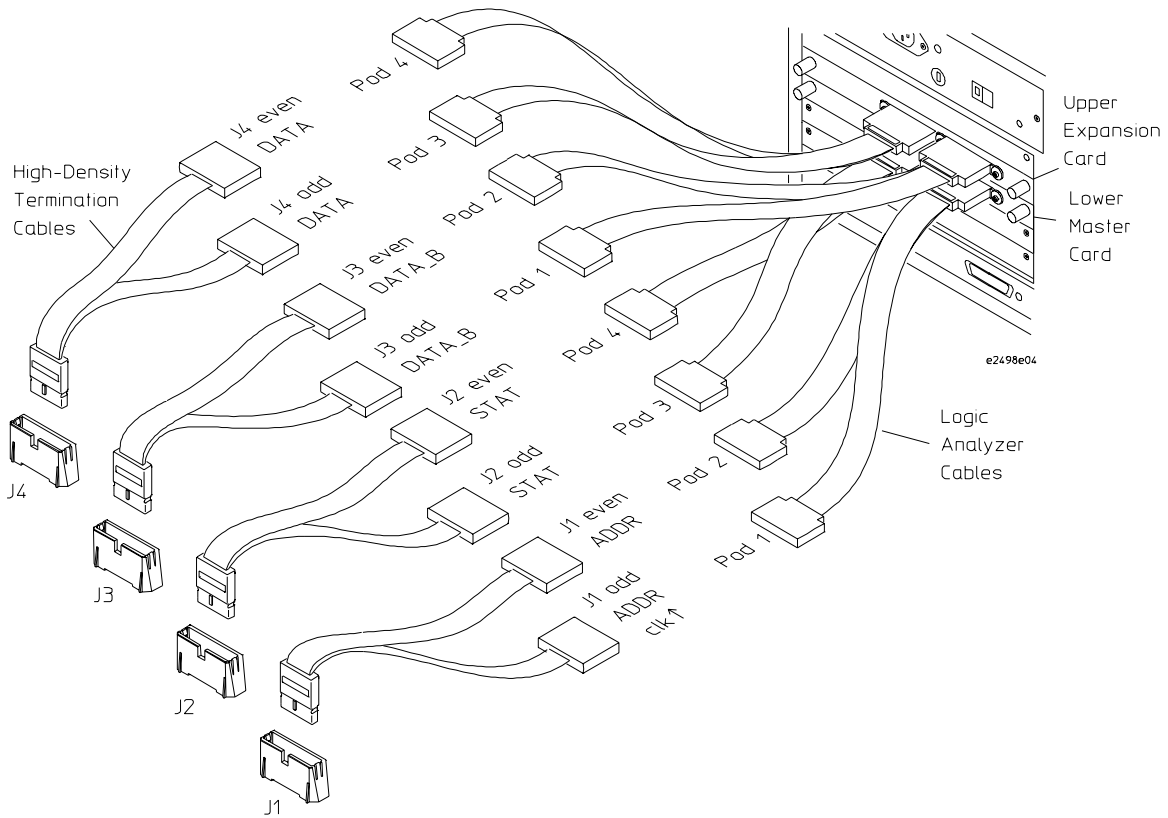
To connect to the 16554/55/56/57 (one-card)

Use the figure below to connect the target system to the two-card 16554/55/56/57 logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



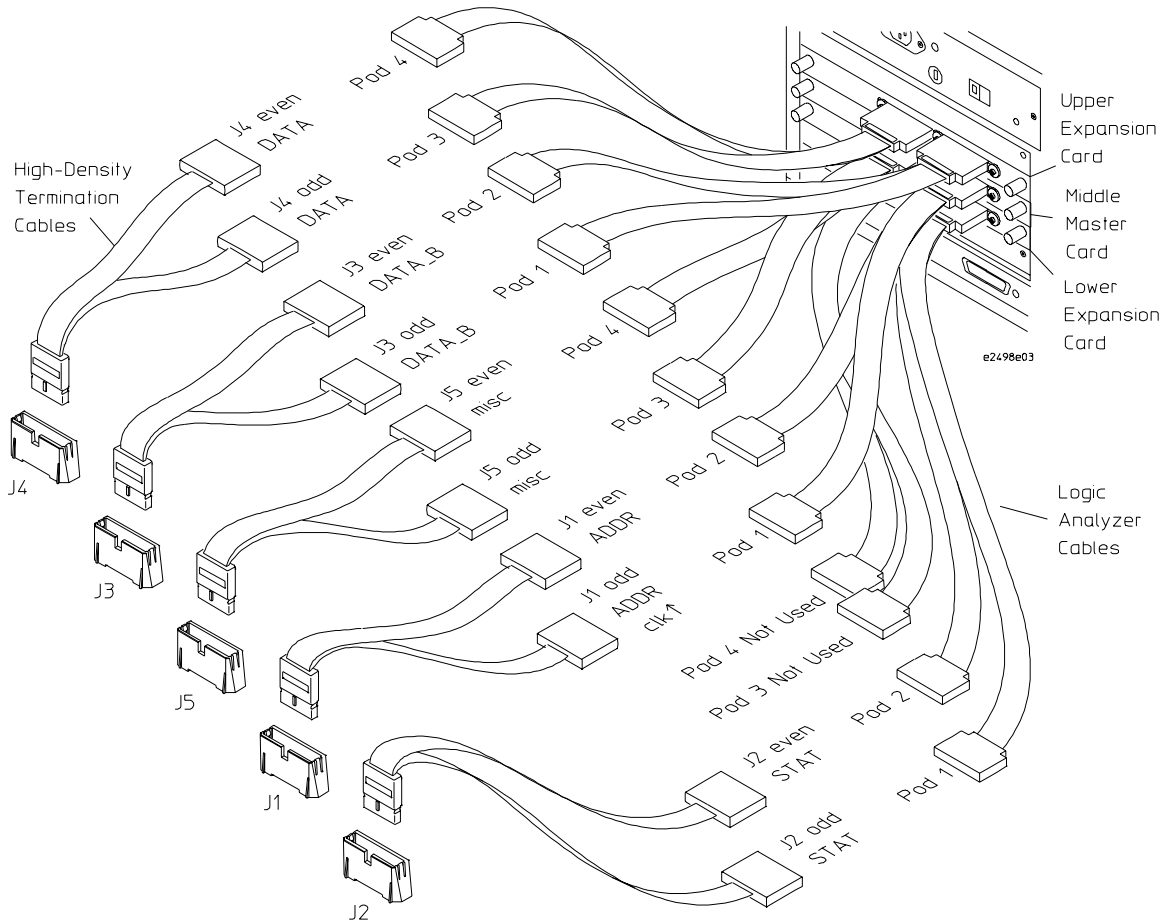
To connect to the 16554/55/56/57 (two-cards)

Use the figure below to connect the target system to the two-card 16554/55/56/57 logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



To connect to the 16554/55/56/57 (three-cards)

Use the figure below to connect the target system to the 16554/55/56/57 logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



Configuring the Logic Analyzer

This chapter describes setting up and using the MPC7400 inverse assembler.

The information in this chapter includes:

- Loading configuration files and the inverse assembler
- Using the format menu
- Using symbols

Configuring 16700-series Logic Analysis Systems

You configure the logic analyzer by loading a configuration file. Normally, this is done using the Setup Assistant (see page 17). If you did not use the Setup Assistant, you can load the configuration and inverse assembler files from the logic analysis system hard disk.

The information in the configuration file includes:

- Label names and channel assignments for the logic analyzer
- Workspace setup
- Inverse assembler file name
- Inverse assembler preference settings

The configuration file you use is determined by the logic analyzer you are using

It is strongly recommended that you do not change the setup related to the MPC7400 sampling, format, pod assignment or configuration dialogs. The configuration file (loaded by the Setup Assistant in 16700-series logic analysis systems) will configure the logic analyzer for making measurements of the MPC7400.

To load configuration files (and the inverse assembler) from the system hard disk

The easiest way to load configuration and inverse assembler files is by using the Setup Assistant. If you choose to use Setup Assistant, it will load the configuration file and inverse assembler for you. See page 17.

- 1 Click on the File Manager icon. Use File Manager to ensure that the subdirectory `/logic/configs/hp/ppc74xx/` exists.

If the above directory does not exist, you need to install the POWERPC7400 Processor Support Package. Close File Manager, then use the procedure on the CD-ROM jacket to install the POWERPC7400 Processor Support Package before you continue.

- 2 Using File Manager, select the configuration file that you want to load from the `/logic/configs/hp/ppc74xx/` directory, then select **Load**. If you have more than one logic analyzer installed in your logic analysis system, use the **Target** field to select the machine you want to load.

The logic analyzer is configured for MPC7400 analysis by loading the appropriate MPC7400 configuration file. Loading the indicated file also automatically loads the correct inverse assembler.

- 3 Close File Manager.

Logic Analyzer Configuration Files

The following table lists the configuration files for the MPC7400 for each supported logic analyzer card configuration.

Logic Analyzer Configuration Files

Analyzer Model	Analyzer Description*	Configuration File $V_{I/O} = 3.3V$	Configuration File $V_{I/O} = 1.8V$
16750/51/52A (one card)	400 MHz STATE 2 GHz TIMING ZOOM	C7400L0	C7400_AVCL0
16750/51/52A (two cards)	400 MHz STATE 2 GHz TIMING ZOOM	C7400L2	C7400_AVCL2
16750/51/52A (three cards)	400 MHz STATE 2 GHz TIMING ZOOM	C7400L3	C7400_AVCL3
16717/18/19A (one card)	333 MHz STATE 2 GHz TIMING ZOOM	C7400L0	C7400_AVCL0
16717/18/19A (two cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7400L2	C7400_AVCL2
16717/18/19A (three cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7400L3	C7400_AVCL3
16716A (one card)	167 MHz STATE 667 MHz TIMING ZOOM	C7400L0	C7400_AVCL0
16715A (one card)	167 MHz STATE 2 GHz TIMING ZOOM		
16716A (two cards)	167 MHz STATE 667 MHz TIMING ZOOM	C7400L2	C7400_AVCL2
16715A (two cards)	167 MHz STATE 2 GHz TIMING ZOOM		
16716A (three cards)	167 MHz STATE 667 MHz TIMING ZOOM	C7400L3	C7400_AVCL3
16715A (three cards)	167 MHz STATE 2 GHz TIMING ZOOM		
16710/11/12A (one card)	100 MHz STATE 500 MHz TIMING	C7400F1	C7400_AVCF1
16710/11/12A (two cards)	100 MHz STATE 500 MHz TIMING	C7400F2	C7400_AVCF2
16557D (one card)	140 MHz STATE 500 MHz TIMING	C7400M0	C7400_AVCM0

Analyzer Model	Analyzer Description*	Configuration File $V_{I/O} = 3.3V$	Configuration File $V_{I/O} = 1.8V$
16557D (two cards)	140 MHz STATE 500 MHz TIMING	C7400M2	C7400_AVCM2
16557D (three cards)	140 MHz STATE 500 MHz TIMING	C7400M3	C7400_AVCM3
16556A/D (one card)	100 MHz STATE 400 MHz TIMING	C7400M0	C7400_AVCM0
16556A/D (two cards)	100 MHz STATE 400 MHz TIMING	C7400M2	C7400_AVCM2
16556A/D (three cards)	100 MHz STATE 400 MHz TIMING	C7400M3	C7400_AVCM3
16555A/D (one card)	110 MHz STATE 500 MHz TIMING	C7400M0	C7400_AVCM0
16555A/D (two cards)	110 MHz STATE 500 MHz TIMING	C7400M2	C7400_AVCM2
16555A/D (three cards)	110 MHz STATE 500 MHz TIMING	C7400M3	C7400_AVCM3
16554A (one card)	70 MHz STATE 250 MHz TIMING	C7400M0	C7400_AVCM0
16554A (two cards)	70 MHz STATE 250 MHz TIMING	C7400M2	C7400_AVCM2
16554A (three cards)	70 MHz STATE 250 MHz TIMING	C7400M3	C7400_AVCM3
16550A (one card)	100 MHz STATE 500 MHz TIMING	C7400F1	C7400_AVCF1
16550A (two cards)	100 MHz STATE 500 MHz TIMING	C7400F2	C7400_AVCF2

*These descriptions are provided for identification purposes only. Actual performance may vary based on system configuration.

NOTE:

Use the Setup Assistant for logic analyzer configuration when possible. See page 17 for instructions for using the Setup Assistant.

NOTE:

The threshold voltage in the $V_{I/O} = 1.8V$ configuration files is set to 900mV instead of TTL levels.

Using the Format Menu

This section describes the organization of MPC7400 signals in the logic analyzer's Format menu.

The configuration files contain predefined format specifications. These format specifications include all labels for monitoring the microprocessor.

Do not modify the ADDR, ADDR_B, DATA, DATA_B, STAT or STAT_B labels in the format specification if you want inverse assembly. Changes to these labels may cause incorrect or incomplete inverse assembly.

Bit ordering conventions

The HP/Agilent logic analyzers and the PowerPC use opposite conventions to designate individual signals on a bus. In PowerPC nomenclature, bit 0 is the most significant; in the logic analyzers, bit 0 is the least significant. In PowerPC, A0 is the most significant bit of the address bus; on the analyzer, this bit is called ADDR31.

Most Significant	Least Significant
A0	A31 <i>PowerPC</i>
ADDR31	ADDR0 <i>Logic Analyzer</i>

This may cause confusion in the waveform window when using Channel Mode Sequential or Individual.

Loading Symbol Information

Symbols represent values in measurements. For example, the symbol INTERRUPT might represent the value 1FF04000 found on the ADDR label, the address where your interrupt handler begins. Symbols are more easily recognized than hexadecimal address values in logic analyzer trace displays, and they are easier to remember when setting up triggers.

Agilent Technologies logic analyzers let you assign user-defined symbol names to particular label values, or you can download symbols from certain object file formats.

When source file line number symbols are downloaded to the logic analyzer, you can set up triggers on source lines using the B4620B Source Correlation Tool Set. The B4620B Source Correlation Tool Set also lets you display the high-level source code associated with captured data.

Three symbol sources may be used in the logic analyzer:

- Predefined MPC7400 symbols
- User-defined symbols
- Object-file symbols

To view predefined MPC7400 symbols

The MPC7400 logic analyzer configuration files include predefined symbols. The predefined symbols for the MPC7400 are listed in the following table.

These symbols appear along with the user-defined symbols in the logic analyzer.

To view the predefined symbols:

- 1 Open the logic analyzer's **Setup** window.
- 2 Select the **Symbol** tab.
- 3 Select the **User Defined** tab.
- 4 Choose a label name from the **Label** list.

The logic analyzer will display the symbols associated with the label.

Predefined Logic Analyzer Symbol Descriptions

Label	Symbol	Encoding
acks	idle	1111
	ARTRY	xxx0
	DRTRY	0xxx
	TA AACK	x00x
	AACK	xx0x
	TA	x0xx
R/-W	rd	1
	wr	0
TSIZ †	burst	xxx0
	8 byte	0001
	1 byte	0011
	2 byte	0101
	3 byte	0111
	4 byte	1001
	5 byte	1011
	6 byte	1101
7 byte	1111	
TT	Kill Block	01100
	Wr Graphics	10100
	Rd Graphics	11100
	Clean Block	00000
	Write	0001
	Wr/Kill	00110
	Read	01010
	Rd/Flush	0111
	Wr/Atomic Flush	1001
	Read Atomic	1101
	Rd/Flush Atomic	1111
	Flush Block	00100
	DSYNC	01000
	eieio	10000
(reserved)	1011	
TLB Invalidate	11000	
STAT	inst fetch xxxx xxxx xxxx xxx1 xxxx 1xxx xxxx 0xxx	

† The least significant bit of the TSIZ label is the TBST- signal.

To create user defined symbols

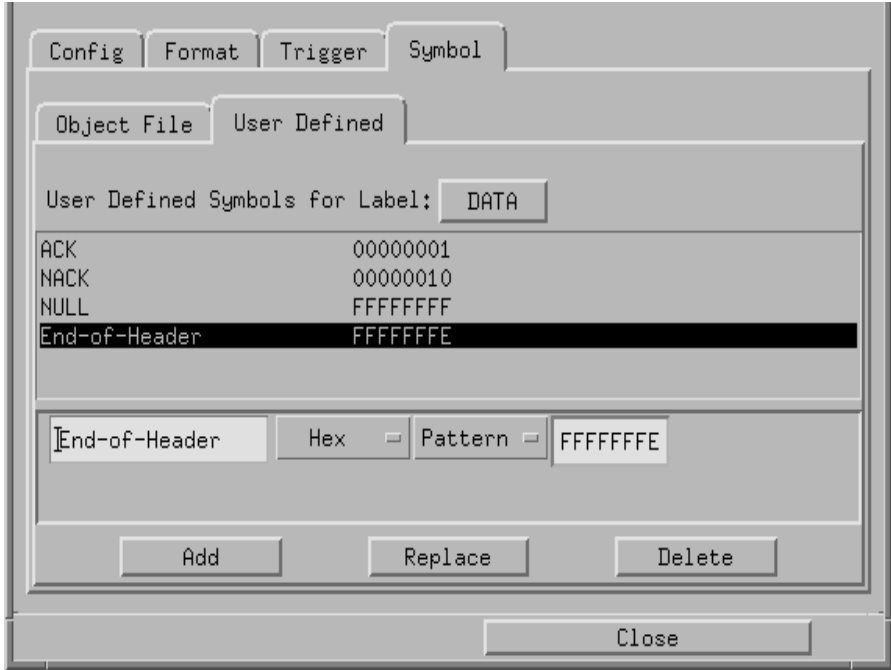
User-defined symbols are symbols you create in the logic analyzer by assigning symbol names to label values. Typically, you assign symbol names to address label values, but you can define symbols for data, status, or other label values as well.

User-defined symbols are saved with logic analyzer configurations.

To create user-defined symbols:

- 1** Open the logic analyzer's Setup window.
- 2** Select the **Symbol** tab.
- 3** Select the **User Defined** tab.
- 4** Choose a label name from the **Label** list.
- 5** Enter the new symbol name and value.
- 6** Select **Add**.

The screen below shows a set of symbols for values found on a DATA label.



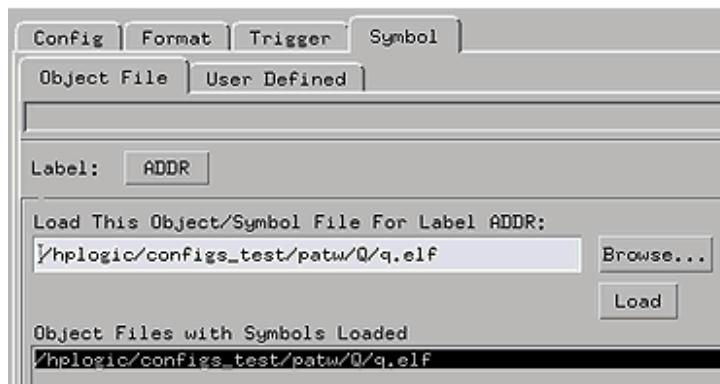
To load object file symbols

The most common way to load program symbols into the logic analyzer is from an object file that is created when the program is compiled. The object file containing symbolic debug information must be in a format the logic analyzer understands.

If your compiler generates files in a format that the logic analyzer doesn't understand, you can use a General-Purpose ASCII (GPA) symbol file *(see Chapter 9, "General-Purpose ASCII (GPA) Symbol File Format," beginning on page 133).

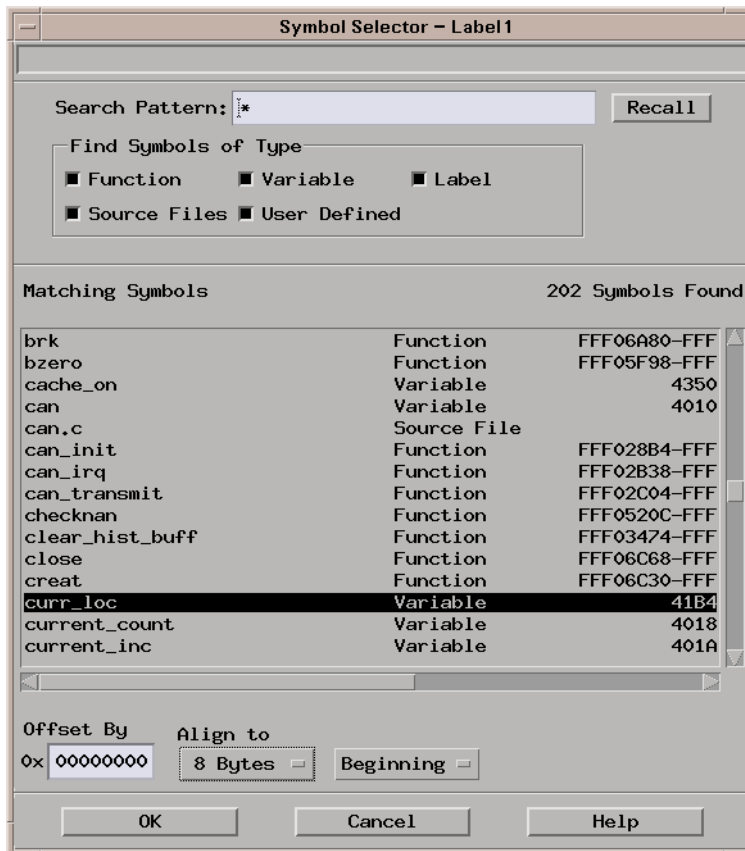
To load symbols in the 16700-series logic analysis system:

- 1 Open the logic analyzer module's **Setup** window.
- 2 Select the **Symbol** tab.
- 3 Select the **Object File** tab.
- 4 Make sure the label is ADDR, then select object files and load their symbol information.



When you load object file symbols into the logic analyzer, a database of symbol/line number to address assignments is generated from the object file.

The Symbol Selector dialog allows you to view the database so you can find a symbol to use in place of a hexadecimal value when defining trigger patterns, trigger ranges, and so on.



Symbol use requirements

In order for symbols and source code to be accurately assigned to address values captured by the logic analyzer, you need:

An accurate bus trace

The E8170B inverse assembler provides MPC7400 microprocessor data when the logic analyzer is properly connected to the target system.

Direct address translation

The Memory Management Unit must perform direct address translation. Otherwise, captured addresses may not be correlated to the correct symbols.

An inverse assembler for trace lists

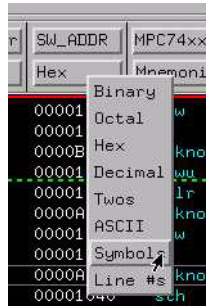
The MPC7400 inverse assembler decodes captured data into program counter (PC) addresses (also known as software addresses) and assembly language mnemonics.

A symbol file

You need an object file containing symbolic debug information in a format the logic analyzer understands. Alternatively, you can use a General Purpose ASCII (GPA) symbol file (see page 133).

To display symbols

- Over a Listing display's label base, right-click the mouse button, and select **Symbols**.



Any symbols that have been defined will be displayed for equivalent captured values.

Loading Symbol Information

Capturing Processor Execution

Chapter 6: Capturing Processor Execution

The normal steps in using the logic analyzer are:

1. Configure the logic analyzer.
2. Format labels for the logic analyzer channels (that is, mapping logic analyzer channels to target system signal names).
3. Load symbols from the program's object file.
4. Set up the trigger, and run the measurement.
5. Display the captured data.

The logic analyzer is configured and labels are created (formatted) for the logic analysis channels when configuration files are loaded. See “Configuring the Logic Analyzer” on page 63.

You can load program object file symbols into the logic analyzer when configuring it. See “To load object file symbols” on page 74.

This chapter describes setting up logic analyzer triggers when using the inverse assembler and/or the B4620B source correlation tool set.

See Chapter 7, “Displaying Captured MPC7400 Execution,” beginning on page 91 for information on displaying captured data.

To Set Up Logic Analyzer Triggers

Triggering allows the logic analyzer to store the data states that you want to see, ensuring quicker analysis of the stored data.

You can also specify which states that are stored in the logic analyzer. The Trigger sequence is set up by the software to store all states.

CAUTION:

If you modify the trigger sequence to store only selected bus cycles, incorrect or incomplete disassembly may be displayed.

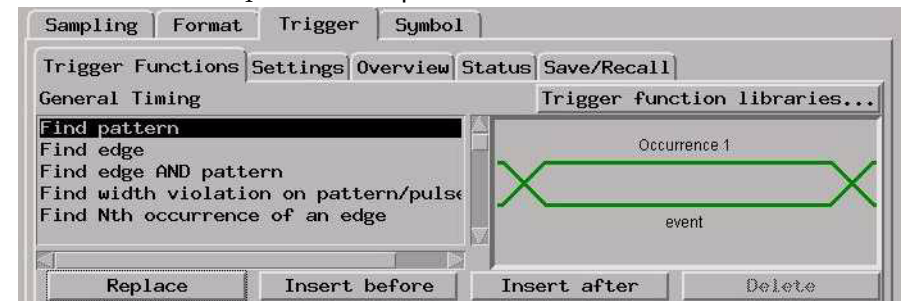
- 1 Open the logic analyzer's Setup window.



- 2 Select the Trigger tab.

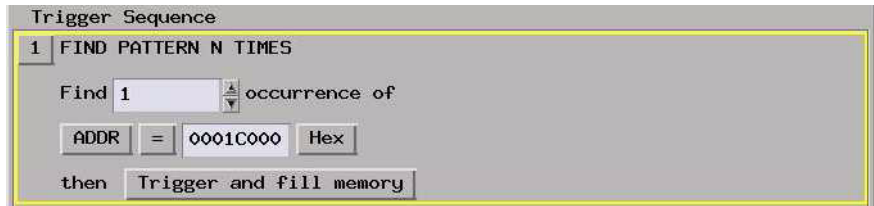


- 3 Select the trigger function that will be used in the logic analysis measurement and press the Replace button.



Chapter 6: Capturing Processor Execution
To Set Up Logic Analyzer Triggers

4 Set up the trigger sequence.



5 Run the measurement.



See Also

See the HP/Agilent 16700-series logic analysis system's on-line help for more information on setting up logic analyzer triggers.

To Setup Trigger Alignment and Offset for Symbols and Source Code

When setting up trigger specifications to capture MPC7400 execution:

- Use the logic analyzer trigger alignment to avoid missed triggers.
- Use the logic analyzer address offset to compensate for relocated code.
- Use the logic analyzer storage qualification to capture the software execution you're interested in and filter out library code execution (whose source file lookups can take a long time if the library source code is not available).

Using trigger alignment

The MPC7400 64-bit data bus can cause missed triggers on some instructions. You should use an 8-byte alignment to avoid missed triggers. Instructions for the MPC7400 are 32 bits long and must be located on even address boundaries. This means that an instruction will often be fetched as the lower 32 bits of one 64-bit memory cycle. When this happens, the address of the instruction in the lower 32 bits of the fetch will not be seen on the address bus. If a trigger was set to occur on this instruction's address, the trigger will not be found by the logic analyzer.

For example:

Bus Activity			High Level		
Address	Data	Mnemonic	Line	C-Source	
00000080	39600000	li r11,0	#13	i = 0;	
00000084	39400001	li r10,1	#14	j = 1;	
00000088	7D4A5A14	add r10,r10,r11	#15	k = j+i;	

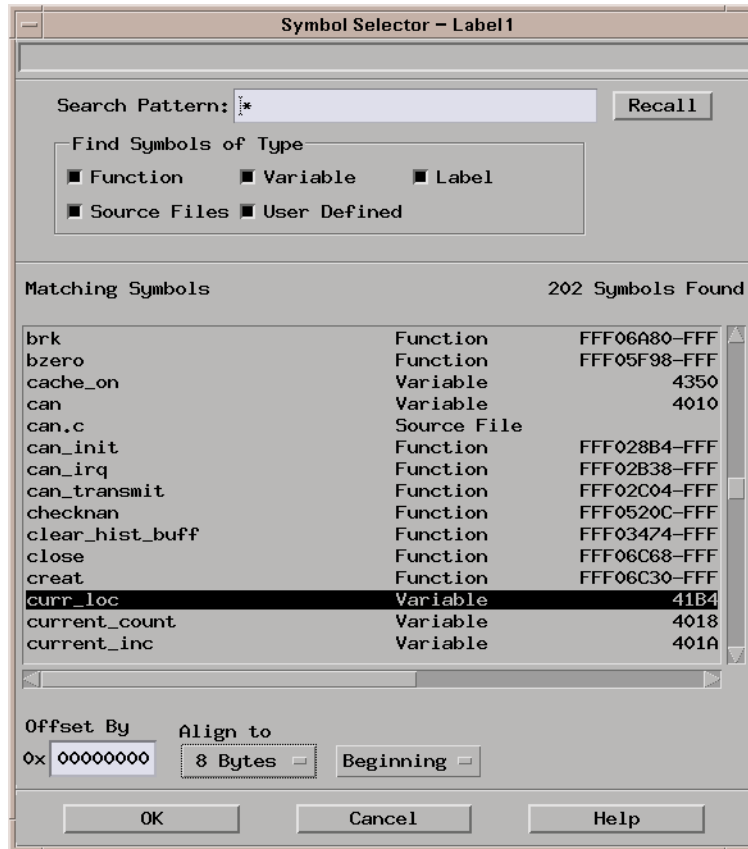
In the above example, instruction fetches will occur at addresses 80 and 88; a trigger set on line #14 (address 84) will not be detected. The instruction at address 84 was actually fetched with the 64-bit memory fetch at address 80, so you needed to trigger on address 80 to catch the fetch of address 84.

To help avoid these missed triggers, the trigger dialogs for symbol addresses allow you to "Align" the address to a 1-, 2-, 4-, or 8-byte boundary. Alignment

To Setup Trigger Alignment and Offset for Symbols and Source Code

affects the least significant address bits of the trigger specification, either setting them to a "don't care" or "zero" value, depending on the logic analyzer.

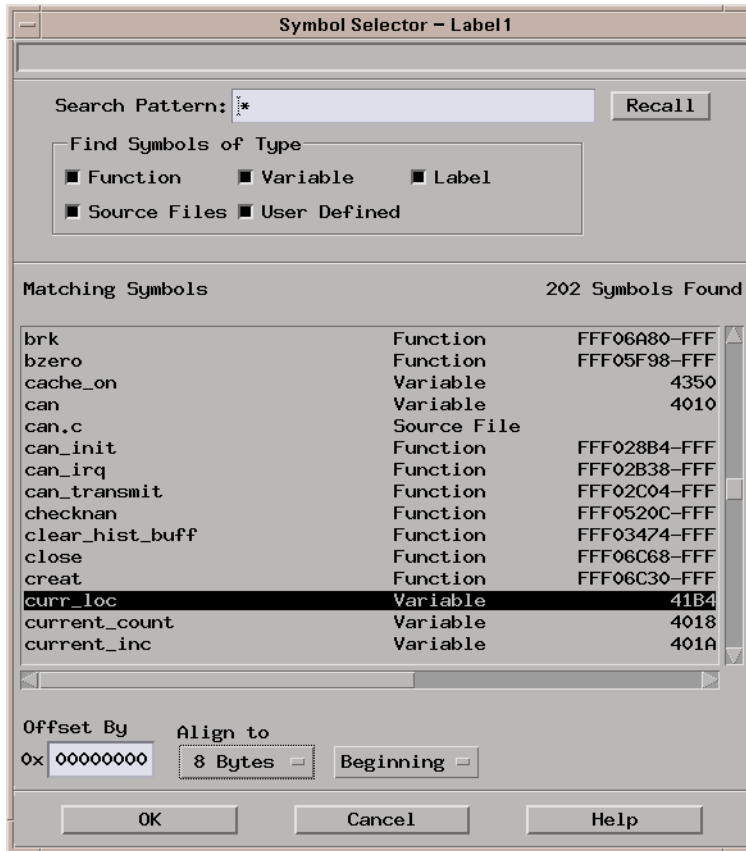
Set the alignment for program fetches to the width of the program memory in bytes using the **Align To** menu in the **Symbol Selector** dialog.



For the MPC7400 with 64-bit (8-byte) wide program memory, use 8-byte alignment. Eight-byte alignment will change the least significant three bits ($2^3 = 8$) of the trigger. Address 84 with 8-byte alignment results in a trigger address range of 80 through 87 for some logic analyzers (3 don't care bits), or an address of 80 on the other analyzers (three 0 bits). Note that either of these triggers would catch line #14 in the example above.

To compensate for relocated code

When code segments are relocated, or when memory management units produce fixed code offsets, you can compensate by using the **Offset by** field in the **Symbol Selector** dialog.



Entering the appropriate address offset will cause the source correlation tool set to reference the correct symbol information for the relocatable or offset code.

To adjust for prefetches, use a trigger offset of 0x8 (prefetch queue depth) to avoid triggering on prefetched instructions. Note that this is not a foolproof scheme, since this may result in a missed trigger if a branch takes place between the base address and the offset address. For the MPC7400, an offset

of 8 is large enough to overcome the prefetch queue.

Be aware of prefetches and adjust your triggering to compensate for them. Note that the MPC7400 has a good Branch Prediction Unit (PBU), and often, when executing loops, the processor will NOT fetch instructions beyond the end of the loop. This means that on most loops, an offset may not be required to avoid a false trigger.

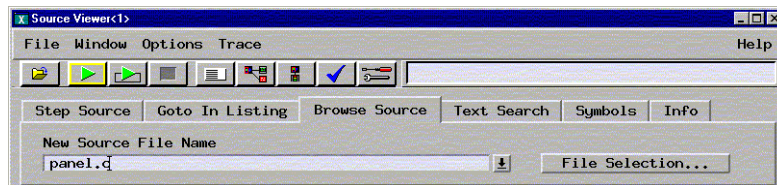
To Trigger on Source Code

The B4620B Source Correlation Tool Set lets you set triggers based upon source code.

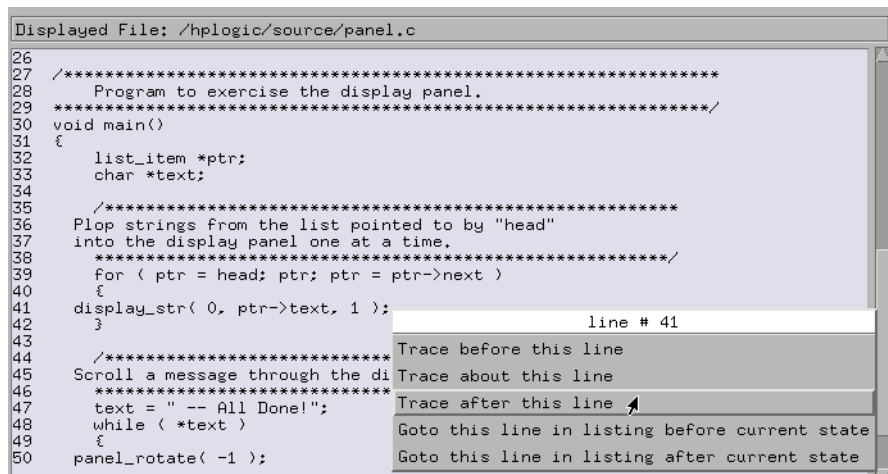
- 1 Open the **Source Viewer** window.



- 2 Browse the source file that contains the code you want to trigger on.



- 3 Click the source code line you want to trigger on and specify whether you want to trace before, about, or after the line. Or, use the Source Viewer's **Trace** menu to trace about a variable, function, or line number.



- 4 Run the measurement.



To avoid capturing library code execution

When viewing the source code associated with captured data, the Source Correlation Tool Set can exhibit long response times to requests for the next source line if the current trace listing corresponds to code from a library that is not in the source code search path. Logic analyzer storage qualification can be used to avoid capturing library code routines.

You should also configure the logic analyzer's storage qualification capabilities to store only those cycles that correspond to software execution (non-idle, etc.).

Chapter 6: Capturing Processor Execution
To Trigger on Source Code

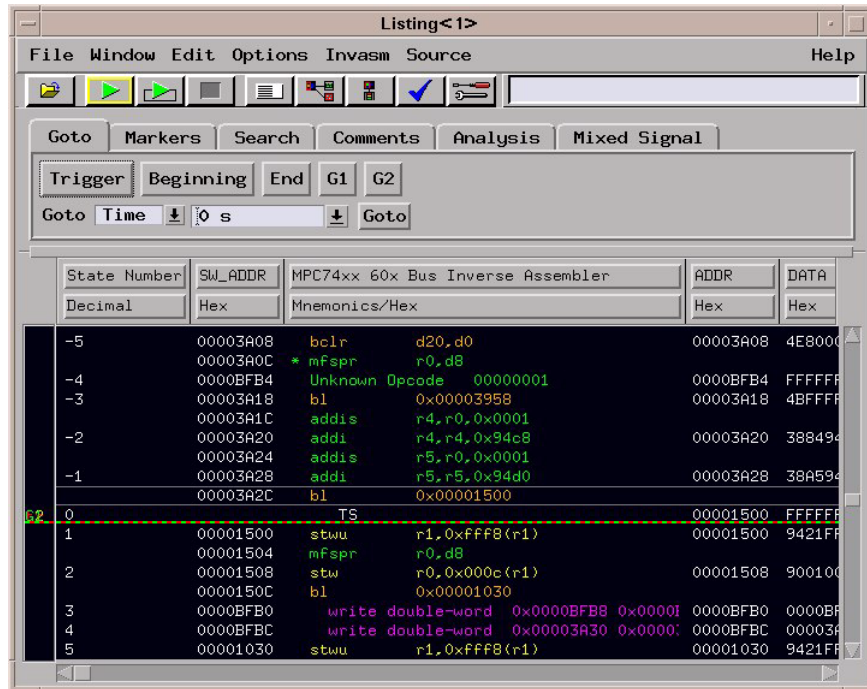
Displaying Captured MPC7400 Execution

To Display Captured State Data

- 1 Open the Listing display window.



The logic analyzer will display the captured state data in the Listing display.



The inverse assembler is loaded when state configuration files are loaded, but it can also be loaded into a Listing display using the Invasm menu. The name of the inverse assembler file is I74XXE, and it is located in the /logic/ia directory.

See Also “To Use the Inverse Assembler” on page 94.

See Also “To use the inverse assembler filters” on page 102 for information on displaying or hiding certain types of microprocessor bus cycles.

To Use the Inverse Assembler

This section discusses the general output format of the inverse assembler and processor-specific information.

To use the Invasm menu

The Invasm menu provides four choices: **Load**, **Preferences**, **Filter**, and **Options**. Access the Invasm menu in the listing window.

You must use the **Preferences** dialog to configure the inverse assembler to match the target system configuration. The other dialogs assist in analyzing and displaying data. The following sections describe these dialogs.

To load the inverse assembler

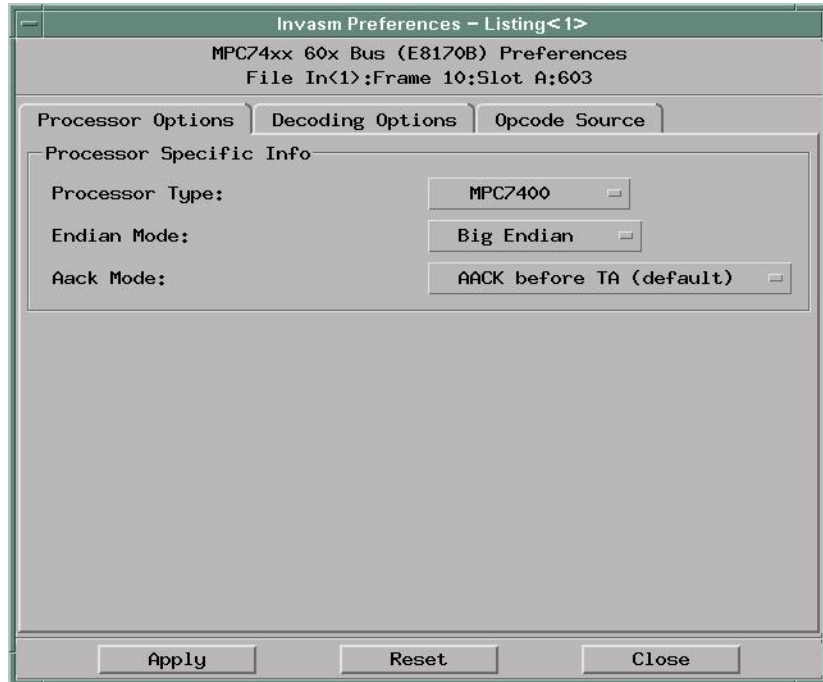
The Load dialog lets you load a different inverse assembler and apply it to the data in the Listing window. In some cases you may have acquired raw data; you can use the Load dialog to apply an inverse assembler to that data.

To set the inverse assembler preferences

The inverse assembler Preferences window contains three tabs: Processor Options, Decoding Options, and Opcode Source.

Processor Options

Inverse Assembler Processor Options



Processor Type

The Processor Type option is used to setup the inverse assembler for the MPC74XX processor you are using.

Endian Mode

The inverse assembler is designed to support both the native big endian mode and the little endian mode of operation. When operating in little-endian mode, the processor uses a technique known as “address munging” to convert internal little endian addresses into external big endian addresses. Internal and external addresses may differ from one another in the three least significant bits.

To Use the Inverse Assembler

Little endian mode causes the instruction word from DL0...31 (DATA_B label; external address xxx4) to be dispatched before the instruction word from DH0...31 (DATA label; external address xxx0). It also causes byte and half-word reads and writes to appear on the opposite side of the bus and swaps the halves of double-word reads and writes. Setting the endian mode to **Little Endian** automatically compensates for these little endian operations.

Aack Mode

The inverse assembler is designed to support both the default Address Acknowledge (AACK) timing and delayed AACK timing. In the default mode, the inverse assembler will search back from the Transfer Acknowledge (TA) state to find the corresponding AACK to retrieve the address and status data. In delayed AACK mode, the inverse assembler will search forward from the TA state to find the corresponding AACK. In delayed AACK mode, the AACK cannot be delayed past the next TA but may be concurrent with the next TA.

Decoding Options

Inverse Assembler Decoding Options Dialog

External Bus Decoding

Cache Off: External Bus Disassembly

Tracking Address:

◆ Data Bus Connected? Yes/No

Simplified Mnemonic Decoding

◆ Enable/Disable Simplified Instruction Mnemonics

Branch Common Compare

Condition Rotate & Shift Special Purpose

Subtract Trap

Exception Decoding

Exception prefix:

External Bus Decoding

Choose **Cache Off: External Bus Disassembly** for traditional inverse assembly or **Cache On: Branch Exception Disassembly** for cache-on trace reconstruction, and provide the tracking address.

Data Bus Connected

Read and write states are always indicated regardless of whether the data bus is connected. However, when the data bus is connected, read/write data will also be displayed. See “Inverse Assembler Modes of Operation” on page 109.

Simplified Mnemonic Decoding

PowerPC assemblers support a number of simplified mnemonics for some popular assembly language instructions, as described in Appendix F of Motorola’s publication *PowerPC Microprocessor Family: The Programming Environments for 32-Bit Microprocessors*. The inverse assembler will show those extensions if you wish to see them. By enabling the Simplified Mnemonic Decoding, you can select which types of simplified mnemonics will be shown. Click the options for the simplified mnemonics you desire.

- Conditional traps and branches decode the condition mnemonically when possible. For some conditions which have no conventional mnemonics (for example, “signed less than or unsigned greater than”), the condition field is displayed in binary.
- The L bit is omitted as a compare operand. Instead, compares are decoded as “cmpw” (or “?cmpd”).
- “Add immediate” instructions with a negative immediate operand are decoded as subtract immediate (“subi”).
- “Subtract from” instructions subf and subfc are decoded as subtract instructions sub and subc with the operands exchanged so that “sub r3 r4 r5” is mnemonically interpreted as “r3 = r4 - r5.”
- ori r0 r0 0000 is decoded as “nop”.
- Add immediate and add immediate shifted instructions, addi and addis, with a null source register are decoded as load immediate and load immediate shifted, li and lis.
- or instructions with identical source registers are decoded as move register, mr.
- nor instructions with identical source registers are decoded as not register, not.

To Use the Inverse Assembler

- xor and eqv instructions with identical source and destination registers are decoded as clear and set, clr and set, respectively.
- The cror, crnor, crxor, and creqv instructions map analogously to crmv, crnot, crclr, and crset.
- When the mtrcf instruction field mask specifies the entire cr, it is decoded as mtrc.

The Extended dialect adds several extended opcodes for the rotate instructions. For example, the function of the rlwinm instruction

```
rlwinm r30 r30 16. 16. 31.
```

is to shift right word immediate, e.g.

```
srwi r30 r30 16.
```

The PowerPC rotate-left instructions have extended mnemonics. The following listing shows the extended mnemonics for the integer rotate instructions.

Mnemonic	Decoded As
rlwimi (rotate left word immediate then mask insert)	inlwi insert from left immediate insrwi insert from right immediate
rlwinm (rotate left word immediate then AND with mask)	rotlwirotate left immediate rotrwirotate right immediate slwishift left immediate srwishift right immediate extlwiextract and left justify immediate extrwiextract and right justify immediate clrlwiclear left immediate clrrwiclear right immediate clrlslwiclear left and shift left immediate
rlwnm (rotate left word then AND with mask)	rotlwrotate left

The inverse assembler supports the following extensions of dialect-sensitive instructions.

Instruction Types	raw	extended
branches	bc %00100,2,FFF00230	bne cr0,FFF00230
trap	tw %10000,r5,r6	tw lt,r5,r6
compare	cmp cr1,0,r0,r16	cmpw cr1,r0,r16
	ori r0,r0,0000	nop
subtract	addi r6,r6,FCFC	subi r6,r6,0304
	subf r7,r19,r16	sub r7,r16,r19
common	addi r3,0,7000	li r3,7000
	addis r3,0,7000	lis r3,7000
	or r4,r5,r5	mr r4,r5
	nor r4,r5,r5	not r4,r5
	xor r7,r7,r7	clr r7
	eqv r8,r8,r8	set r8
special purpose	mtrcf %11111111,r5	mtrcr r5
condition	creqv 7,7,7	crset 7
	crxor 8,8,8	crclr 8
	cror 7,8,8	crmv 7,8
	crnor 8,9,9	crnot 8,9
rotates and shifts	rlwnm r8,r7,r6,0,31.	rotlw r8,r7,r6
	rlwimi r3,r3,24.,8,23.	inslwi r3,r3,16.,8
	rlwimi r8,r3,17,8,23.	insrwi r8,r3,7,8
	rlwinm r6,r4,8,0,14	extlwi r6,r4,15,8
	rlwinm r6,r4,16,24,31	extrwi r6,r4,8,8
	rlwinm. r6,r4,4,0,31	rotlwi. r6,r4,4
	rlwinm r6,r4,28,0,31	rotrwi r6,r4,4
	rlwinm r6,r4,1,0,30	slwi r6,r4,1
	rlwinm r6,r4,31,1,31	srwi r6,r4,1
	rlwinm r6,r4,0,1,31	clrlwi r6,r4,1
	rlwinm r6,r4,0,0,7	clrrwi r6,r4,14
	rlwinm r6,r4,6,6,25	clrlslwi r6,r4,12,6

To Use the Inverse Assembler**Exception Decoding**

The inverse assembler can output the types of exceptions that occur. The PowerPC architecture allows for two locations of the exception vector table. You can determine which location is set up for your target by looking at the MSR.IP bit 25. This can be done by examining the initialization code.

Listing Window Showing Trace with Data Bus Connected: Cache Off

The screenshot shows the MPC74xx Inverse Assembler interface. At the top, there are tabs for Search, Goto, Markers, Comments, Analysis, and Mixed Signal. Below these are search filters: Label (ADDR), Value (9d90), when (Present), and buttons for Next, Prev, Advanced searching..., Set G1, and Set G2.

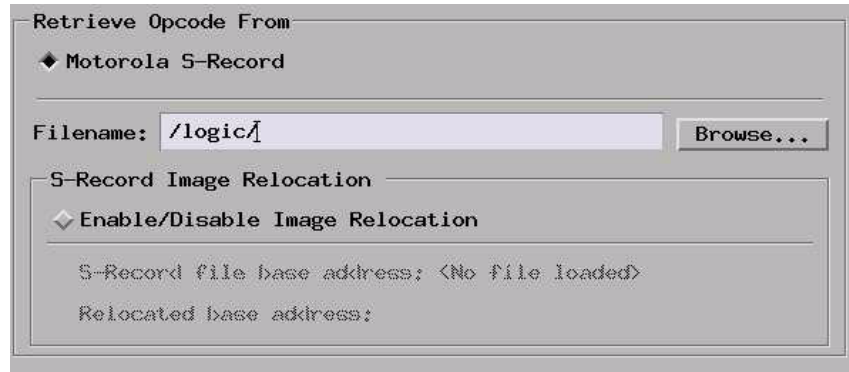
The main window displays a table of instructions with the following columns: State Number, SW_ADDR, and Mnemonics/Hex. The table contains the following data:

State Number	SW_ADDR	Mnemonics/Hex
280	ABSOLUTE 00009D90	read word 0x00000001
281	ABSOLUTE 00003B78	? addi r3,r0,0x0001
	ABSOLUTE 00003B7C	? stw r3,0x88d8(r13)
282	ABSOLUTE 00003B80	? stw r3,0x88d8(r13)
	ABSOLUTE 00003B84	lwz r12,0x88d8(r13)
283	ABSOLUTE 00003B88	addis r11,r0,0x41c6
	ABSOLUTE 00003B8C	ori r11,r11,0x4e6d
284	ABSOLUTE 00009D8C	read word 0x237c228a
285	ABSOLUTE 00003B90	mullw r12,r12,r11
	ABSOLUTE 00003B94	addi r3,r12,0x3039
286	ABSOLUTE 00003B98	stw r3,0x88d8(r13)
	ABSOLUTE 00003B9C	rlwinm r3,r3,d16,d17,d31
287	ABSOLUTE 00003BA0	bclr d20,d0
	ABSOLUTE 00003BA4	* stw r3,0x88d8(r13)
288	ABSOLUTE 00009D8C	write word 0xaf1cf0fb
289	ABSOLUTE 00001298	addi r10,r0,0x0019
	ABSOLUTE 0000129C	divw r0,r3,r10

Read and write data are displayed because the data bus is connected.

Opcode Source

Inverse Assembler Preferences Opcode Source Dialog



Specifying use of Motorola S-record executable file

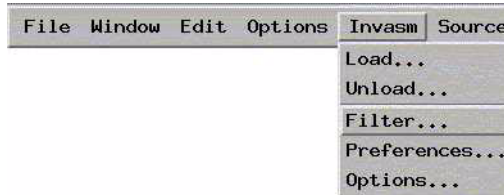
Select **Motorola S-Record** in the **Retrieve Opcode From** dialog to have a Motorola S-Record supply execution trace information to the cache-on trace reconstruction tool. Use the **Browse...** button to locate the S-record file.

S-Record Image Relocation

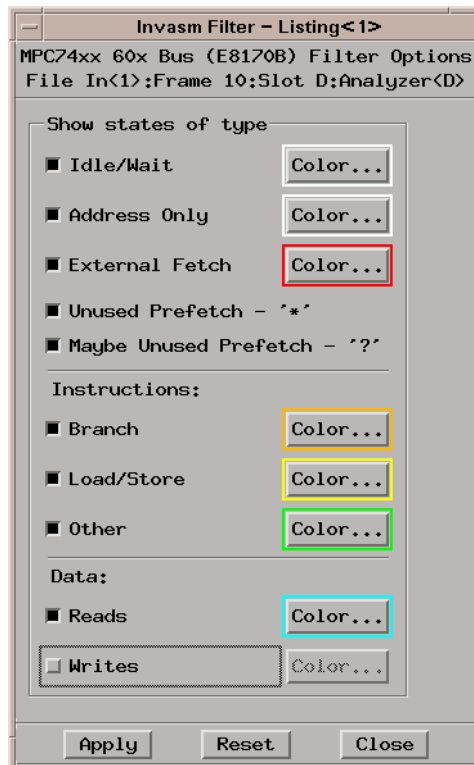
The Image Relocation portion of the dialog box allows you to relocate the SREC file to some other location in memory. This is useful when the loaded file is moved to some other location in memory. For example, the starting address in the SREC file is 1000. However, memory starting at 1000 is relocated to 5000. In order for the inverse assembler to retrieve the correct data, the entire SREC file must be relocated to 5000. Enter the relocated base address; all the resulting offsets will be calculated by the inverse assembler.

To use the inverse assembler filters

- In the Listing display window, choose the **Filter...** command from the Invasm menu.



Filter Window



The inverse assembler filtering options allow you to display or hide certain types of microprocessor bus cycles. Because the filter options do not affect the data that is stored by the logic analyzer (they only affect whether that data is displayed), they let you display the same data in different ways.

Filtering allows faster analysis in two ways:

- Unneeded information can be taken out of the display. For example, suppressing idle/wait states will let you view more instruction cycles in each screen.
- Particular operations can be isolated by suppressing all other operations. For example, Branch instructions can be shown, with all other states suppressed, allowing quick analysis of branch instructions.

You can also use color to distinguish between cycle types (when they are displayed).

To Interpret the Inverse Assembler Output

Data Formats

General purpose registers are displayed as r0, r1, r2...r31. Special purpose registers are displayed using their mnemonic.

Most numerical data is displayed in hexadecimal, for example, “`stwu r1,0xffff(r1)`.” Bit numbers and shift counts are displayed in decimal with a dot suffix, for example, “`cror 31. 31. 31.`”

A few instructions display their operands in binary with a “%” prefix, for example, “`mtfsfi 4 %0101.`”

The inverse assembler decodes the full PowerPC instruction set architecture, including 64-bit mode instructions and AltiVec instructions. When unimplemented opcodes are encountered, the listing displays “illegal opcode.”

An instruction word of 00000000 is decoded as “illegal opcode.” Otherwise, if an opcode is invalid, it is shown as “unknown opcode.”

Branch Instructions

If the address of a branch relative instruction is known, its target is presented as an absolute hex address (or as a symbol if it matches an ADDR pattern or range symbol). If the address of a branch relative instruction is not known, its target is displayed as a hexadecimal offset such as +00000C30 or -00000048.

Overfetch Marking

Overfetch refers to instructions which are fetched but not executed by the processor. They may arise from the following sources:

- When the 7400 executes a branch instruction, the instructions between the branch and the branch target are not executed. These instructions are indicated with an asterisk “*”, or if the bus trace is ambiguous, with a question mark “?”. If the instruction cache is enabled, the branch target may already be in the cache and will not be fetched over the bus. The remaining cache line containing the branch will be marked as overfetch.

For conditional branches whose target addresses are not known, or are known but not seen in the bus traffic, the inverse assembler cannot always determine if the branch was taken and will not mark ensuing states as overfetch.

To Use Cache-On Trace Reconstruction

Traditional inverse assembly, in which the external processor bus states are captured and decoded, may be implemented by disabling the target's cache. However, this will slow the target significantly, and may induce timing related problems. The target system's performance will be much better if the cache-on trace reconstruction feature is enabled when using the inverse assembler.

The cache-on trace reconstruction feature of the inverse assembler utilizes the branch trace mode. In order to trace in the cache the user must set the MSR.BE bit 22. This BE bit enables a branch trace exception to be taken after a successful completion of a branch instruction. This feature also requires that the data bus is connected and an S-Record executable file is loaded.

The branch exception is located at 0x00000D00 for an exception prefix MSR.IP=0 or 0xFFFF0D00 for an exception prefix MSR.IP=1. The interrupt routine writes the branch target address SRR0 to the tracking address (location in RAM which is non-cached or write-through mode is enabled for that memory block) so that the IA can track the program flow. Also, the tracking address must be on a word boundary.

Example branch exception routine:

```
0x00000d00:  mfspr    r7, d26
0x00000d04:  addis   r8, r0, 0x0000
0x00000d08:  stw     r7, 0x0100(r8)
0x00000d0C:  rfi
```

This branch exception writes the branch target address to a tracking address of 0x00000100.

If you want to nest interrupts you must save and restore the SRR0 special purpose register before writing it out to the tracking address. Also, you must write out the exception address at the beginning of the exception.

To Use Cache-On Trace Reconstruction

Example program exception routine:

```
0x00000700: addis r6, r0, 0x0000
0x00000704: addi  r6, 0x0700
0x00000708: addis r8, r0, 0x0000
0x0000070C: stw   r6, 0x0100(r8)
0x00000710: .
0x00000714: .
0x00000718: .
0x0000071C: mfspr r7, d26
0x00000720: stw   r7, 0x0100(r8)
0x00000724: rfi
```

To enable cache-on trace reconstruction:

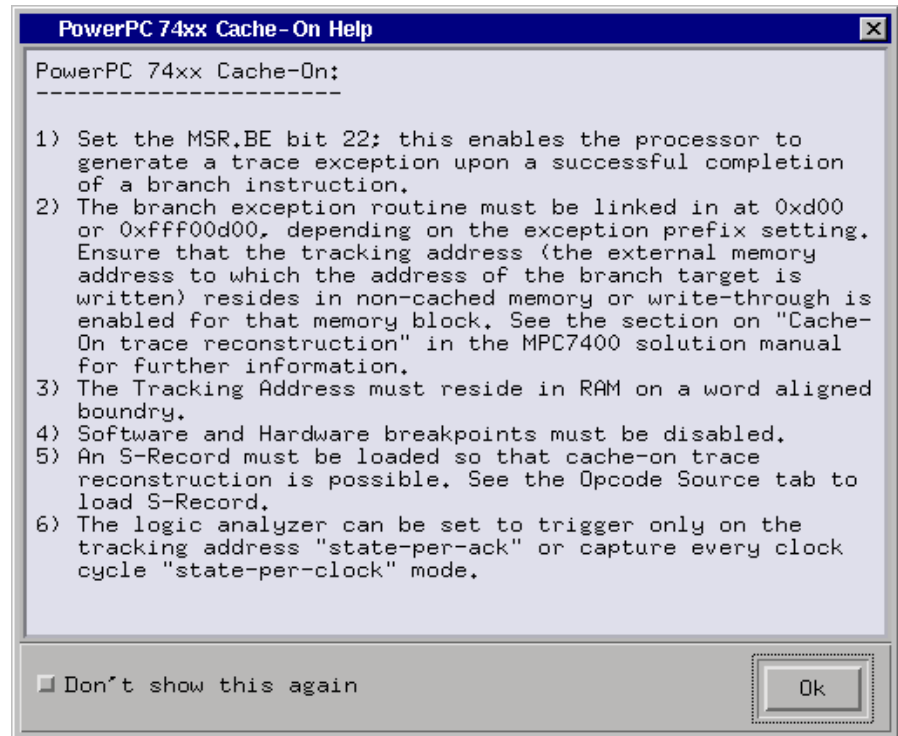
- 1** In the Decoding Options tab External Bus Decoding dialog:
 - a** Set the cache-on mode
 - b** Set data bus connected
 - c** Provide the tracking address
- 2** In the Opcode Source tab:
 - a** Load an S-Record executable file

Minimizing effects of cache-on trace on system performance

- Enable cache-on trace via the MSR.BE bit only for selected portions of code or specific tasks.
- Minimize the number of instructions in the exception handler.
 - Dedicate registers for writing out branch messages.
 - Do not save/restore any system registers.
- Make sure the instruction handler is in the instruction cache.
- Perform compiler optimization for the least number of branches.

When cache-on mode is enabled the following dialog will appear.

Cache-on help dialog



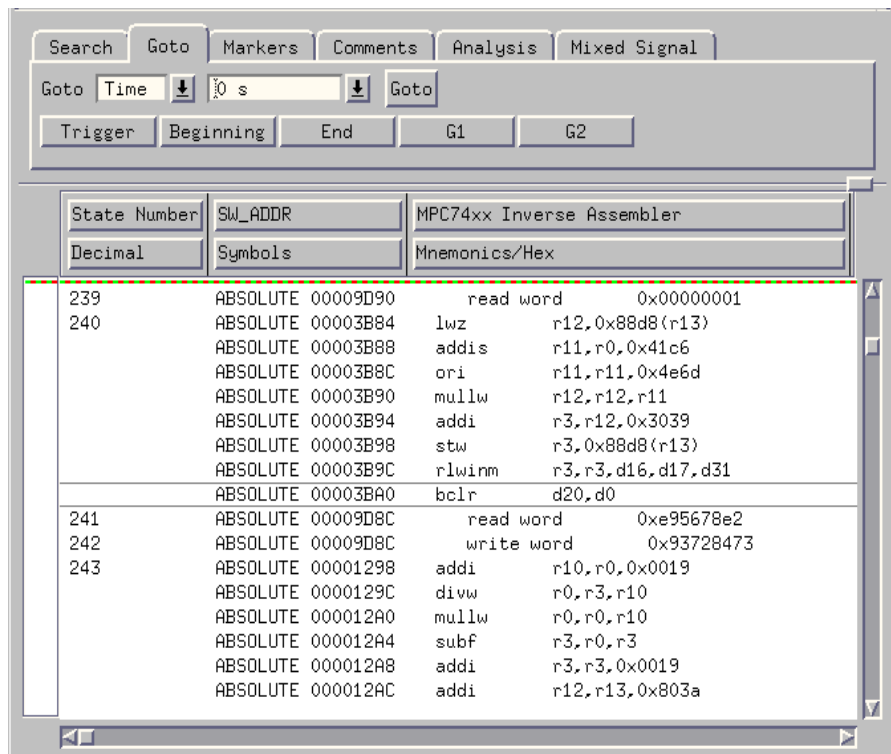
The Don't show this again button can be selected to prevent this dialog from appearing until the inverse assembler is loaded again.

To enable branch exception disassembly

The following trace shows cache-on execution using branch trace exception disassembly. See page 105 for an explanation of this feature.

To enable branch trace exception, set the MSR.BE bit 22.

Cache-on trace, S-Record executable file loaded, data bus connected



The screenshot shows the MPC7400 Inverse Assembler interface. At the top, there are tabs for Search, Goto, Markers, Comments, Analysis, and Mixed Signal. Below these is a Goto field with a Time dropdown set to 10 s and a Goto button. There are also buttons for Trigger, Beginning, End, G1, and G2. The main display area is a table with columns for State Number, SW_ADDR, and Mnemonics/Hex. The table contains the following data:

State Number	SW_ADDR	Mnemonics/Hex
239	ABSOLUTE 00009D90	read word 0x00000001
240	ABSOLUTE 00003B84	lwx r12,0x88d8(r13)
	ABSOLUTE 00003B88	addis r11,r0,0x41c6
	ABSOLUTE 00003B8C	ori r11,r11,0x4e6d
	ABSOLUTE 00003B90	mullw r12,r12,r11
	ABSOLUTE 00003B94	addi r3,r12,0x3039
	ABSOLUTE 00003B98	stw r3,0x88d8(r13)
	ABSOLUTE 00003B9C	rlwinm r3,r3,d16,d17,d31
	ABSOLUTE 00003BA0	bclr d20,d0
241	ABSOLUTE 00009D8C	read word 0xe95678e2
242	ABSOLUTE 00009D8C	write word 0x93728473
243	ABSOLUTE 00001298	addi r10,r0,0x0019
	ABSOLUTE 0000129C	divw r0,r3,r10
	ABSOLUTE 000012A0	mullw r0,r0,r10
	ABSOLUTE 000012A4	subf r3,r0,r3
	ABSOLUTE 000012A8	addi r3,r3,0x0019
	ABSOLUTE 000012AC	addi r12,r13,0x803a

Inverse Assembler Modes of Operation

The following table describes the various modes in which the inverse assembler can operate. An explanation of how to set up the inverse assembler to operate in these modes follows.

Inverse Assembler Modes of Operation

IA Cache Decoding	Data Bus Connected	S-Record Loaded	Result
off	no	no	Error message: opcode retrieval requires that the data bus is connected or an S-Record executable file is loaded.
off	no	yes	Opcodes are fetched from the S-Record executable file and decoded into instruction mnemonics. R/W data will not be displayed.
off	yes	no	Traditional Inverse Assembly: Opcodes are fetched from the data bus and decoded into instruction mnemonics. R/W data will be displayed.
off	yes	yes	Opcodes are fetched from the S-Record executable file and decoded into instruction mnemonics. R/W data will be displayed.
on	no	no	Error message: cache-on decoding requires that the data bus is connected and that an S-Record executable file is loaded.
on	no	yes	Error message: cache-on decoding requires that the data bus is connected and that an S-Record executable file is loaded.
on	yes	no	Error message: cache-on decoding requires that the data bus is connected and that an S-Record executable file is loaded.
on	yes	yes	Cache-on Trace Reconstruction: Tracking address data provides the address so opcodes can be fetched from the S-Record executable file and decoded into instruction mnemonics. R/W data will be displayed.

NOTE:

Read and write states are always indicated regardless of whether the data bus is connected. When the data bus is connected, read/write data will also be displayed.

To enable/disable the instruction cache on the MPC7400

When the instruction cache is enabled, many PowerPC instructions are executed from the cache and do not appear on the external bus. To get an execution trace on the bus, the instruction cache can be disabled. This must be done in supervisor mode.

To disable the cache with the emulation module:

Use your debugger or the Emulation Control Interface to configure the HID0 register.

Register values for controlling the cache

Value	Meaning
0000 8000	Enable Instruction Cache
0000 4000	Enable Data Cache
0000 0800	Invalidate Instruction Cache
0000 0400	Invalidate Data Cache

To disable the cache with code:

- Disable the instruction cache with the following code:

```
mf spr    r3, hid0
rlwinm   r3, r3, 0, 17, 15 # clear bit 16 (ICE)
mt spr   hid0, r3
isync
```

- To also disable the data cache use:

```
mf spr    r3, hid0
rlwinm   r3, r3, 0, 18, 15 # clear ICE and DCE
mt spr   hid0, r3
isync
```

- To invalidate and disable both caches use:

```
mf spr    r3, hid0
ori      r3, 0C00          # set ICFI and DCFI
mt spr   hid0, r3
rlwinm   r3, r3, 0, 22, 19 # clear ICFI and DCFI
```

```
mtspr    hid0, r3
rlwinm   r3, r3, 0, 18, 15    # clear ICE and DCE
mtspr    hid0, r3
isync
```

- Enable the instruction cache with the following code:

```
mfspir   r3, hid0
rlwinm   r3, r3, 1, 17, 15    # set ICE
mtspr    hid0, r3
isync
```

To View the Source Code Associated With Captured Data

The B4620B Source Correlation Tool Set lets you view the high-level source code associated with captured data and set up triggers based on source code.

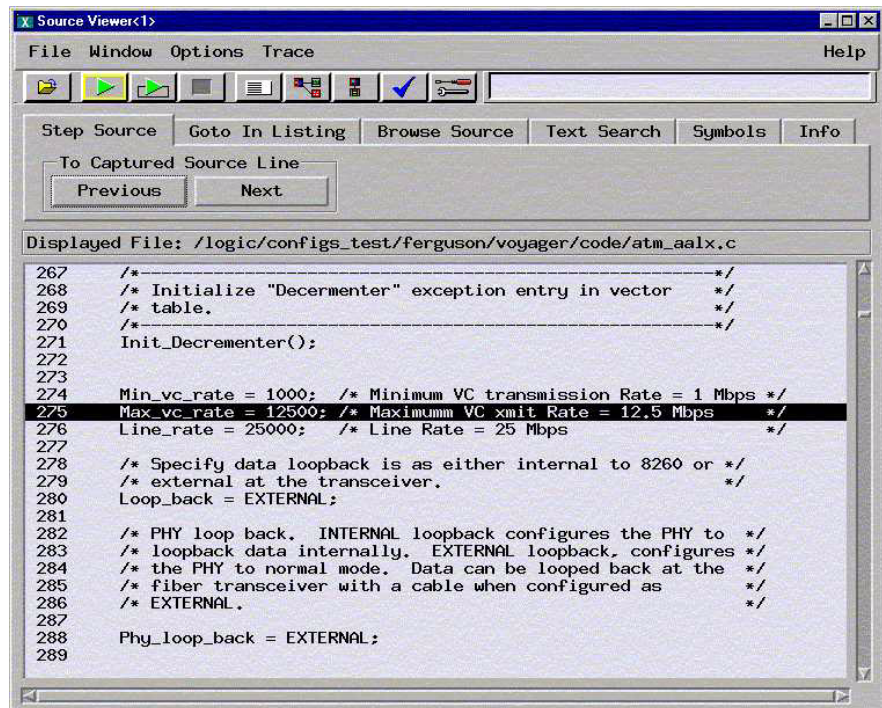
The source correlation tool set correlates the logic analyzer's address label with a line of high-level source code whose address, symbol name, file name, and line numbers are described in a symbol file downloaded to the logic analyzer (see "To load object file symbols" on page 74).

- In the Listing display window, select Source Viewer from the Source menu.



- Or, open the Source Viewer window from the logic analyzer's icon in the main system window.





Inverse Assembler Generated SW_ADDR (Software Address) Label

In the 16700-series logic analysis system, the MPC7400 inverse assembler generates a "SW_ADDR" label. The SW_ADDR label is displayed as another column in the Listing tool. This label is also known as the software address generated by the inverse assembler.

The **Goto In Listing** commands in the 16700-series logic analysis system perform a pattern search on the SW_ADDR label in the Listing display (when an inverse assembler is loaded). Because the inverse assembler is called for each line that is searched, the search can be slow, especially with a deep memory logic analyzer.

Also, a single line of source code will generate many assembly instructions. The **Goto In Listing** commands will not find a given line of source code unless the first assembly instruction generated from the source line has been acquired by the logic analyzer.

For example, if the compiler unrolls a loop in the source code, the trace could

Chapter 7: Displaying Captured MPC7400 Execution

To View the Source Code Associated With Captured Data

begin after the first assembly instruction of the loop has been executed. A **Goto In Listing** command would not find the source line.

Access to Source Code Files

The source correlation tool set must be able to access the high-level source code files referenced by the symbol information so that these source files can be displayed next to and correlated with the logic analyzer's execution trace acquisition. This requires you to be aware of a number of issues.

Source File Search Path. Verify that the correct file search paths for the source code have been entered into the source correlation tool set. The B4620B source correlation tool set can often read and access the correct source code file from information contained in the symbol file if the source code files have not been moved since they were compiled.

Network Access to Source Files. If source code files are being referenced across a network, the logic analyzer networking must be compatible with the user's network environment. Agilent Technologies logic analyzers currently support Ethernet networks running a TCP/IP protocol and support ftp, telnet, NFS client/server and X-Window client/server applications. Some PC networks may require extensions to the normal LAN protocols to support the TCP/IP protocol and/or these networking applications. Users should contact their LAN system administrators to help set up the logic analyzer on their network.

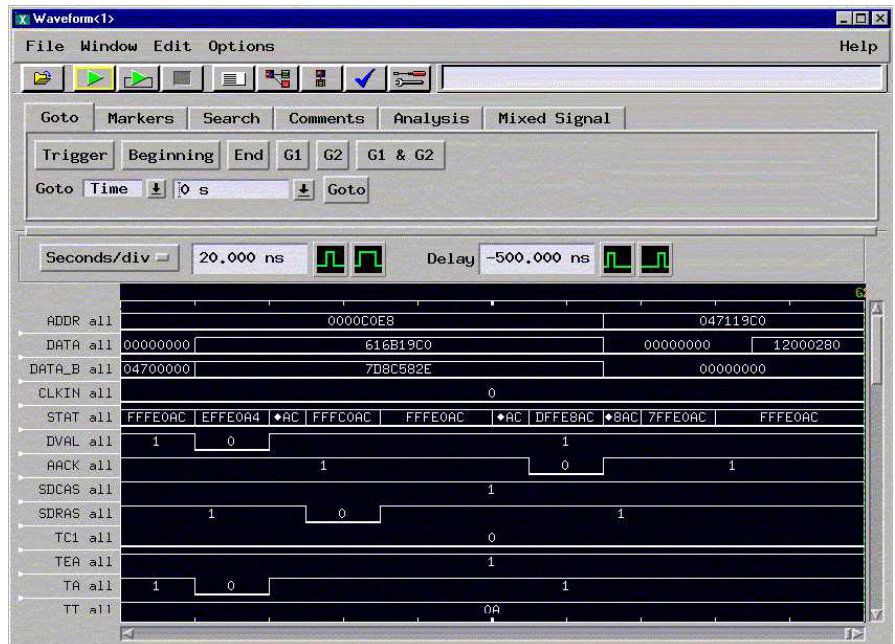
Source File Version Control. If the source code files are under a source code or version control utility, check the file names and paths carefully. These utilities can change source code file paths and file names. If these names are changed from the information contained in the symbol file, the source correlation tool set will not be able to find the proper source code file. These version control utilities usually provide an "export" command that creates a set of source code files with unmodified names. The source correlation tool set can then be given the correct path to these files.

See Also

More information on configuring and using the source correlation tool set can be found in the on-line help for your logic analysis system.

To Display Captured Timing Analysis Mode Data

- Open the Waveform display for your logic analyzer.



You can also use the Waveform display in the state analysis mode to display state timing diagrams

Chapter 7: Displaying Captured MPC7400 Execution
To Display Captured Timing Analysis Mode Data

Coordinating Logic Analysis with
Processor Execution

This chapter describes how to use a logic analyzer, an emulation module, and other features of your 1616700-series logic analysis system to gain insight into your target system.

What are some of the tools I can use?

You can use a combination of all of the following tools to control and measure the behavior of your target system:

- Your logic analyzer, to acquire data from the processor bus while it is running full-speed.
- Your emulation module, to control the execution of your target processor and to examine the state of the processor and of the target system.
- The Emulation Control Interface, to control and configure the emulation module, and to display or change target registers and memory.
- Display tools including the Listing tool, Chart tool, and System Performance Analyzer tool, to provide different views of the data collected using the logic analyzer.
- Your debugger, to control your target system using the emulation module. Do not use the debugger at the same time as the Emulation Control Interface.
- The B4620B source correlation tool set, to relate the analysis trace to your high-level source code.

Which assembly-level listing should I use?

Several windows display assembly language instructions. Be careful to use to the correct window for your purposes:

- The Listing tool shows processor states that were captured during a “Run” of the logic analyzer. Those states are disassembled and displayed in the Listing window.
- The Emulation Control Interface shows the disassembled contents of a section of memory in the Memory Disassembly window.
- Your debugger shows your program as it was actually assembled, and (if it supports the emulation module) shows which line of assembly code corresponds to the value of the program counter on your target system.

Which source-level listing should I use?

Different tools display source code for different uses:

- The Source Viewer window allows you to follow how the processor executed code as the analyzer captured a trace. You can use the Source Viewer to set analyzer triggers. The Source Viewer window is available only if you have licensed the B4620B source correlation tool set.
- Your debugger shows which line of code corresponds to the current value of the program counter on your target system. Use your debugger to set breakpoints.

Where can I find practical examples of measurements?

The Measurement Examples section in the on-line help contains quick reminders of how to perform common measurements.

A few of the many things outlined in the measurement examples are:

- How to find glitches.
- How to find NULL pointer de-references.
- How to profile system performance.

To find the measurement examples, select the Help icon in the logic analysis system window, then select “Measurement Examples.”

Stopping Processor Execution on a Logic Analyzer Trigger

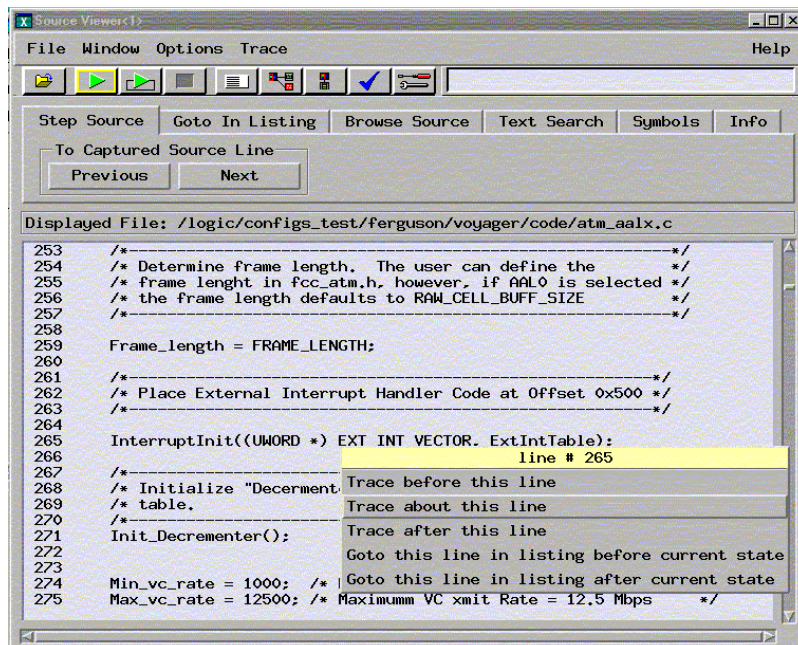
You can trigger the emulation module from the logic analyzer using either the Source Viewer window or the Intermodule window. If you are using the B4620B source correlation tool set, using the Source Viewer window is the easiest method.

To stop on a source line trigger (Source Viewer window)

If you have the B4620B source correlation tool set, you can easily stop the processor when a particular line of code is reached.

- 1 In the Source window, select the line of source code where you want to set the trigger, then select **Trace about this line**.

The logic analyzer trigger is now set.



2 Select **Trace->Enable - Break Emulator On Trigger**.

The emulation module is now set to halt the processor after receiving a trigger from the logic analyzer.

To disable the processor stop on trigger, select **Trace->Disable - Break Emulator On Trigger**.

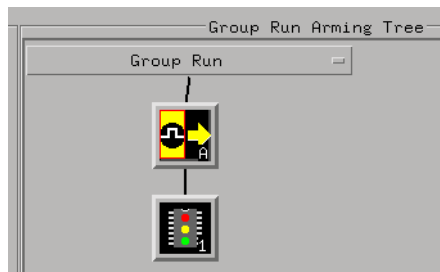
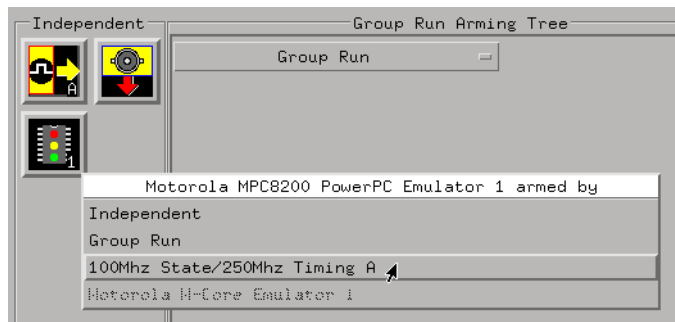
3 Select **Group Run** in the Source window (or other logic analyzer window).

4 If your target system is not already running, select **Group Run** in the emulation Run Control window to start your target.

To stop on any trigger (Intermodule window)

Use the Intermodule window if you do not have the B4620B source correlation tool set or if you need to use a more sophisticated trigger than is possible in the Source Viewer window.

- 1 Create a logic analyzer trigger.
- 2 In the Intermodule window, select the emulation module icon; then, select the analyzer which is intended to trigger it.



The emulation module is now set to stop the processor when the logic analyzer triggers.

- 3 Select **Group Run** in the Source window (or other logic analyzer window).

- 4 If your target system is not already running, select **Group Run** in the emulation Run Control window to start your target.

See Also

See the on-line help for your logic analysis system for more information on setting triggers.

To minimize the “skid” effect

There is a finite amount of time between when the logic analyzer triggers, and when the processor actually stops. During this time, the processor will continue to execute instructions. This latency is referred to as the skid effect.

To minimize the skid effect:

- 1 In the Emulation Control Interface, open the Configuration window.
- 2 Set processor clock speed to the maximum value which your target can support.

The amount of skid will depend on the processor’s execution speed and whether code is executing from the cache.

To stop the analyzer and view a measurement

- To view an analysis measurement you may have to select **Stop** after the trigger occurs.

When the target processor stops it may cause the analyzer qualified clock to stop. Therefore, most intermodule measurements will have to be stopped to see the measurement.

Chapter 8: Coordinating Logic Analysis with Processor Execution

Stopping Processor Execution on a Logic Analyzer Trigger

Example

An intermodule measurement has been set up where the analyzer is triggering the emulation module. The following sequence could occur:

1. The analyzer triggers.
2. The trigger (“Break In”) is sent to the emulation module.
3. The emulation module stops the user program which is running on the target processor. The processor enters a background debug monitor.
4. Because the processor has stopped, the analyzer stops receiving a qualified clock signal.
5. If the trigger position is “End”, the measurement will be completed.

If the trigger position is not “End”, the analyzer may continue waiting for more states.

6. The user selects **Stop** in a logic analyzer window, which tells the logic analyzer to stop waiting, and to display the trace.
-

Tracing Until the Processor Halts

If you are using a state analyzer, you can begin a trace, run the processor, then manually end the trace when the processor has halted.

To halt the processor, you can set a breakpoint using the Emulation Control Interface or a debugger.

Some possible uses for this measurement are:

- To store and display processor bus activity leading up to a system crash.
- To capture processor activity before a breakpoint.
- To determine why a function is being called. (You can set a breakpoint at the start of the function then use this measurement to see how the function is getting called.)

This kind of measurement is easier than setting up an intermodule measurement trigger.

To capture a trace before the processor halts

- 1** Set the logic analyzer to trigger on nostate.
- 2** Set the trigger point (position) to **End**.
- 3** In a logic analyzer window, select **Run**.
- 4** In the Emulation Control Interface or debugger select **Run**.
- 5** When the emulation module halts, select **Stop** in the logic analyzer window to complete the measurement.

This is the recommended method to do state analysis of the processor bus when the processor halts.

If you need to capture the interaction of another bus when the processor halts or you need to make a timing or oscilloscope measurement you will need to trigger the logic analyzer from the emulation module (described in the next section).

Triggering the Logic Analyzer when Processor Execution Stops

You can create an intermodule measurement which will allow the emulation module to trigger another module such as a timing analyzer or oscilloscope.

If you are only using a state analyzer to capture the processor bus, it will be much simpler to trace until a processor halts (see “Tracing Until the Processor Halts” on page 125).

Before you trigger a logic analyzer (or another module) from the emulation module, you should understand a few things about the emulation module trigger:

The Emulation Module Trigger Signal

The trigger signal coming from the emulation module is an “In Background Debug Monitor” (“In Monitor”) signal. This may cause confusion because a variety of conditions could cause this signal and falsely trigger your analyzer.

The “In Monitor” trigger signal can be caused by:

- The most common method to generate the signal is to select **Run** and then select **Break** in the Emulation Control Interface. Going from “Run” (Running User Program) to “Break” (“In Monitor”) generates the trigger signal.
- Another method to generate the “In Monitor” signal is to select **Reset** and then select **Break**. Going from the reset state of the processor to the “In Monitor” state will generate the signal. Some processors that do not remain in reset will not generate an In Monitor signal in the reset to break transition.
- In addition, an “In Monitor” signal is generated any time a debugger or other user interface reads a register, reads memory, sets breakpoints or steps. Care must be taken to not falsely trigger the logic analyzers listening to the “In Monitor” signal.

Group Run

The intermodule bus signals can still be active even without a Group Run.

The following setups can operate independently of Group Run:

- Port In connected to an emulation module.
- Emulation modules connected in series.
- Emulation module connected to Port Out.

Here are some examples:

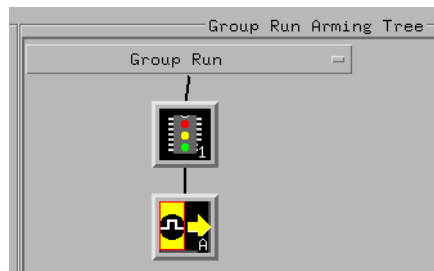
- If “Group Run” is armed from “Port In” and an emulation module is connected to Group Run, any “Port In” signal will cause the emulation module to go into monitor. The **Group Run** button does not have to be selected for this to operate.
- If two emulation modules are connected together so that one triggers another, the first one going into monitor will cause the second one to go into monitor.
- If an emulation module is connected to Port Out, the state of the emulation module will be sent out the Port Out without regard to “Group Run”.

The current emulation module state (Running or In Monitor) should be monitored closely when they are part of a Group Run measurement so that valid measurements are obtained.

Group Run into an emulation module does not mean that the Group Run will Run the emulation module.

The emulation module **Run**, **Break**, **Step**, and **Reset** are independent of the **Group Run** of the Analyzers.

For example, suppose you have the following intermodule measurement set up:



Chapter 8: Coordinating Logic Analysis with Processor Execution

Triggering the Logic Analyzer when Processor Execution Stops

Selecting the **Group Run** button (at the very top of the Intermodule window or a logic analyzer window) will start the analyzer running. The analyzer will then wait for an arm signal. Now, when the emulation module transitions into “Monitor” from “Running” (or from “Reset”), it will send the arm signal to the analyzer. If the emulation module is “In Monitor” when you select **Group Run**, you will then have to go to the emulation module or your debugger interface and manually start it running.

Debuggers can cause triggers

Emulation module user interfaces may introduce additional states into your analysis measurement and in some cases falsely trigger your analysis measurement.

When a debugger causes your target to break into monitor it will typically read memory around the program stack and around the current program counter. This will generate additional states which appear in the listing.

You can often distinguish these additional states because the time tags will be in the microsecond and millisecond range. You can use the time tag information to determine when the processor went into monitor. Typically the time between states will be in the nanoseconds while the processor is running and will be in the microsecond and millisecond range when the debugger has halted the processor and is reading memory.

Note also that some debugger commands may cause the processor to break temporarily to read registers and memory. These states that the debugger introduces will also show up in your trace listing.

If you define a trigger on some state and the debugger happens to read the same state, then you may falsely trigger your analyzer measurement. In summary, when you are making an analysis measurement be aware that the debugger could be impacting your measurement.

To trigger the analyzer when the processor halts

Remember: if you are only using a state analyzer to capture the processor bus then it will be much simpler to trace until a processor halts (see “Tracing Until the Processor Halts” on page 125).

- 1 Set the logic analyzer to trigger on anystate.
- 2 Set the trigger point to **center** or **end**.
- 3 In the Intermodule window, select the logic analyzer you want to trigger and select the emulation module.

The logic analyzer is now set to trigger on a processor halt.

- 4 Select **Group Run** to start the analyzer(s).
- 5 Select **Run** in the Emulation Control Interface or use your debugger to start the target processor running.

NOTE:

Selecting **Group Run** will not start the emulation module. The emulation module run, break, step, reset are independent of the Group Run of the analyzers.

- 6 Wait for the Run Control window in the Emulation Control Interface or the status display in your debugger to show that the processor has stopped.

The logic analyzer will store states up until the processor stops, but may continue running.

You may or may not see a “slow clock” error message. In fact, if you are using a state analyzer on the processor bus, the status may never change upon receiving the emulation module trigger (analysis arm). This occurs because the qualified processor clock needed to switch the state analyzer to the next state is stopped. For example, the state analyzer before the arm event may have a status of “Occurrences Remaining in Level 1: 1” and after the arm event it may have the same status of “Occurrences Remaining in Level 1: 1”.

- 7 If necessary, in the logic analyzer window, select **Stop** to complete the measurement.

If you are using a timing analyzer or oscilloscope, the measurement should complete automatically when the processor halts. If you are using a state logic

analyzer, select **Stop** if needed to complete the measurement.

To trigger the analyzer when the processor reaches a breakpoint

This measurement is exactly like the one on the previous page, but with the one additional complexity of setting breakpoints. Be aware that setting breakpoints may cause a false trigger and that the breakpoints set may not be valid after a reset.

Remember: if you are only using a state analyzer to capture the processor bus, it will be much simpler to trace until a processor halts (see “Tracing Until the Processor Halts” on page 125).

- 1 Set the logic analyzer to trigger on anystate.
- 2 Set the trigger point to center or end.
- 3 In the Intermodule window, select the logic analyzer you want to trigger and select the emulation module.

The logic analyzer is now set to trigger on a processor halt.

- 4 Set the breakpoint.

If you are going to run the emulation module from Reset you must do a **Reset** followed by **Break** to properly set the breakpoints. The Reset will clear all on-chip hardware breakpoint registers. The Break command will then reinitialize the breakpoint registers. If you are using software breakpoints which insert an illegal instruction into your program at the breakpoint location you will not need to do the Reset, Break sequence. Instead, you must take care to properly insert your software breakpoint in your RAM program location.

- 5 Select **Group Run** to start the analyzer(s).
- 6 Select **Run** in the Emulation Control Interface or use your debugger to start the target processor running.

NOTE:

Selecting **Group Run** will not start the emulation module. The emulation module run, break, step, reset are independent of the Group Run of the analyzers.

- 7 Wait for the Run Control window in the Emulation Control Interface or the status display in your debugger to show that the processor has stopped.

The logic analyzer will store states up until the processor stops, but may continue running.

You may or may not see a “slow clock” error message. In fact, if you are using a state analyzer on the processor bus, the status may never change upon receiving the emulation module trigger (analysis arm). This occurs because the qualified processor clock needed to switch the state analyzer to the next state is stopped. For example, the state analyzer before the arm event may have a status of “Occurrences Remaining in Level 1: 1” and after the arm event it may have the same status of “Occurrences Remaining in Level 1: 1”.

- 8 If necessary, in the logic analyzer window, select Stop to complete the measurement.

If you are using a timing analyzer or oscilloscope the measurement should complete automatically when the processor halts. If you are using a state logic analyzer, select **Stop** if needed to complete the measurement.

Chapter 8: Coordinating Logic Analysis with Processor Execution
Triggering the Logic Analyzer when Processor Execution Stops

General-Purpose ASCII (GPA) Symbol
File Format

Chapter 9: General-Purpose ASCII (GPA) Symbol File Format

General-purpose ASCII (GPA) format files are loaded into a logic analyzer just like other object files, but they are usually created differently.

If your compiler does not include symbol information in the output, or if you want to define a symbol not in the object file, you can create an ASCII format symbol file.

Typically, ASCII format symbol files are created using text processing tools to convert compiler or linker map file output that has symbolic information into the proper format.

You can typically get symbol table information from a linker map file to create a General-Purpose ASCII (GPA) symbol file.

Various kinds of symbols are defined in different records in the GPA file. Record headers are enclosed in square brackets; for example, [VARIABLES]. For a summary of GPA file records and associated symbol definition syntax, refer to the "GPA Record Format Summary" that follows.

Each entry in the symbol file must consist of a symbol name followed by an address or address range.

While symbol names can be very long, the logic analyzer only uses the first 16 characters.

The address or address range corresponding to a given symbol appears as a hexadecimal number. The address or address range must immediately follow the symbol name, appear on the same line, and be separated from the symbol name by one or more blank spaces or tabs. Ensure that address ranges are in the following format:

```
beginning address..ending address
```

Example

```
main      00001000..00001009
test      00001010..0000101F
var1      00001E22      #this is a variable
```

This example defines two symbols that correspond to address ranges and one point symbol that corresponds to a single address.

For more detailed descriptions of GPA file records and associated symbol definition syntax, refer to these topics that follow:

- SECTIONS
- FUNCTIONS
- VARIABLES
- SOURCE LINES
- START ADDRESS
- Comments

GPA Record Format Summary

Format

```
[SECTIONS]
section_name start..end attribute

[FUNCTIONS]
func_name start..end

[VARIABLES]
var_name start [size]
var_name start..end

[SOURCE LINES]
File: file_name
line# address

[START ADDRESS]
address

#Comments
```

If no record header is specified, [VARIABLES] is assumed. Lines without a preceding header are assumed to be symbol definitions in one of the VARIABLES formats.

Example

This is an example GPA file that contains several different kinds of records:

```
[SECTIONS]
prog      00001000..0000101F
data      40002000..40009FFF
common    FFFF0000..FFFF1000

[FUNCTIONS]
main      00001000..00001009
test      00001010..0000101F

[VARIABLES]
total     40002000  4
value     40008000  4

[SOURCE LINES]
File: main.c
10        00001000
11        00001002
14        0000100A
22        0000101E

File: test.c
5         00001010
7         00001012
11        0000101A
```

SECTIONS

Format

```
[SECTIONS]
  section_name start..end attribute
```

Use SECTIONS to define symbols for regions of memory, such as sections, segments, or classes.

`section_name` A symbol representing the name of the section.

`start` The first address of the section, in hexadecimal.

`end` The last address of the section, in hexadecimal.

`attribute` This is optional, and may be one of the following:

NORMAL (default)—The section is a normal, relocatable section, such as code or data.

NONRELOC—The section contains variables or code that cannot be relocated; this is an absolute segment.

Enable Section Relocation

To enable section relocation, section definitions must appear before any other definitions in the file.

Example

```
[SECTIONS]
  prog          00001000..00001FFF
  data          00002000..00003FFF
  display_io    00008000..0000801F  NONRELOC
```

If you use section definitions in a GPA symbol file, any subsequent function or variable definitions must be within the address ranges of one of the defined sections. Functions and variables that are not within the range are ignored.

FUNCTIONS

Format [FUNCTIONS]
 func_name start..end

Use FUNCTIONS to define symbols for program functions, procedures, or subroutines.

func_name A symbol representing the function name.

start The first address of the function, in hexadecimal.

end The last address of the function, in hexadecimal.

Example [FUNCTIONS]
 main 00001000..00001009
 test 00001010..0000101F

VARIABLES

Format

```
[VARIABLES]
  var_name  start [size]
  var_name  start..end
```

You can specify symbols for variables either by using the address of the variable, the address and the size of the variable, or a range of addresses occupied by the variable. If you specify only the address of a variable, the size is assumed to be one byte.

var_name A symbol representing the variable name.

start The first address of the variable, in hexadecimal.

end The last address of the variable, in hexadecimal.

size This is optional, and indicates the size of the variable, in bytes, in decimal.

Example

```
[VARIABLES]
  subtotal  40002000  4
  total     40002004  4
  data_array 40003000..4000302F
  status_char 40002345
```

SOURCE LINES

Format [SOURCE LINES]
 File: file_name
 line# address

Use SOURCE LINES to associate addresses with lines in your source files.

`file_name` The name of a file.

`line#` The number of a line in the file, in decimal.

`address` The address of the source line, in hexadecimal.

Example [SOURCE LINES]
 File: main.c
 10 00001000
 11 00001002
 14 0000100A
 22 0000101E

START ADDRESS

Format [START ADDRESS]
 address

address The address of the program entry point, in hexadecimal.

Example [START ADDRESS]
 00001000

Comments

Format #comment text

Use the # character to include comments in a file. Any text following the # character is ignored. You can put comments on a line alone or on the same line following a symbol entry.

Example #This is a comment.

Troubleshooting the Inverse
Assembler

Chapter 10: Troubleshooting the Inverse Assembler

If you encounter difficulties while making measurements, use this chapter to guide you through some possible solutions. Each heading lists a problem you may encounter, along with some possible solutions.

If you still have difficulty using the analyzer after trying the suggestions in this chapter, please contact your local Agilent Technologies service center.

CAUTION:

When you are working with the analyzer, be sure to power down both the analyzer and the target system before disconnecting or connecting cables. Otherwise, you may damage circuitry in the analyzer or target system.

Logic Analyzer Problems

This section lists general problems that you might encounter while using the logic analyzer.

Intermittent data errors

This problem is usually caused by poor connections, incorrect signal levels, or marginal timing.

- ❑ Remove and re-seat all cables and probes, ensuring that there are no bent pins or poor probe connections.
- ❑ Adjust the threshold level of the data pod to match the logic levels in the system under test.
- ❑ Use an oscilloscope to check the signal integrity of the data lines.

Clock signals for the state analyzer must meet particular pulse shape and timing requirements. Data inputs for the analyzer must meet pulse shape and setup and hold time requirements.

See Also

See “Capacitive loading” on page 148 for information on other sources of intermittent data errors.

Unwanted triggers

Unwanted triggers can be caused by instructions that were fetched but not executed.

- ❑ Add the prefetch queue or pipeline depth to the trigger address to avoid this problem.

The logic analyzer captures prefetches, even if they are not executed. When you are specifying a trigger condition or a storage qualification that follows an instruction that may cause branching, an unused prefetch may generate an unwanted trigger.

No activity on activity indicators

- ❑ Check for loose cables or board connections.
 - ❑ Check for bent or damaged pins on the connectors.
-

No trace list display

If there is no trace list display, it may be that your trigger specification is not correct for the data you want to capture, or that the trace memory is only partially filled.

- ❑ Check your trigger sequencer specification to ensure that it will capture the events of interest.
 - ❑ Try stopping the analyzer; if the trace list is partially filled, this should display the contents of trace memory.
-

Analyzer won't power up

If logic analyzer power is cycled when the logic analyzer is connected to a target system or emulation probe that remains powered up, the logic analyzer may not be able to power up. Some logic analyzers are inhibited from powering up when they are connected to a target system or emulation probe that is already powered up.

- ❑ Remove power from the target system, then disconnect all logic analyzer cabling from the target system. This will allow the logic analyzer to power up. Reconnect logic analyzer cabling after power up.
-

Target System Problems

This section lists problems that you might encounter with the target system.

Target system will not boot up

If the target system will not boot up after connecting the logic analyzer, the microprocessor (if socketed) or the cables may not be installed properly, or they may not be making electrical contact.

- ❑ Ensure that you are following the correct power-on sequence for the target system, logic analyzer (and analysis probe if used).
 - a** Power up the analyzer.
 - b** Power up the target system.

- ❑ Verify that the microprocessor and the cables are securely inserted into their respective sockets.

- ❑ Verify that the logic analyzer cables are in the proper sockets of the target system and are firmly inserted.

Erratic trace measurements

- ❑ Do a full reset of the target system before beginning the measurement.

Some designs require a full reset to ensure correct configuration.

- ❑ Ensure that your target system meets the timing requirements of the processor with the logic analyzer probe connected.

See “Capacitive loading” in this chapter. While logic analyzer loading is slight, pin protectors, extenders, and adapters may increase it to unacceptable levels. If the target system design has close timing margins, such loading may cause incorrect processor functioning and give erratic trace results.

- ❑ Ensure that you have sufficient cooling for the microprocessor.

Ensure that you have ambient temperature conditions and air flow that meet or exceed the requirements of the microprocessor manufacturer.

Capacitive loading

Excessive capacitive loading can degrade signals, resulting in incorrect capture or system lockup in the microprocessor. All interfaces add additional capacitive loading, as can custom probe fixtures you design for your application.

Careful layout of your target system can minimize loading problems and result in better margins for your design. This is especially important for systems that are running at frequencies greater than 50 MHz.

- ❑ Remove as many pin protectors, extenders, and adapters as possible.

Inverse Assembler Problems

This section lists problems that you might encounter while using the inverse assembler.

When you obtain incorrect inverse assembly results, it may be unclear whether the problem is in the connectors or in your target system. If you follow the suggestions in this section to ensure that you are using inverse assembler correctly, you can proceed with confidence in debugging your target system.

No inverse assembly or incorrect inverse assembly

This problem may be due to incorrect synchronization, modified configuration, incorrect connections, or a hardware problem in the target system. A locked status line can cause incorrect or incomplete inverse assembly.

- ❑ Ensure that the correct processor is selected in the processor options preferences menu.

Refer to page 94 to select the correct processor type.

- ❑ Ensure that each logic analyzer pod is connected to the correct connector.

There is not always a one-to-one correspondence between analyzer pod numbers and connector numbers. Target systems must supply address (ADDR), data (DATA), and status (STAT) information to the analyzer in a predefined order. The cable connections are often altered to support that need. Thus, one target system might require that you connect cable 2 to analyzer pod 2, while another will require you to connect cable 5 to analyzer pod 2. See Chapter 3 for connection information.

- ❑ Check the activity indicators for status lines locked in a high or low state.
- ❑ Verify that the STAT, DATA, and ADDR format labels have not been modified from their default values.

These labels must remain as they are configured by the configuration file. Do

Inverse Assembler Problems

not change the names of these labels or the bit assignments within the labels. Some analysis probes also require other data labels. See “Configuring the Logic Analyzer” on page 63 for more information.

- Verify that memory managers have been disabled.

In most cases, if the memory managers remain enabled you should still get inverse assembly. It may be incorrect because the logical address may not map to the physical address.

- Verify that the preferences are set correctly if you have not disabled the cache.

For instructions on how to disable the cache, see page 110.

If the cache is enabled, follow the instructions in “To use the inverse assembler filters” on page 102.

If the cache is enabled, ensure that an S-Record is loaded and that the preferences are set correctly (see page 94.)

- Verify that storage qualification has not excluded storage of all the needed opcodes and operands.
- Verify that the endian selection is correct in the preferences menu (see page 94).

Inverse assembler will not load or run

The inverse assembler may not run if you do not have the correct system software, or if the inverse assembler is not in the same disk as the configuration files that you are using.

- ❑ Ensure that you have the correct system software loaded on your analyzer.
- ❑ Ensure that the inverse assembler is on the same disk as the configuration files you are loading.

Configuration files for the state analyzer contain a pointer to the name of the corresponding inverse assembler. If you delete the inverse assembler or rename it, the configuration process will fail to load the disassembler.

See Chapter 3, “Setting Up the Logic Analysis System,” beginning on page 43 for details.

Intermodule Measurement Problems

Some problems occur only when you are trying to make a measurement involving multiple modules.

An event wasn't captured by one of the modules

If you are trying to capture an event that occurs very shortly after the event that arms one of the measurement modules, it may be missed due to internal analyzer delays. For example, suppose you set an oscilloscope module to trigger upon receiving a trigger signal from the logic analyzer because you are trying to capture a pulse that occurs right after the analyzer's trigger state. If the pulse occurs too soon after the analyzer's trigger state, the oscilloscope will miss the pulse.

- ❑ Adjust the skew in the Intermodule menu.

You may be able to specify a skew value that enables the event to be captured.

- ❑ Change the trigger specification for modules upstream of the one with the problem.

If you are using a logic analyzer to trigger an oscilloscope module, try specifying a trigger state one state before the one you are using. This may be more difficult than working with the skew because the prior state may occur more often and not always be related to the event you are trying to capture with the oscilloscope.

Logic Analyzer Messages

This section lists some of the messages that the analyzer displays when it encounters a problem.

“. . . Inverse Assembler Not Found”

This error occurs if you rename or delete the inverse assembler file that is attached to the configuration file.

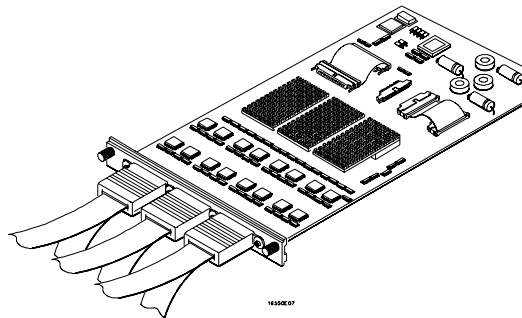
Ensure that the inverse assembler file is not renamed or deleted, and that it is located in /logic/ia.

See Chapter 3, “Setting Up the Logic Analysis System,” beginning on page 43 for details.

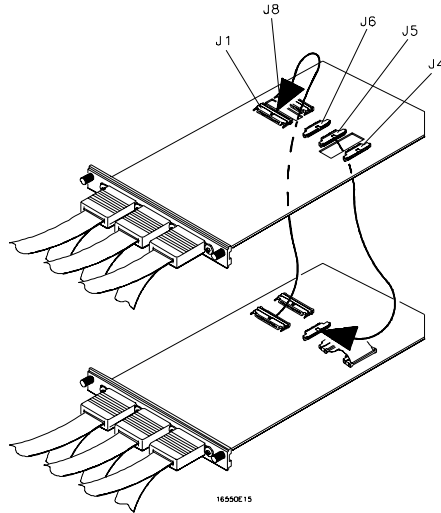
“Measurement Initialization Error”

This error occurs when you have installed the cables incorrectly for one or two 16550A logic analysis cards. The following diagrams show the correct cable connections for one-card and two-card installations. Ensure that your cable connections match the silk screening on the card, and that they are fully seated in the connectors. Then, repeat the measurement.

Cable Connections for One-Card 16550A Installations



Cable Connections for Two-Card 16550A Installations



See Also

The *HP/Agilent 16550A 100-MHz State/500-MHz Timing Logic Analyzer Service Guide*.

“No Configuration File Loaded”

This is usually caused by trying to load a configuration file for one type of module/system into a different type of module/system.

- ❑ Verify that the appropriate module has been selected from the Load {module} from File {file name} in the disk operation menu. Selecting Load {All} will cause incorrect operation when loading most analysis probe interface configuration files.

See Also

See “To load configuration files (and the inverse assembler) from the system hard disk” on page 66 for details on loading configuration files.

“Selected File is Incompatible”

This occurs when you try to load a configuration file for the wrong module. Ensure that you are loading the appropriate configuration file for your logic analyzer.

“Slow or Missing Clock”

- ❑ This error message might occur if the logic analyzer cards are not firmly seated in the logic analysis system frame. Ensure that the cards are firmly seated.
 - ❑ This error might occur if the target system is not running properly. Ensure that the target system is on and operating properly.
 - ❑ If the error message persists, check that the logic analyzer pods are connected to the proper connectors on the target system or analysis probe interface. See Chapter 3 to determine the proper connections.
-

“Time from Arm Greater Than 41.93 ms”

The 16550A state/timing analyzers have a counter to keep track of the time from when an analyzer is armed to when it triggers. The width and clock rate of this counter allow it to count for up to 41.93 ms before it overflows. Once the counter has overflowed, the system does not have the data it needs to calculate the time between module triggers. The system must know this time to be able to display data from multiple modules on a single screen.

“Waiting for Trigger”

If a trigger pattern is specified, this message indicates that the specified trigger pattern has not occurred. Verify that the triggering pattern is correctly set.

- ❑ When analyzing microprocessors that fetch only from word-aligned addresses, ensure that the trigger condition is set to look for an opcode fetch at an address corresponding to a word boundary.

Hardware Reference

This chapter contains additional reference information including the specifications and characteristics for the target system when using the inverse assembler software and signal mapping for E8170B software.

Operating characteristics

The following operating characteristics are not specifications, but are typical operating characteristics for the E8170B Inverse Assembler.

Operating Characteristics	
Microprocessor Compatibility	MPC7400
Maximum Supported Microprocessor Bus Speed	200 MHz for the 16750/51/52A Logic Analyzers. 167 MHz for the 16715/16/17/18/19A Logic Analyzers. 140 MHz for the 16557D Logic Analyzers. 110 MHz for the 16555A/D Logic Analyzers. 100 MHz for the 16556A/D Logic Analyzers. 100 MHz for the 16710/11/12A Logic Analyzers. 100 MHz for the 16550A Logic Analyzers.
Logic Analyzers Supported	16715/16/17/18/19/50/51/52A (one, two, or three cards) 16710/11/12A (one or two cards) 16550A (one or two cards) 16554A/55A/56A (one, two or three cards) 16555D/56D/57D (one, two or three cards)

Operating Characteristics	
Probes Required	The E8170B requires eight logic analyzer pods (136 channels) for traditional inverse assembly (with 64-bit data).
Signal Line Loading	Typically 100 k Ω plus 10 pF.
Setup/Hold Requirement	For all signals, the logic analyzers require a minimum combined setup/hold window. For 16715/16/17/18/19/50/51/52A logic analyzers the combined window must be at least 2.5 ns (1.25 ns using eye finder). For 16710/11/12A logic analyzers, the combined window must be at least 4.0 ns. For 16557 logic analyzers, the combined setup/hold time must be at least 3.0 ns. For 16550/54/55/56 logic analyzers, the combined setup/hold time must be at least 3.5 ns.

Glossary

Analysis Probe A probing solution connected to the target microprocessor. It provides an interface between the signals of the target microprocessor and the inputs of the logic analyzer. Formerly called a "preprocessor."

Background Debug Monitor Also called Debug Mode, In Background, and In Monitor. The normal processor execution is suspended and the processor waits for commands from the debug port. The debug port commands include the ability to read and write memory, read and write registers, set breakpoints and start the processor running (exit the Background Debug Monitor).

Debug Mode See *Background Debug Monitor*.

Debug Port A hardware interface designed into a microprocessor that allows developers to control microprocessor execution, set breakpoints, and access microprocessor registers or target system memory using a tool like the emulation probe.

Elastomeric Probe Adapter A connector that is fastened on top of a target microprocessor using a retainer and knurled nut. The conductive elastomer on the bottom

of the probe adapter makes contact with pins of the target microprocessor and delivers their signals to connection points on top of the probe adapter.

Emulation Migration The hardware and software required to use an emulation probe with a new processor family.

Emulation Module An emulation module is installed within the mainframe of a logic analysis system. An E5901A emulation module is used with a *target interface module* (TIM) or an analysis probe. An E5901B emulation module is used with an E5900B *emulation probe* and does not use a TIM.

Emulation Probe An emulation probe is a standalone instrument connected via LAN to the mainframe of a logic analyzer or to a host computer. It provides run control within an emulation and analysis test setup. Formerly called a "processor probe" or "software probe."

Emulator An emulation module or an emulation probe.

Extender A part whose only function is to provide connections from one location to another. One or more extenders might be stacked to

Glossary

raise a probe above a target microprocessor to avoid mechanical contact with other components installed close to the target microprocessor. Sometimes called a "connector board."

Flexible Adapter Two connection devices coupled with a flexible cable. Used for connecting probing hardware on the target microprocessor to the analysis probe.

Gateway Address An IP address 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 across networks or subnets, this address must be set to the address of the gateway machine.

General-Purpose Flexible Adapter A cable assembly that connects the signals from an elastomeric probe adapter to an analysis probe. Normally, a male-to-male header or transition board makes the connections from the general-purpose flexible adapter to the analysis probe.

High-Density Adapter Cable A cable assembly that delivers signals from an analysis probe hardware interface to the logic analyzer pod

cables. A high-density adapter cable has a single *MICTOR connector* that is installed into the analysis probe, and two cables that are connected to corresponding odd and even logic analyzer pod cables.

High-Density Termination Adapter Cable Same as a High-Density Adapter Cable, except it has a termination in the *MICTOR connector*.

In Background, In Monitor See *Background Debug Monitor*.

Inverse Assembler Software that displays captured bus activity as assembly language mnemonics. In addition, inverse assemblers may show execution history or decode control busses.

IP address Also called Internet Protocol address or Internet address. A 32-bit network address. It is usually represented as decimal numbers separated by periods; for example, 192.35.12.6.

Jumper Moveable direct electrical connection between two points.

JTAG (OnCE) port See *debug port*.

Label Labels are used to group and

Glossary

identify logic analyzer channels. A label consists of a name and an associated bit or group of bits.

Link-Level Address The unique address of the LAN interface. This value is set at the factory and cannot be changed. The link-level address of a particular piece of equipment is often printed on a label above the LAN connector. An example of a link-level address in hexadecimal: 0800090012AB. Also known as an LLA, Ethernet address, hardware address, physical address, or MAC address.

Mainframe Logic Analyzer A logic analyzer that resides on one or more board assemblies installed in a 16500, 1660-series, or 16600/700-series mainframe.

Male-to-male Header A board assembly that makes point-to-point connections between the female pins of a flexible adapter or transition board and the female pins of an analysis probe.

MICTOR Connector A high-density matched impedance connector manufactured by AMP Corporation. *High-density adapter cables* can be used to connect the logic analyzer to MICTOR connectors on the target system.

Monitor, In See *Background Debug Monitor*.

Pod A collection of logic analyzer channels associated with a single cable and connector.

Preprocessor See *Analysis Probe*.

Preprocessor Interface See *Analysis Probe*.

Probe Adapter See *Elastomeric Probe Adapter*.

Processor Probe See *Emulation Probe*.

Run Control Probe See *Emulation Probe* and *Emulation Module*.

Setup Assistant Wizard software program which guides a user through the process of connecting and configuring a logic analyzer to make measurements on a specific microprocessor. The setup assistant icon is located in the main system window.

Shunt Connector. See *Jumper*.

Solution A set of tools for debugging your target system. A solution includes probing, inverse assembly, the B4620B Source Correlation Tool

Glossary

Set, and an emulation module.

Stand-Alone Logic Analyzer A standalone logic analyzer has a predefined set of hardware components which provide a specific set of capabilities. A standalone logic analyzer differs from a mainframe logic analyzer in that it does not offer card slots for installation of additional capabilities, and its specifications are not modified based upon selection from a set of optional hardware boards that may be installed within its frame.

State Analysis A mode of logic analysis in which the logic analyzer is configured to capture data synchronously with a clock signal in the target system.

Subnet Mask A subnet mask blocks out part of an IP address so the networking software can determine whether the destination host is on a local or remote network. It is usually represented as decimal numbers separated by periods; for example, 255.255.255.0.

Symbol Symbols represent patterns and ranges of values found on labeled sets of bits. Two kinds of symbols are available:

1) Object file symbols — Symbols from your source code, and symbols

generated by your compiler. Object file symbols may represent global variables, functions, labels, and source line numbers.

2) User-defined symbols — Symbols you create.

Target Board Adapter A daughter board inside the E5900B emulation probe which customizes the emulation probe for a particular microprocessor family. The target board adapter provides an interface to the ribbon cable which connects to the debug port on the target system.

Target Control Port An 8-bit, TTL port on a logic analysis system that you can use to send signals to your target system. It does not function like a pattern generator or emulation module, but more like a remote control for the target's switches.

Target Interface Module A small circuit board which connects the 50-pin cable from an E5901A emulation module or E5900A emulation probe to signals from the debug port on a target system. Not used with the E5900B emulation probe.

TIM See *Target Interface Module*.

Timing Analysis A mode of logic analysis in which the logic analyzer is configured to capture data at a rate

determined by an internal sample rate clock, asynchronous to signals in the target system.

Transition Board A board assembly that obtains signals connected to one side and rearranges them in a different order for delivery at the other side of the board.

Trigger Specification A set of conditions that must be true before the instrument triggers. See the printed or online documentation of your logic analyzer for details.

1/4-Flexible Adapter An adapter that obtains one-quarter of the signals from an elastomeric probe adapter (one side of a target microprocessor) and makes them available for probing.

Symbols

% prefix, 104
* symbol in listing, 104
? symbol in listing, 104

A

access to source code files, 114
acquire data, 118
ADDR label, modifying, 69
addresses
 branch target, 104
 mask, 94
 offset, 86
analysis arm, 129
analysis mode
 state, 115
analysis probe
 definition, 161
 equipment required, 18
 equipment supplied, 18
 inverse assembly, 94
 operating characteristics, 158
 overview, 2
 power on/power off sequence, 44
 processors supported, 5
 product numbers, 5
analysis probe problems
 erratic trace measurements, 148
 target system will not boot, 147
analyzer problems, 145
 capacitive loading, 148
 intermittent data errors, 145
 unwanted triggers, 145
arm, 129, 131
arm event, 129, 131
arm signal, 128
assembly-level listing, 118
assistant
 See setup assistant

B

B4620B source correlation tool set,
 118, 119, 120, 122
background debug monitor, 161
bits
 labels, 69
 LSB and MSB, 69
branch exception disassembly, 108
branch instructions, 104
break emulator on trigger, 121
break into monitor, 128
break temporarily, 128
breakpoint, 125, 130
 triggering analyzer on, 130
breakpoint registers, 130
breakpoints, 126
 software, 130
bus activity, processor, 125

C

cache
 trace problems and, 150
cache-on execution tracker
 performance hints, 106
cache-on trace reconstruction,
 105, 109
caches
 enabling and disabling, 110
captured data, source code
 associated with, 112
captured execution, displaying, 91
cards
 See logic analyzers
CD-ROM, installing software from,
 47
Chart tool, 118
checklist, setup, 16
clock speed
 processor, 123
clocks
 slow, 155
color, 103

comments, in GPA files, 142
configuration
 checklist, 16
configuration files, 93
 installing, 63
 list of, 67
 loading, 66
Configuration window, 123
configuring the logic analyzer, 63
connection
 analysis probe, 49
 emulation module, 45
 setup checklist, 16
connector
 JTAG, designing, 41
connector board, 161
connector, MICTOR, 25, 27
control execution, 118
coordinating logic analysis, 117
crash, system, 125

D

DATA label, modifying, 69
debug mode, 161
debug port, 161
 definition, 161
debugger, 118, 119, 125, 128, 129,
 130
decoding
 exception, 100
 simplified mnemonic, 97
deep memory logic analyzer, 113
disassembly, branch exception,
 108
display timing analysis mode data,
 115
Display tools, 118
displaying captured execution, 91

E

elastomeric probe adapter
 definition, 161

- Emulation Control Interface, 118, 123, 125, 126, 129, 130
 - emulation migration
 - definition, 161
 - emulation module, 118, 125, 129
 - connected to Port Out, 127
 - definition, 161
 - description of, 3
 - icon, 122
 - Port In connected to, 127
 - product numbers, 5
 - stop processor on trigger, 122
 - trigger, 120
 - trigger signal, 126
 - triggering logic analyzers, 126
 - emulation modules
 - connected in series, 127
 - emulation probe
 - definition, 161
 - emulation solution, 3
 - See* solution
 - equipment required
 - analysis probe, 18
 - equipment supplied, 18
 - analysis probe, 18
 - ordering information, 5
 - overview, 5
 - Ethernet networks, 114
 - examples of measurements, 119
 - exception decoding, 100
 - extender, 161
- F**
- false trigger, 128
 - files
 - loading vs. installing, 46
 - filters, inverse assembler, 102
 - flexible adapter
 - definition, 162
 - flowchart, setup, 16
 - format menu, 69
 - ftp, 114
- G**
- full solution, 3
 - function calls, 125
 - FUNCTIONS in GPA format, 139
- H**
- gateway address
 - definition, 162
 - General-Purpose ASCII format
 - comments, 142
 - FUNCTIONS, 139
 - record format summary, 136
 - SECTIONS, 138
 - SOURCE LINES, 141
 - START ADDRESS, 142
 - VARIABLES, 140
 - general-purpose flexible adapter
 - definition, 162
 - glitches, 119
 - Group Run, 127, 128, 129, 130
- I**
- halt the processor, 121
 - hardware breakpoint registers, 130
 - high-density adapter cable
 - definition, 162
 - high-density termination adapter
 - definition, 162
 - high-level source code, 118
- J**
- intermodule measurement
 - problems, 152
 - an event wasn't captured, 152
 - Intermodule window, 120, 122, 128, 129, 130
 - Invasm menu, 93
 - inverse assembled data, 104
 - inverse assembler, 93, 113
 - definition, 162
 - filename, 93
 - filters, 102
 - loading, 94
 - operating modes, 109
 - preferences, 94
 - requirements for, 94
 - inverse assembler problems, 149
 - failure to load or run, 151
 - incorrect inverse assembly, 149
 - no inverse assembly, 149
 - inverse assembly, 94
 - cache-on, 109
 - Pods required, 20, 159
 - traditional, 105, 109
 - IP address
 - definition, 162
- K**
- 18620E inverse assembler file, 93
 - illegal instruction, 130
 - illegal opcode, 104
 - In Monitor, 126, 127, 128
 - information sources, 4
 - installation, software, 43
 - instruction cache
 - See* cache
 - instruction decoding, 97
 - intermodule measurement, 123, 126
 - trigger, 125
- L**
- labels
 - definition, 162
 - LAN protocols, 114
 - LAN system administrators, 114
 - latency, 123
 - library code execution, 89
 - link-level address
 - definition, 163
 - listing
 - incorrect, 149
-

- Listing display window, 92
- Listing tool, 118
- Load menu, 94
- loading configuration files, 66
- loading configurations, vs.
 - installing, 63
- logic analysis
 - coordinating, 117
- logic analyzer
 - configuring, 66
 - deep memory, 113
 - stopping, 123
 - storage qualification, 89
 - trigger setup, 81
- logic analyzers
 - 16550A connections, 57
 - 16554/55/56/57 connections, 59, 60
 - 16600 and 16700-series, 17
 - 16710/11/12A connections, 55
 - 16715/16/17/18/19A connections, 52
 - supported, 20
- LSB, 69

- M**
- mainframe logic analyzer
 - definition, 163
- male-to-male header
 - definition, 163
- measurement examples, 119
- measurement, viewing, 123
- memory, 118
- memory banks, 94
- Memory Disassembly window, 118
- microprocessor bus cycles, 103
- microprocessors supported, 5
- MICTOR connector, 25, 27
- MICTOR connector, definition, 163
- mnemonics, 97
- modules, logic analyzer, 45
- monitor, background debug, 161

- MSB, 69

- N**
- network access to source files, 114
- NFS client/server, 114
- nostate, trigger on, 125
- NULL pointer de-references, 119

- O**
- occurrences remaining in level 1, 129, 131
- offset, address, 86
- on-chip hardware breakpoint
 - registers, 130
- online configuration help, 17
- operating modes, inverse
 - assembler, 109
- Options menu, 94
- oscilloscope, 129, 131
 - measurement, 125

- P**
- parts supplied, 18
- performance, profile system, 119
- Pods, logic analyzer, 163
- Port In, 127
- Port Out, 127
- power on/power off sequence, 44
- preprocessor
 - See* analysis probe
- preprocessor interface
 - See* *analysis probe*
- problems
 - analysis probe, 143
- processor activity, 125
- processor bus, 131
 - activity, 125
 - state analysis of, 125
- processor clock speed, 123

- processor execution
 - coordinating, 117
 - stopping on analyzer trigger, 120
 - stops trigger the analyzer, 126
- processor halt, 129, 130, 131
 - trace before, 125
 - triggering on, 129
- processor support package, 47
- processors supported, 5
- profile system performance, 119
- program counter, 118, 119, 128
- program stack, 128

- Q**
- qualified processor clock, 129, 131

- R**
- record format, General-Purpose
 - ASCII, 136
- references, 4
- registers, 118
 - listing format, 104
- run control tool
 - See* emulation control interface
- Run Control window, 121, 123, 129, 131
- Running, emulation module state, 127

- S**
- search path, source code, 89, 114
- SECTIONS in GPA format, 138
- setting breakpoints, 130
- Setup Assistant, 17
- setup assistant, 17
 - definition, 163
- setup checklist, 16
- skid effect, 123
- slow clock error message, 129, 131
- software
 - installing, 43
 - software addresses, 113

- software breakpoints, 130
 - solution
 - at a glance, 2
 - definition, 163
 - solutions
 - description of, 2
 - product numbers, 5
 - source code, 118
 - associated with captured data, 112
 - search path, 89
 - source correlation tool set, 114
 - source files
 - network access to, 114
 - search path, 114
 - version control, 114
 - source line trigger, stopping
 - processor execution, 120
 - SOURCE LINES in GPA format, 141
 - Source Viewer window, 119, 120, 122
 - Source window, 120
 - source-level listing, 119
 - speed, processor clock, 123
 - stack, program, 128
 - stand-alone logic analyzer
 - definition, 164
 - START ADDRESS in GPA format, 142
 - STAT
 - modifying, 69
 - state analysis, 125, 164
 - definition, 164
 - state analyzer, 131
 - state mode, 115
 - state of the processor, 118
 - status display, 129
 - stop on any trigger, 122
 - stop the analyzer, 123
 - storage qualification, 89
 - subnet mask
 - definition, 164
 - suppressing all other operations, 103
 - SW_ADDR label, 113
 - symbols
 - definition, 164
 - predefined, 70, 72
 - symbols, displaying, 77
 - system administrators, 114
 - system crash, 125
 - System Performance Analyzer tool, 118
 - system performance, profile, 119
- T**
- target board adapter
 - definition, 164
 - target control port, 164
 - target interface module (TIM)
 - definition, 164
 - target system, 118
 - boot failure, 147
 - power sequence, 44
 - requirements for analysis probe, 24
 - TCP/IP protocol, 114
 - telnet, 114
 - temporary breaks, 128
 - timing analysis, 125, 164
 - definition, 164
 - timing analysis mode
 - data, displaying, 115
 - timing analyzer, 129, 131
 - tools, 118
 - trace
 - erratic, 148
 - missing display, 146
 - trace about this line, 120
 - trace reconstruction, cache-on, 105
 - trace until processor halt, 126
 - transition board
 - definition, 165
- U**
- unknown opcode, 104
 - unnneeded information, 103
- V**
- VARIABLES in GPA format, 140
 - version control, source file, 114
 - view a measurement, 123
- W**
- Waveform display, 115
 - web sites
 - Agilent logic analyzers, 4
 - See Also under debugger names
 - wizard
 - See setup assistant
- X**
- X-Window client/server, 114

© Copyright Agilent Technologies
1994-2000
All Rights Reserved.

Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subparagraph (C) (1) (ii) of the Rights in Technical Data and Computer Software Clause in 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 set forth in FAR
52.227-19 (c) (1,2).

Document Warranty

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

Agilent Technologies shall not be liable for errors contained herein or for damages in connection with the furnishing, performance, or use of this material.

Safety

This apparatus has been designed and tested in accordance with IEC Publication 1010, Safety Requirements for Measuring Apparatus, and has been supplied in a safe condition. This is a Safety Class I instrument (provided with terminal for protective earthing). Before applying power, verify that the correct safety precautions are taken (see the following warnings). In addition, note the external markings on the instrument that are described under "Safety Symbols."

Warning

- Before turning on the instrument, you must connect the protective earth terminal of the instrument to the protective conductor of the (mains) power cord. The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. You must not negate the protective action by using an extension cord (power cable) without a protective conductor (grounding). Grounding one conductor of a two-conductor outlet is not sufficient protection.
- Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuseholders. To do so could cause a shock or fire hazard.

- Service instructions are for trained service personnel. To avoid dangerous electric shock, do not perform any service unless qualified to do so. Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.

- If you energize this instrument by an auto transformer (for voltage reduction), make sure the common terminal is connected to the earth terminal of the power source.

- Whenever it is likely that the ground protection is impaired, you must make the instrument inoperative and secure it against any unintended operation.

- Do not operate the instrument in the presence of flammable gasses or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.

- Do not install substitute parts or perform any unauthorized modification to the instrument.

- Capacitors inside the instrument may retain a charge even if the instrument is disconnected from its source of supply.

Safety Symbols



Instruction manual symbol: the product is marked with this symbol when it is necessary for you to refer to the instruction manual in order to protect against damage to the product.



Hazardous voltage symbol.



Earth terminal symbol: Used to indicate a circuit common connected to grounded chassis.

WARNING

The Warning sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a Warning sign until the indicated conditions are fully understood and met.

CAUTION

The Caution sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a Caution symbol until the indicated conditions are fully understood or met.

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

Product Warranty

This Agilent Technologies product has a warranty against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Agilent Technologies will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies.

For products returned to Agilent Technologies for warranty service, the Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to the Buyer. However, the 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 software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental 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 or fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are the 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.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products. For any assistance, contact your nearest Agilent Technologies Sales Office.

Certification

Agilent Technologies Company 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 Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

About this edition

This is the *Logic Analysis Support for the Motorola MPC7400 User's Guide*.

Publication number
E8170-97001, October 2000
Printed in USA.

The information in this manual previously appeared in:
E8170-97000 December 1999
E2498-97004 December, 1999
E2498-97003 December 1998
E2498-97002 October 1998
E2498-97001 January 1998
E3454-97000 September 1997

New editions are complete revisions of the manual. Many product updates 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 updates.