

User's Guide

Publication number E2498-97006
September 2000

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

© Copyright Agilent Technologies 1994-2000
All Rights Reserved

Logic Analysis Support for the PowerPC 7XX

Logic Analysis Support for the PowerPC 7XX— At a Glance

The Agilent Technologies E9603A inverse assembler, in conjunction with an Agilent Technologies logic analyzer, allows you to view PowerPC 7XX assembly instructions that are executing in your target system.

The inverse assembler is identified as “Agilent Technologies E2498A” in the Setup Assistant.

If You Purchased an emulation solution

The E9486A Emulation Solution lets you use an Agilent Technologies 16600/16700-series logic analysis system to debug and characterize PowerPC 7XX target systems. 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.

For more information on an emulation solution

The *Emulation for the PowerPC 7XX User's Guide* describes setting up and using the emulation probe and emulation module.

Information about using the logic analysis system with the emulation probe/module can be found in Chapter 9, “Coordinating Logic Analysis with Processor Execution,” beginning on page 171.

Additional Equipment Included in an Emulation Solution

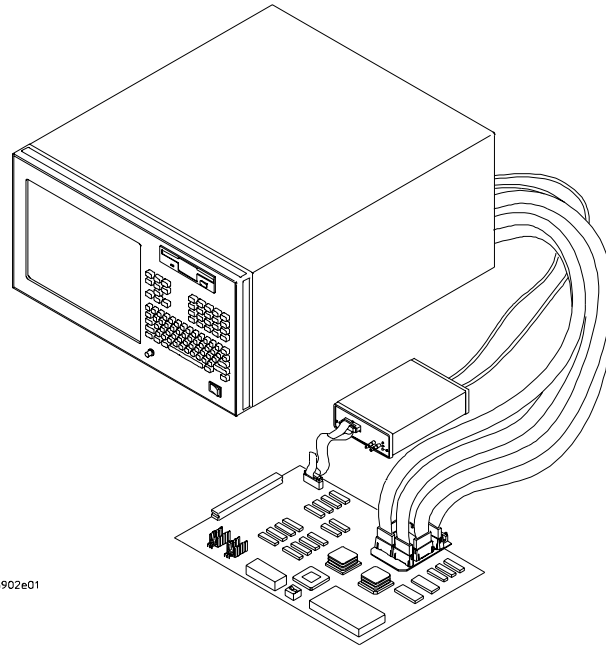
Emulation Probe

The emulation module plugs into your Agilent Technologies 16600/700-series logic analysis system frame, and the emulation probe connects to the emulation module and the JTAG port on your target system. The emulation probe lets you use a microprocessor's built-in debugging features including run control and access to registers and memory. A high-level source debugger can use the emulation probe/module to debug code running on the target system.

Source Correlation Tool Set

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

Emulation Solution



e5902e01

In This Book

This book documents the following product:

Processors supported	Product ordered	Includes
PPC 740, PPC 745, PPC 750, and PPC 755	E9586A Option #001	E2498A inverse assembler

Related equipment

The following equipment is included in the PowerPC 7XX emulation solution:

Processors supported	Product ordered	Includes
PPC 740, PPC 745, PPC 750, and PPC 755	E9486A Option #001	E2498A inverse assembler, emulation probe, emulation module, 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 20.

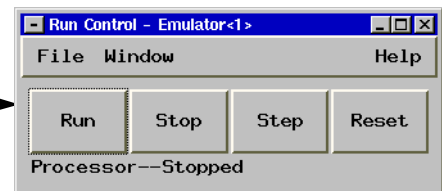


Use the appropriate Run button



Click here to start a measurement.

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



Additional Information Sources

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

If you have a probing adapter, the instructions for connecting the probe to your target system are in the **Probing Adapter** documentation.

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 Technologies 16600- or 16700-series logic analysis system, the **online help** for the Emulation Control Interface has additional information on using the emulation probe.

The **measurement examples** include valuable tips for making emulation and analysis measurements. You can find the measurement examples under the system help in your Agilent Technologies 16600/700-series logic analysis system.

Contents

If You Purchased an emulation solution	2
In This Book	4
Related equipment	4

1 Equipment and Requirements 17

Setup Checklist	19
Setup Assistant	20
Inverse Assembler Software	22
Equipment supplied	22
Additional equipment required	22
Additional equipment supported	23
Compatible Logic Analyzers	24
Logic analyzer software version requirements	25
Emulation Solution	26

2 Preparing the Target System 27

Designing Logic Analyzer Connectors into Your Target System	29
Using High-Density Connectors	29
Inverse assembler—signal-to-connector mapping	33
Designing a Debug Port Connector into Your Target System	44

3 Setting Up the Logic Analysis System 45

- Power-on/Power-off Sequence 46
 - To power on 16600 and 16700-series logic analysis systems 46
 - To power on all other logic analyzers 47
 - To power off 47

- Installing Logic Analyzer Modules 48
 - Installing an emulation module 48

- Installing Software 49
 - To install software from CD-ROM (16600/700-series logic analysis systems) 50

4 Probing the Target System 53

- Connecting the Logic Analyzer to the Target System 54
 - 64-bit data 54
 - 32-bit data 55
 - No data 55
 - To connect the high-density termination cables to the target system 56
 - To connect to the 16715/16/17/18/19A and 16750/51/52 logic analyzers (one card) 57
 - To connect to the 16715/16/17/18/19A and 16750/51/52 logic analyzers (two cards) 58
 - To connect to the 16715/16/17/18/19A and 16750/51/52 logic analyzers (three cards) 59
 - To connect to the 16710/11/12A logic analyzer (one card) 60
 - To connect to the 16710/11/12A logic analyzer (two cards) 61
 - To connect to the 16603A logic analyzer 62
 - To connect to the 16602A logic analyzer 63
 - To connect to the 16601A logic analyzer 64

To connect to the 16600A logic analyzer	65
To connect to the 16554/55/56/57 (one-card)	66
To connect to the 16554/55/56/57 (two-cards)	67
To connect to the 16554/55/56/57 (three-cards)	68
To connect to the 16550A analyzer (one card)	69
To connect to the 16550A analyzer (two cards)	70
To connect to the 1670A/D logic analyzer	71
To connect to the 1660A/C logic analyzers	72

5 Configuring the 16600/700-Series Logic Analyzer 73

Configuring 16600/700-series Logic Analysis Systems	75
To load configuration files (and the inverse assembler) from hard disk	76
To load configuration files (and the inverse assembler) from floppy disk	77
To list software packages that are installed	78
Logic analyzer configuration files (16600/700-series)	78
Logic Analyzer Configuration	81
Format menu	81
Trigger dialog	86
Using the Inverse Assembler	88
Using Cache-On Trace Reconstruction	88
To minimize effects of cache-on trace on system performance	89
Enabling branch exception disassembly	91
Inverse Assembler Modes of Operation	92
To use the Invasm menu	93
Loading the Inverse Assembler	93

Contents

Setting the Inverse Assembler Preferences	94
To set the inverse assembler preferences	94
To set the memory map preferences	95
To set the processor options preferences	96
To set the decoding options preferences	98
To set the opcode source preferences	103
To enable/disable the instruction cache on the PPC7XX	104
Loading Symbol Information	108
To load object file symbols	108
To view predefined symbols for the PPC7XX	109
To create user-defined symbols	110
Symbol use requirements	111
To use object file symbols in the 16600/700	112
Compilers for PPC7XX	113
Inverse assembler generated PC (software address) label	115
Triggering on Symbols and Source Code	116
Using trigger alignment	116
To correlate relocatable code using the address offset	117
Using storage qualification	118
Modes of Operation	119
State-per-ack mode	119
State-per-clock mode	119
Timing mode	120
Modes of Analysis	121
Inverse assembly analysis	121

6 Configuring the 1660/1670/16500B/C-Series Logic Analyzer 123

Configuring 1660/1670/16500B/C-Series Logic Analysis Systems	125
To load configuration files and the inverse assembler—1660/1670/16500B/C-series logic analyzers	126
Logic analyzer configuration files (1660/1670/16500B/C-series logic analyzers)	127
To select a different inverse assembler	128
Modes of Operation	129
State-per-ack mode	129
State-per-clock mode	129
Timing mode	130
Logic Analyzer Configuration	131
Format menu	131
Trigger dialog	137
Using the Inverse Assembler	139
To disable the instruction cache on the PPC7XX	139
To display captured state data	141
Inverse assembler output format	143
To use the Invasm menu (1660, 1670, and 16500B/C mainframes)	146

7 Capturing Processor Execution 151

- Trigger sequence 153

- Setting Up Logic Analyzer Triggers 154
 - To set up logic analyzer triggers 154

- Triggering on Symbols and Source Code 156
 - Using the Address Offset 156
 - Using Storage Qualification 156
 - To qualify stored data 157

8 Displaying Captured Processor Execution 159

- To display captured state data 160
- To display symbols 161
- To interpret the inverse assembled data 162
- To use the inverse assembler filters 164
- To view the source code associated with captured data 166
- To display captured timing analysis mode data 169

9 Coordinating Logic Analysis with Processor Execution 171

- What are some of the tools I can use? 172
- Which assembly-level listing should I use? 172
- Which source-level listing should I use? 173
- Where can I find practical examples of measurements? 173

Contents

Triggering the Emulation Probe from the Analyzer	174
To stop the processor when the logic analyzer triggers on a line of source code (Source Viewer window)	174
To stop the processor when the logic analyzer triggers (Intermodule window)	176
To minimize the "skid" effect	177
To stop the analyzer and view a measurement	178
Tracing Until the Processor Halts	179
To capture a trace before the processor halts	180
Triggering the Logic Analyzer from the Emulation Probe	181
The emulation probe trigger signal	181
Group Run	182
Debuggers can cause triggers	184
To trigger the analyzer when the processor halts - timing mode	185
To trigger the analyzer when the processor reaches a breakpoint	187

10 General-Purpose ASCII (GPA) Symbol File Format 189

General-Purpose ASCII (GPA) Symbol File Format	190
GPA Record Format Summary	192
SECTIONS	194
FUNCTIONS	195
VARIABLES	196
SOURCE LINES	197
START ADDRESS	198
Comments	198

11 Specifications and Characteristics 199

Operating Characteristics 200

12 Troubleshooting the Logic Analyzer 203

Logic Analyzer Problems 205

Intermittent data errors 205

Unwanted triggers 206

No activity on activity indicators 206

No trace list display 207

Analyzer won't power up 207

Analysis Probe Problems 208

Target system will not boot up 208

Erratic trace measurements 209

Capacitive loading 209

Inverse Assembler Problems 210

No inverse assembly or incorrect inverse assembly 210

Inverse assembler will not load or run 212

If the inverse assembler cannot determine cycle sizes 212

Intermodule Measurement Problems 213

An event wasn't captured by one of the modules 213

Contents

Analysis Probe Messages	214
“... Inverse Assembler Not Found”	214
“Measurement Initialization Error”	215
“No Configuration File Loaded”	216
“Selected File is Incompatible”	216
“Slow or Missing Clock”	216
“Time from Arm Greater Than 41.93 ms”	217
“Waiting for Trigger”	217
Returning Parts to Agilent Technologies for Service	218
To return a part to Agilent Technologies	218
To obtain replacement parts	219
Cleaning the Instrument	220
Glossary	221
Index	227

Equipment and Requirements

Chapter 1: Equipment and Requirements

This chapter describes:

- Setup Checklist
- Setup Assistant
- Equipment used with the inverse assembler
- List of compatible logic analyzers
- Emulation Solution

Setup Checklist

Follow these steps to connect your equipment:

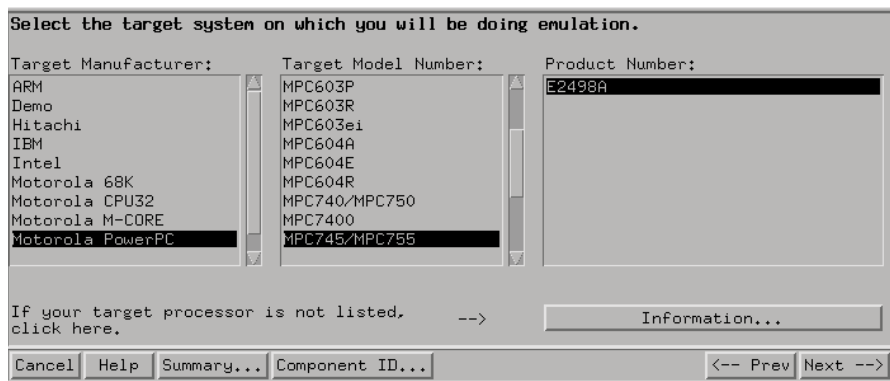
- Check that you received all of the necessary equipment. See page 22.
- If you need to install an emulation module in an Agilent Technologies 16600/700-series logic analysis system, see your emulation manual.
- Install the software. See page 49.
- If you have an Agilent Technologies 16600/700-series logic analysis system, use the Setup Assistant to help you connect and configure your system. See page 20. If you do not have a 16600/700-series logic analysis system, connect the logic analyzer cables. See page 54.

Setup Assistant

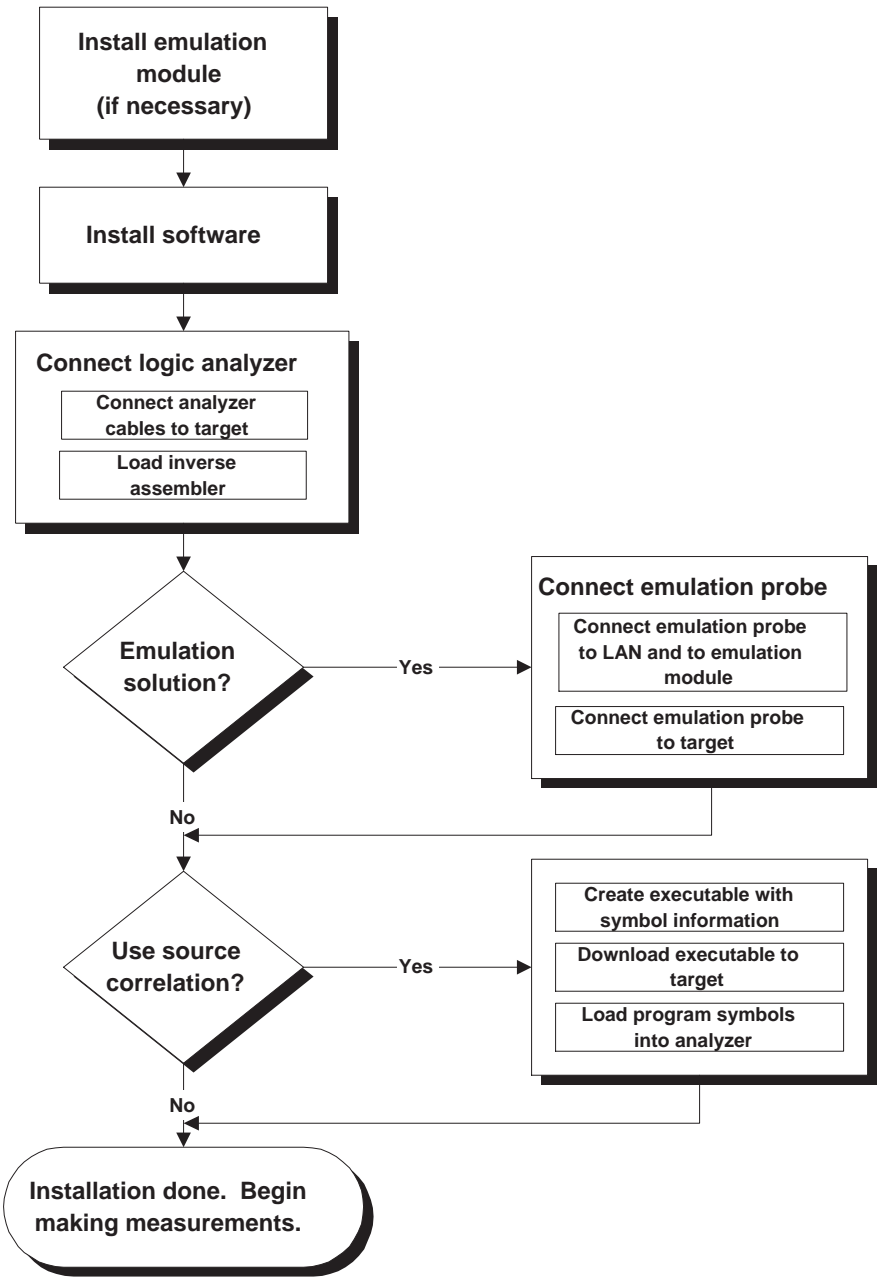
The Setup Assistant is the online tool for connecting and configuring your logic analysis system for microprocessor and bus analysis. The Setup Assistant is available on the HP/Agilent Technologies 16600A and 16700-series logic analysis systems. You can use the Setup Assistant in place of the connection and configuration procedures provided in this manual.

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

Start the Setup Assistant by clicking  in the system window.



If you ordered this product with your Agilent Technologies 16600/700-series logic analysis system, the logic analysis system has the latest software installed, including support for this product.



E2498F02.VSI

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 Agilent Technologies E2498A consists of the following:

- Logic analyzer configuration files and the inverse assembler on a CD-ROM (for Agilent Technologies 16600/700 series logic analysis systems).
 - Logic analyzer configuration files and the inverse assembler on a 3.5-inch disk (for other Agilent Technologies logic analyzers).
 - This User's Guide.
-

Additional equipment required

In addition to the items listed above, the following is required to analyze a PowerPC 7XX target circuit:

- Connector headers on your target system which supply signals to the logic analyzer.
- Agilent Technologies termination adapter cables to attach your target system to a logic analyzer.
- One of the logic analyzers listed in the table on page 24.

Additional equipment supported

Agilent Technologies B4620B Source Correlation Tool Set.

The inverse assembler may be used with the Agilent Technologies B4620B Source Correlation Tool Set on a 16600/700-series logic analysis system. The software is already installed on the Agilent Technologies 16600/700-series logic analysis system's disk. All you need is the entitlement certificate for licensing the source correlation tool set software. The CD-ROM is included in case you need to re-install the software.

Compatible Logic Analyzers

Compatible Logic Analyzers

The following table lists the logic analyzers supported by the E2476B and E2498A inverse assembler software. Logic analyzer software version requirements are shown on page 25.

The software requires eight logic analyzer pods (136 channels) for inverse assembly. The analysis probe contains two additional pods you can monitor.

Logic Analyzers Supported

Agilent Technologies Logic Analyzer	Channel Count	State Speed *	Timing Speed *	Memory Depth
1660A/AS/C/CS/CP/E/ES/EP	136	100 MHz	250 MHz	4 k states
1670A	136	70 MHz	125 MHz	64 k or .5 M states
1670D	136	100 MHz	125 MHz	64 k or 1 M states
1670E	136	100 MHz	125 MHz	1 M
16550A (2 cards)	102/card	100 MHz	500 MHz	4 k states
16554A (2 or 3 cards)	68/card	70 MHz	250 MHz	512 k states
16555A (2 or 3 cards)	68/card	110 MHz	250 MHz	1 M states
16555D (2 or 3 cards)	68/card	110 MHz	500 MHz	2 M states
16556A (2 or 3 cards)	68/card	100 MHz	200 MHz	1 M states
16556D (2 or 3 cards)	68/card	110 MHz	400 MHz	2 M states
16557D (2 or 3 cards)	68/card	140 MHz	500 MHz	2 M states
16600A	204	100 MHz	125 MHz	64 k states
16601A	136	100 MHz	125 MHz	64 k states
16710A (2 cards)	102/card	100 MHz	500 MHz	8 k states
16711A (2 cards)	102/card	100 MHz	500 MHz	32 k states
16712A (2 cards)	102/card	100 MHz	500 MHz	128 k states
16715A (2 or 3 cards)	68/card	167 MHz	667 MHz	2 M states
16716A (2 or 3 cards)	68/card	167 MHz	2 GHz	512 k states
16717A (2 or 3 cards)	68/card	333 MHz	2 GHz	2 M states
16718A (2 or 3 cards)	68/card	333 MHz	2 GHz	8 M states
16719A (2 or 3 cards)	68/card	333 MHz	2 GHz	32 M states
16750A (2 or 3 cards)	68/card	400 MHz	2 GHz	4/8 M states
16751A (2 or 3 cards)	68/card	400 MHz	2 GHz	16/32 M states
16752A (2 or 3 cards)	68/card	400 MHz	2 GHz	32/64 M states

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

Logic analyzer software version requirements

The logic analyzers must have software with a version number greater than or equal to those listed below to make a measurement with the E2498A. You can obtain the latest software at the following web site:

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

If your software version is older than those listed, load new system software with the higher version numbers before loading the E2498A software. See “Installing Software” on page 49 for instructions for loading software.

Logic Analyzer Software Version Requirements

Agilent Technologies Logic Analyzer	Minimum Logic Analyzer Software Version for use with E2476B/77A
16600-series	The latest 16600 logic analyzer software version is on the CD-ROM shipped with this product.
1660-series	Software version A.02.01
1670-series	Software version A.02.02
Mainframes*	
16700-series	The latest 16700 logic analyzer software version is on the CD-ROM shipped with this product.
16500C Mainframe	Software version A.01.07
16500B Mainframe	Software version A.03.14

* The mainframes are used with the Agilent Technologies 16550 and 16554/55/56/57 logic analyzers. The 16557D requires the 16500C mainframe or the 16600/700-series logic analysis system.

Emulation Solution

If you ordered an emulation solution you received an emulation probe, an emulation module, and accessories, which are described in the *Emulation for the PowerPC 7XX User's Guide*.

The combination of an inverse assembler, an emulation probe, and an Agilent Technologies 16600- or 16700-series logic analysis system lets you both view PowerPC 7XX assembly instructions that are executing on your target system and use the target processor's built-in JTAG debugging features.

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 using the logic analysis system.

Preparing the Target System

Chapter 2: Preparing the Target System

This chapter describes the necessary steps for connecting your PowerPC 7XX target system to a logic analyzer on which the inverse assembler can operate.

Because PowerPC 7XX circuits will vary with each design, it is important that you design into your target system the headers to which a logic analyzer can connect. If your system already contains connector headers with incompatible pinouts, you can still connect to a logic analyzer using an adapter cable and General Purpose probes.

Designing Logic Analyzer Connectors into Your Target System

The logic analyzer can be connected directly to connectors on your target system. This section describes what kind of connectors to use, and how to connect the correct signals to the connectors.

Using High-Density Connectors

High-density MICTOR (*Matched Impedance Connector*) connectors are recommended for connecting the target system to the logic analyzer because they require less board space and provide higher signal integrity than medium-density connectors. Each connector carries 32 signals and two clocks.

- Each 32-signal high-density header connector requires approximately 1.1" x 0.4" of printed-circuit board space. A minimum of four connectors (eight logic analyzer pods) is required; a fifth connector may be used.
- The part number for the high-density MICTOR connector is: AMP P/N 2-767004-2 or Agilent P/N 1252-7431.
- Each MICTOR connector requires one Agilent Technologies E5346A high-density termination adapter cable to attach to the logic analyzer. This is a Y-cable where the single end connects to the high-density header connector, and each of the two opposite ends connects to a logic analyzer pod.
- Any probed signal line must be able to supply a minimum of 600 mV to the probe tip and handle a minimum loading of 90 k Ω shunted by 10 pF. The maximum input voltage for the logic analyzer is +/- 40 volts peak.
- If a printed-circuit board already has a header connector attached, but the signal pinouts do not match the requirement, an adapter (Agilent part number E5346-60002) can be used to route the signals to the correct pods.
- A plastic shroud (Agilent part number E5346-44701) is available to secure the mechanical connection of the high-density cable to the MICTOR header connector.

Designing Logic Analyzer Connectors into Your Target System

See Also

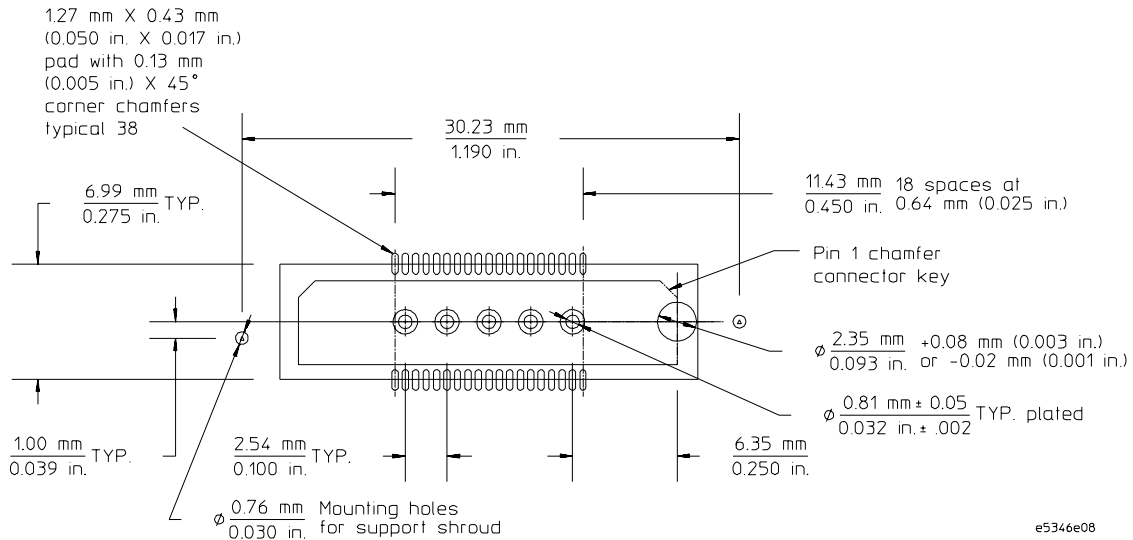
More information on this connector is available in Portable Document Format (PDF) from the web site:

<http://www.tm.agilent.com/>

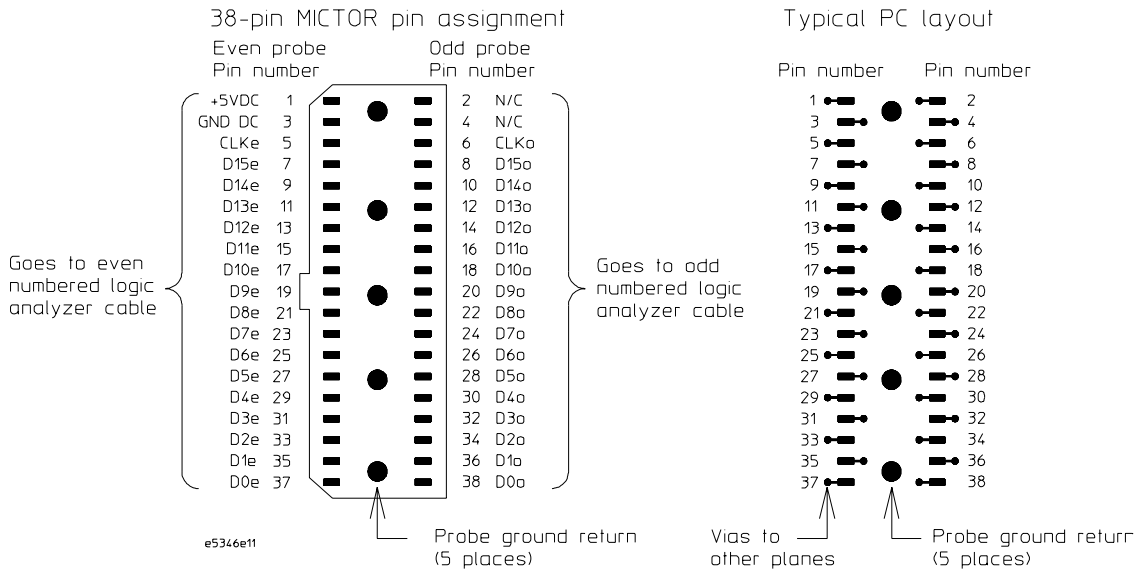
When you reach this web site, search on E5346A.

High-Density Connector Mechanical Specifications

Dimensions of the AMP MICTOR 2-767004-2 surface mount connector are shown below. The holes for mounting a support shroud are off-center to allow 0.40 in (1.20 mm) centers when using multiple connectors.



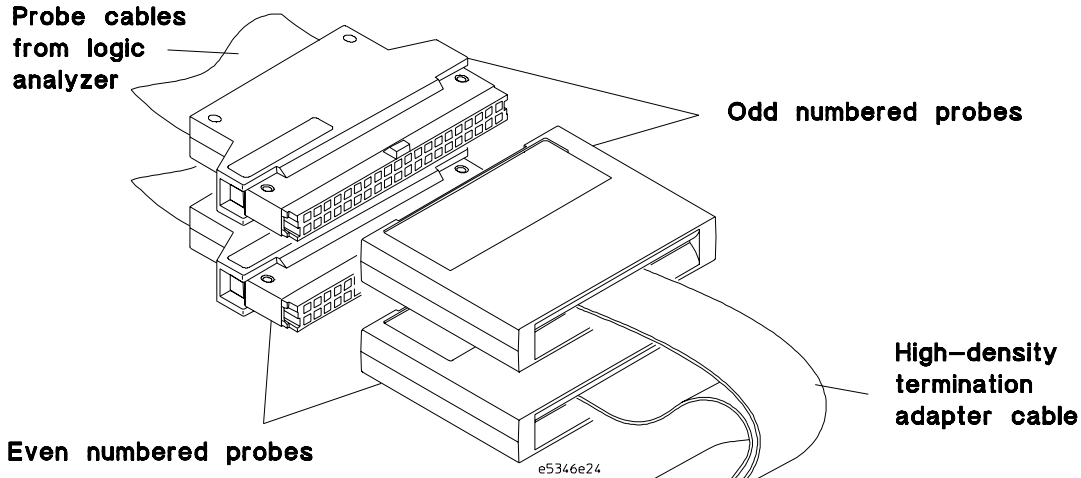
The high-density connector pin assignment and recommended circuit board routing are shown in the following figure.



Five center inline pins on the connector are the signal ground returns and must be connected to ground.

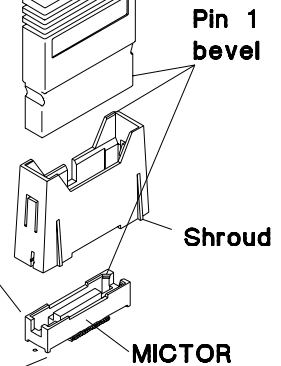
Chapter 2: Preparing the Target System
Designing Logic Analyzer Connectors into Your Target System

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	Reserved. Do not use.
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	



Inverse assembler—signal-to-connector mapping

The following tables show the electrical signal-to-connector mapping required by the Agilent Technologies E2498A 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.

Recommended Configuration Connection Notes

- 'nc' pins **MUST** be a true no-connect on the target. The signals are used for other functions unavailable to target probing.
- Five center inline pins on the connector are the signal ground returns and must be connected to ground.
- Any blank pins can be used for user defined signals.
- '#' or overscore denotes an active low signal.

Designing Logic Analyzer Connectors into Your Target System

PowerPC 7XX 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

PowerPC 7XX 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

Designing Logic Analyzer Connectors into Your Target System

PowerPC 7XX 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-

PowerPC 7XX 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-

Designing Logic Analyzer Connectors into Your Target System

PowerPC 7XX 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

PowerPC 7XX 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

Designing Logic Analyzer Connectors into Your Target System

PowerPC 7XX 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

PowerPC 7XX 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

Designing Logic Analyzer Connectors into Your Target System

PowerPC 7XX 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		LSSDMO
32	3	PLLCF0	PLLCFG	
34	2	PLLCF1	PLLCFG	
36	1	PLLCF2	PLLCFG	
38	0	PLLCF3	PLLCFG	

PowerPC 7XX 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 Debug Port Connector into Your Target System

Refer to the *Emulation for the PowerPC 7XX User's Guide* for information on designing a debug port.

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 16600 and 16700-series logic analysis systems

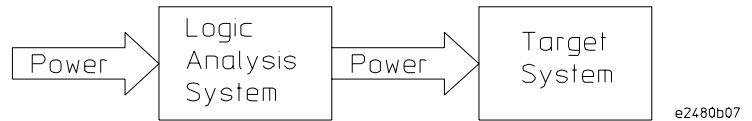
Ensure the target system is powered off.

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

To power on all other logic analyzers

With all components connected, power on your system in the following order:

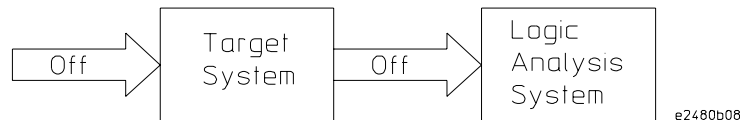
- 1 Logic analysis system.
- 2 Your target system.



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 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 the Agilent Technologies 16600/700-series logic analysis system's *Installation Guide* for instructions on installing modules.

Installing an emulation module

If you ordered an emulation module as part of your logic analysis system, it is already installed in the mainframe.

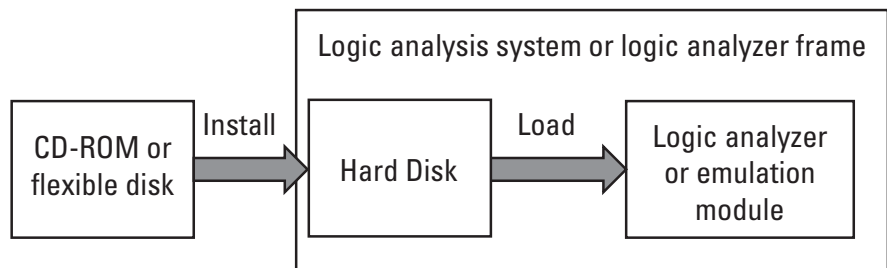
If you ordered an emulation module separately, use the information provided in the *Emulation for the PowerPC 7XX User's Guide* to install your emulation module.

Installing Software

This section explains how to install the software you will need for your inverse assembler or emulation solution.

Installing and loading

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 (16600/700-series logic analysis systems)

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 Agilent Technologies B4620B Source Correlation Tool Set is installed with the logic analysis system's operating system.

To install software from CD-ROM (16600/700-series logic analysis systems)

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 16600/700-series logic analysis system's 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 the "**PPC7XX**" 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**.

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/ppc7xx`. 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.

Installing Software

Probing the Target System

Connecting the Logic Analyzer to the Target System

Each table on the following pages corresponds to a particular logic analyzer. The tables contain connection diagrams for that logic analyzer. If you have used the recommended signal routing for the headers, these tables will apply to your target system. You can also use the Setup Assistant to guide you through the connection process. See page 20.

CAUTION:

Be sure to power down the target system before connecting or disconnecting cables. Otherwise, you may damage circuitry in the analyzer or target system.

There are three types of analysis available:

64-bit data

This type of analysis captures all of the data signals through the data bus. Use the appropriate page, listed below, for your logic analyzer. Connectors J1 through J4 are required for inverse assembly. Connector J5 contains additional signals you might want to monitor.

- 16750/51/52 logic analyzers - two cards (see page 58)
- 16750/51/52 logic analyzers - three cards (see page 59)
- 16715/16/17/18/19A logic analyzers - two cards (see page 58)
- 16715/16/17/18/19A logic analyzers - three cards (see page 59)
- 16710/11/12A logic analyzers - two cards (see page 61)
- 16601A logic analyzers (see page 64)
- 16600A logic analyzers (see page 65)
- 16554/55/56/57 logic analyzers - two cards (see page 67)
- 16554/55/56/57 logic analyzers - three cards (see page 68)
- 16550A logic analyzers- two cards (see page 70)
- 1670A/D logic analyzers (see page 71)
- 1660A/C logic analyzers (see page 72)

32-bit data

This type of analysis will allow you to trace the 32 bits of DATA only, not DATA_B. Use the appropriate page, listed below, for your logic analyzer. The configuration file names are included with the connection diagrams. Connectors J1 and J2 are required for tracing program flow. Connectors J3 and J4 allow you to trace DATA.

- 16550A logic analyzers - one card (see page 69)

No data

This type of analysis allows you to trace program flow only using J1 and J2. Data values in the inverse assembly listing will be taken from the S-Record file, not from the data bus. Use the appropriate page listed below for your logic analyzer. The configuration file names are included with the connection diagrams.

- 16750/51/52 logic analyzers - one card (see page 57)
- 16715/16/17/18/19A logic analyzers - one card (see page 57)
- 16710/11/12A logic analyzers - one card (see page 60)
- 16554/55/56/57 logic analyzers - one card (see page 66)
- 16550A logic analyzers - one card (see page 69)

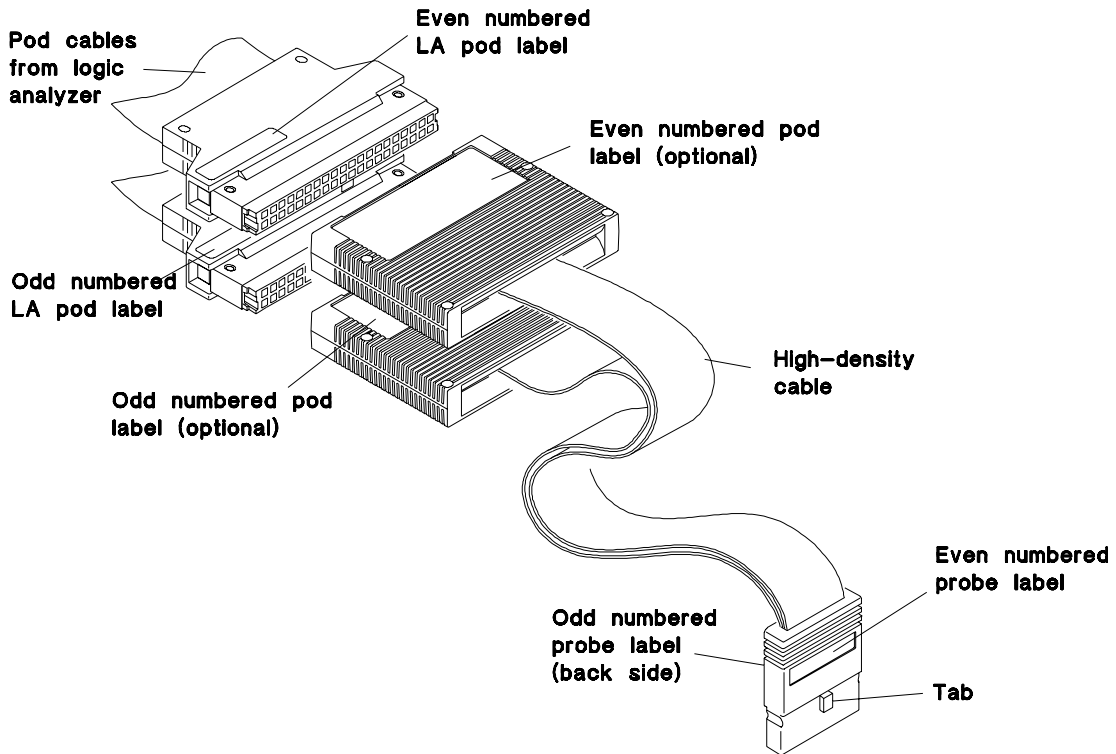
NOTE:

To connect logic analyzers not listed here use the Setup Assistant, as described on page 20.

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

The 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.

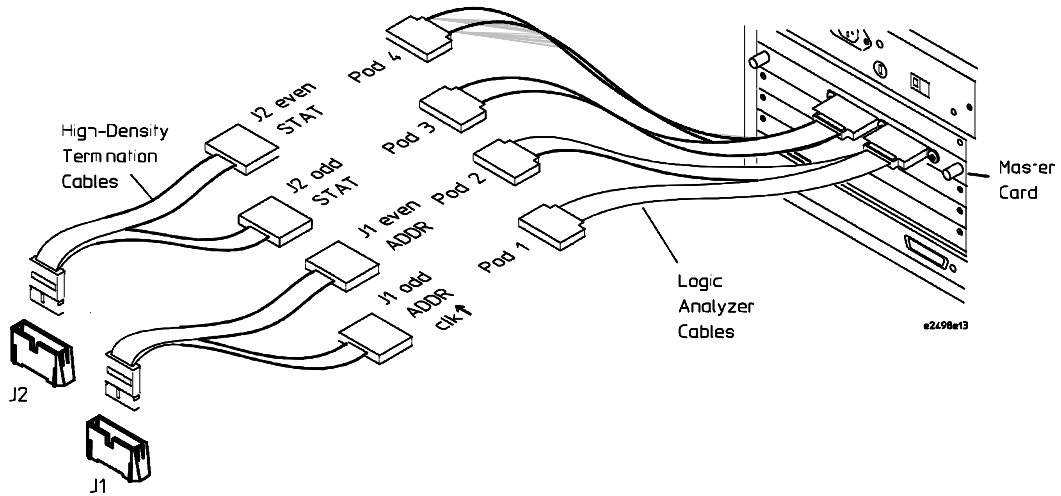
Agilent Technologies E5346A Cable Numbering



e2498e08

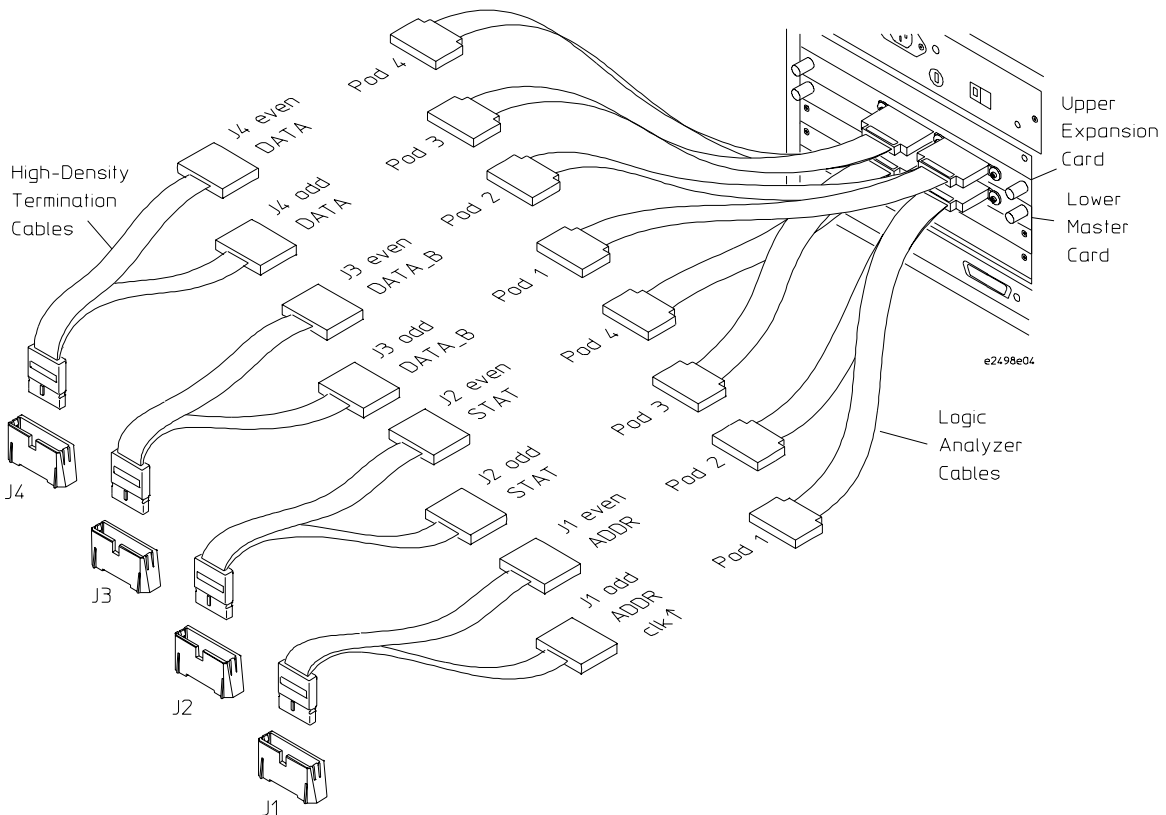
To connect to the 16715/16/17/18/19A and 16750/51/52 logic analyzers (one card)

Use the figure below to connect the target system to the Agilent Technologies 16715/16/17/18/19A and 16750/51/52 logic analyzers. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



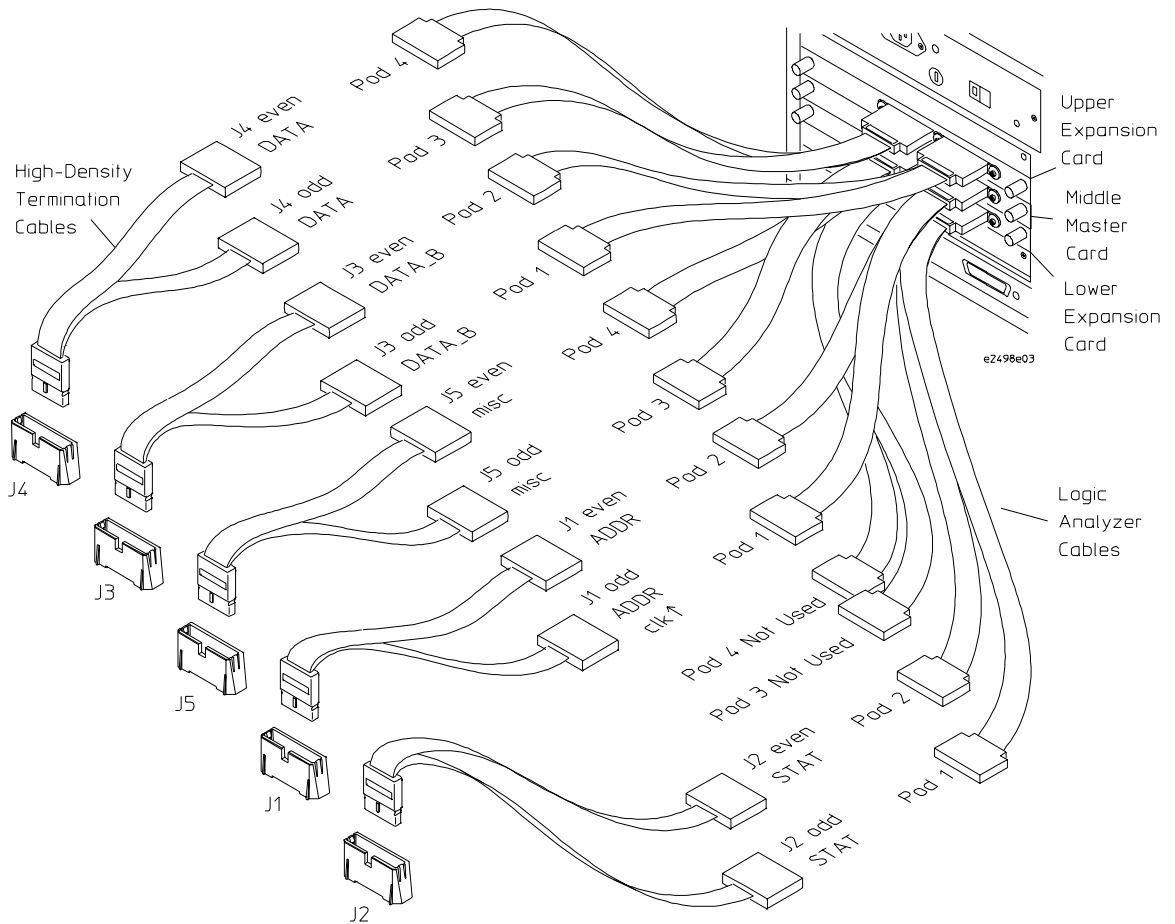
To connect to the 16715/16/17/18/19A and 16750/51/52 logic analyzers (two cards)

Use the figure below to connect the target system to the Agilent Technologies 16715/16/17/18/19A and 16750/51/52 logic analyzers. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



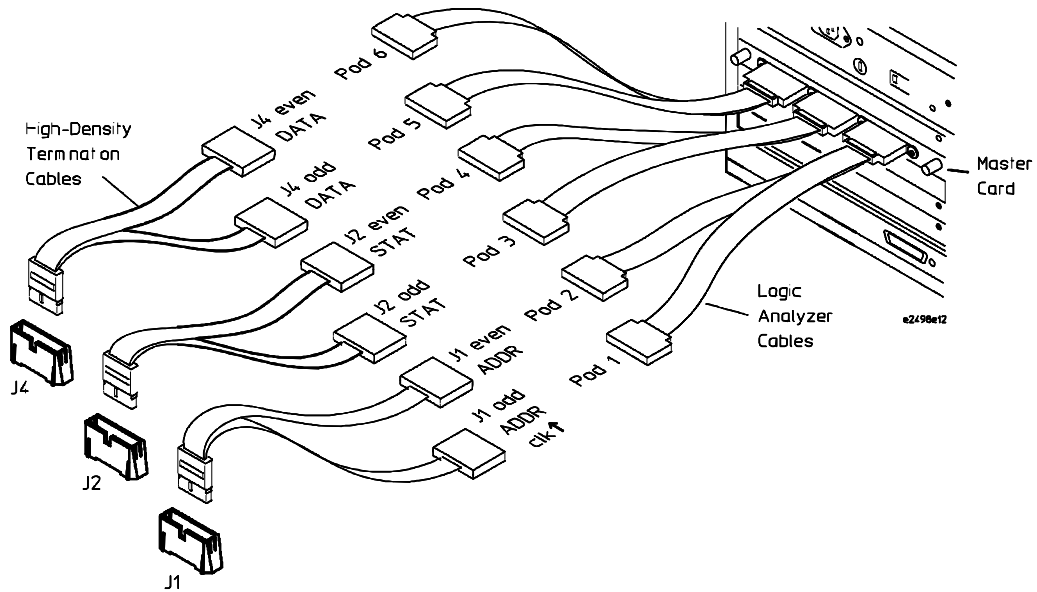
To connect to the 16715/16/17/18/19A and 16750/51/52 logic analyzers (three cards)

Use the figure below to connect the target system to the Agilent Technologies 16715/16/17/18/19A and 16750/51/52 logic analyzers. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



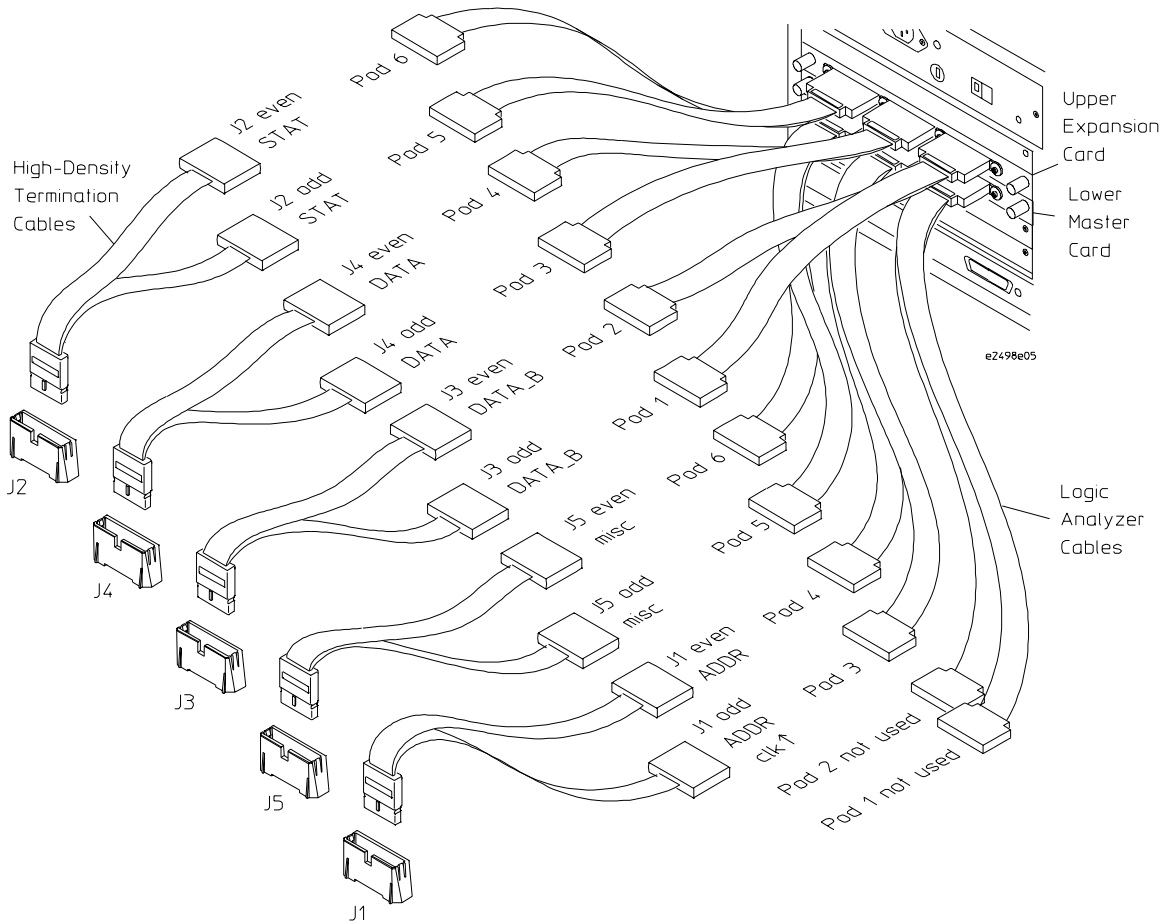
To connect to the 16710/11/12A logic analyzer (one card)

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



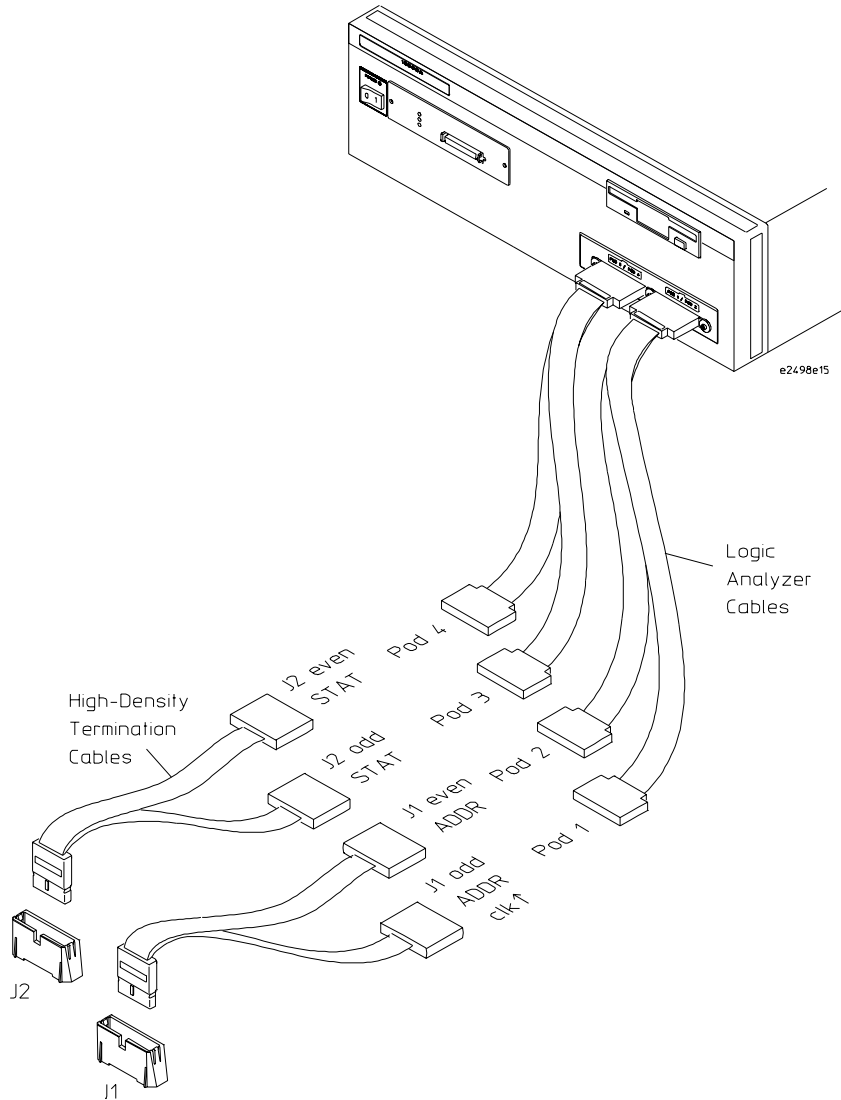
To connect to the 16710/11/12A logic analyzer (two cards)

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



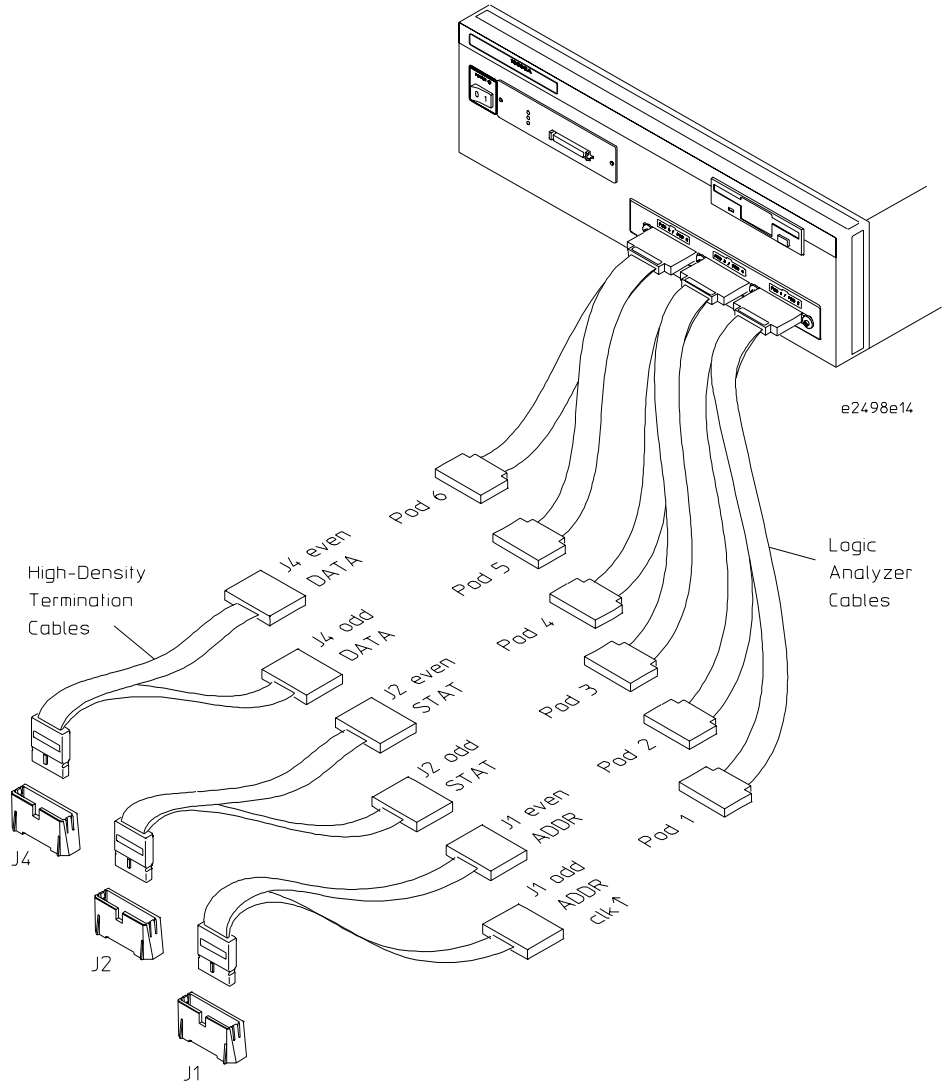
To connect to the 16603A logic analyzer

Use the figure below to connect the target system to the Agilent Technologies 16600A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



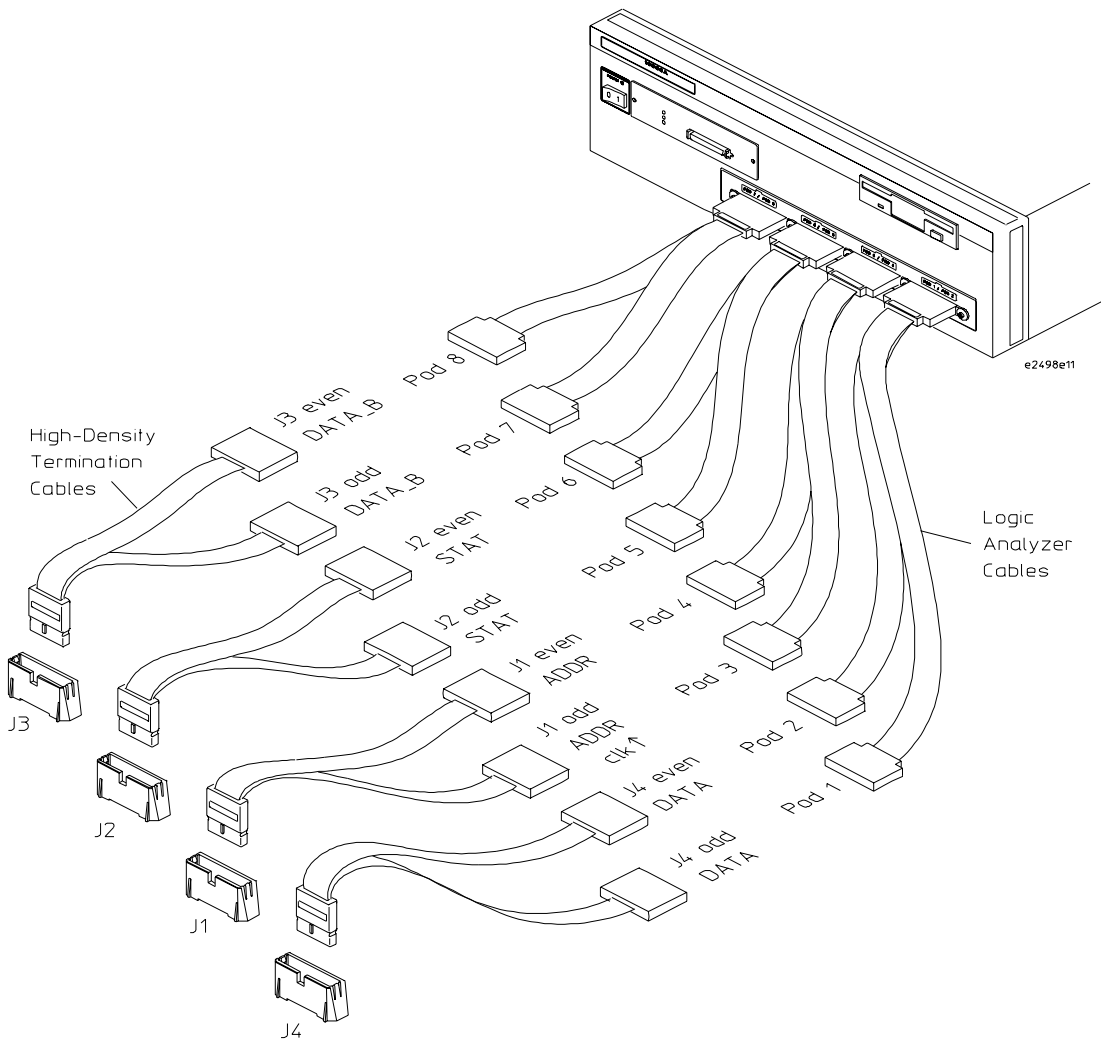
To connect to the 16602A logic analyzer

Use the figure below to connect the target system to the Agilent Technologies 16600A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



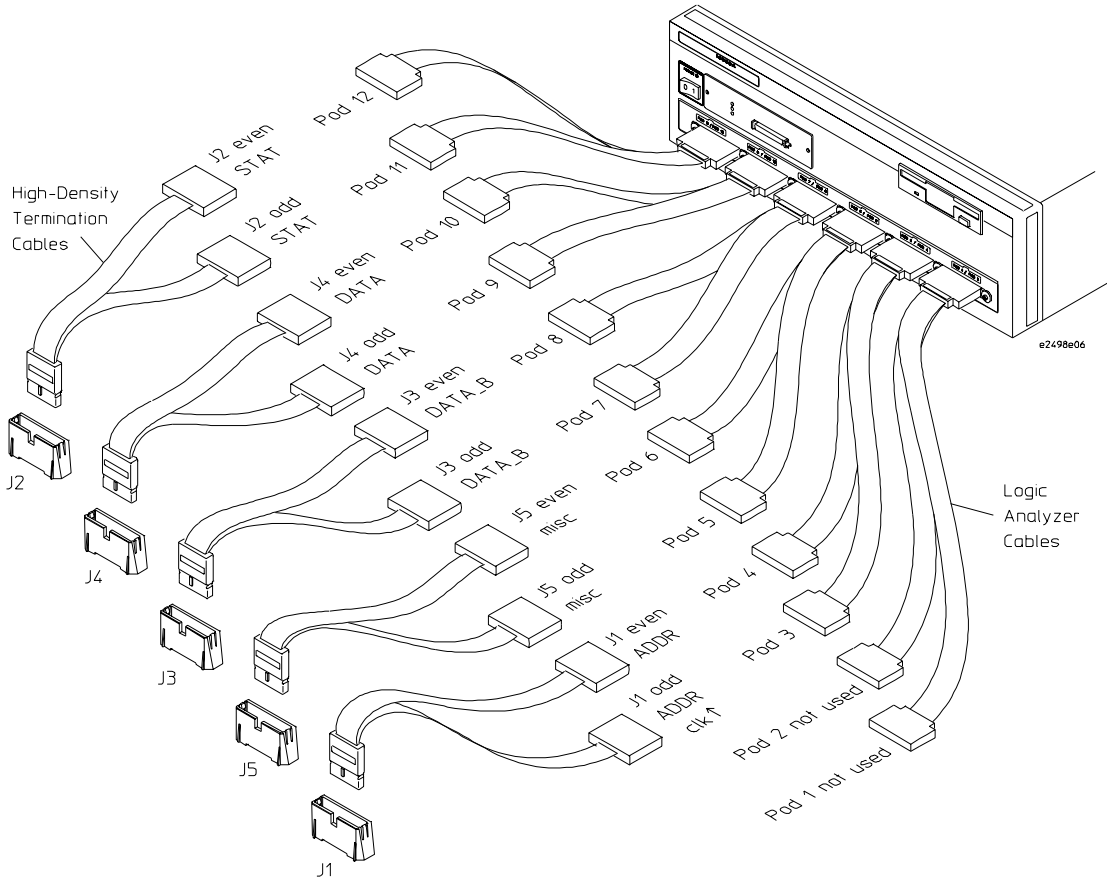
To connect to the 16601A logic analyzer

Use the figure below to connect the analysis probe to the Agilent Technologies 16601A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



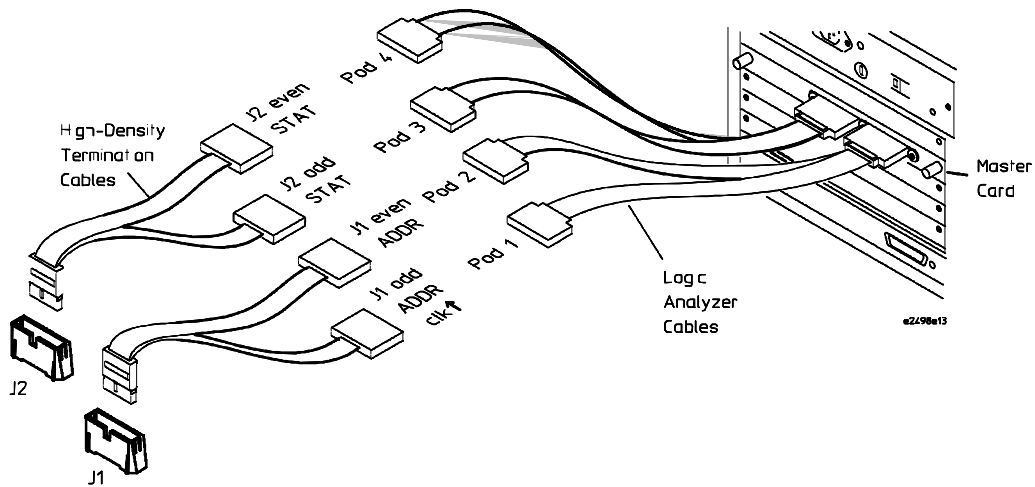
To connect to the 16600A logic analyzer

Use the figure below to connect the target system to the Agilent Technologies 16600A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



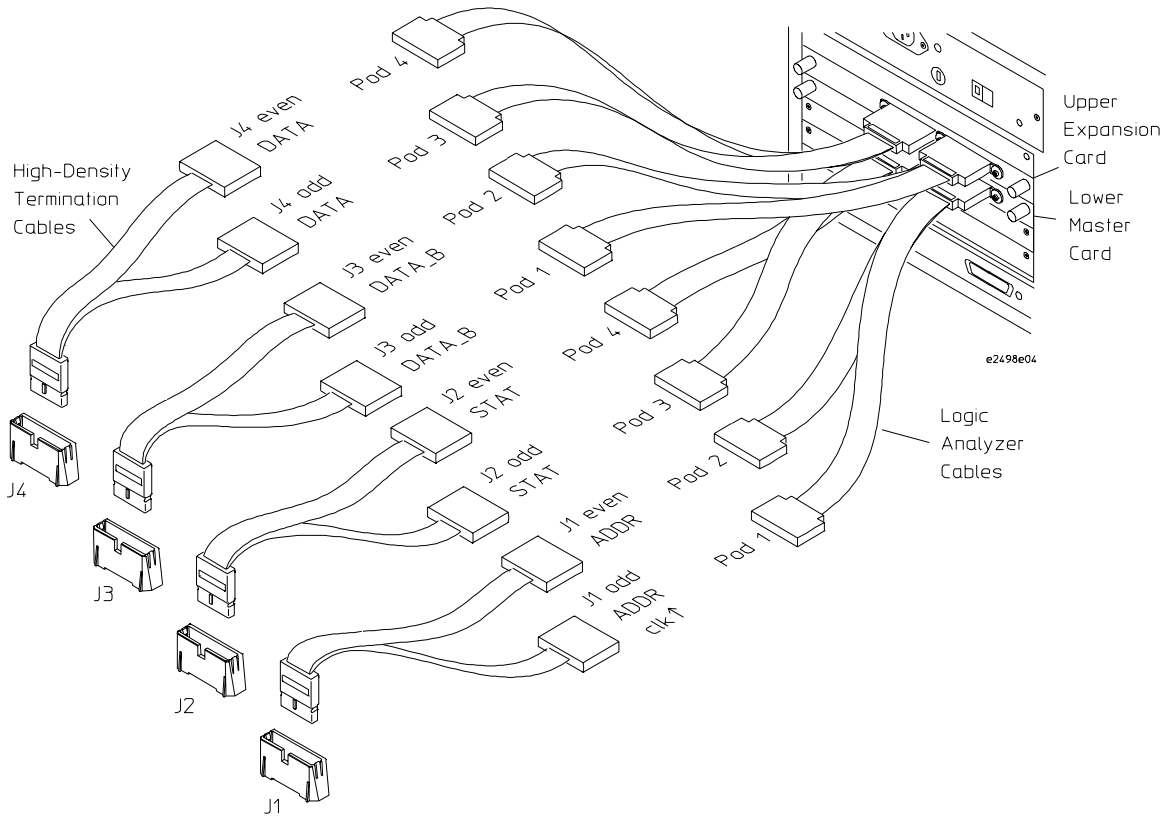
To connect to the 16554/55/56/57 (one-card)

Use the figure below to connect the target system to the two-card Agilent Technologies 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.



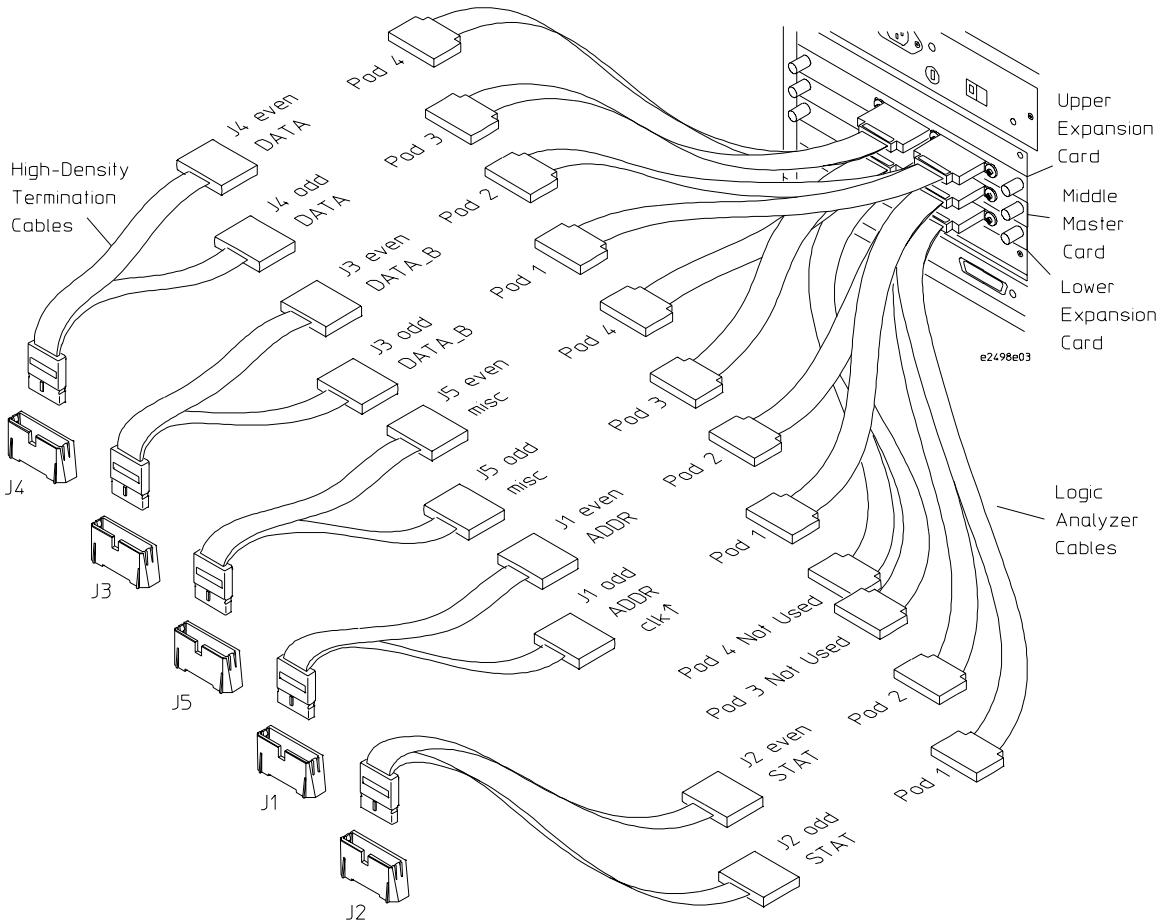
To connect to the 16554/55/56/57 (two-cards)

Use the figure below to connect the target system to the two-card Agilent Technologies 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.



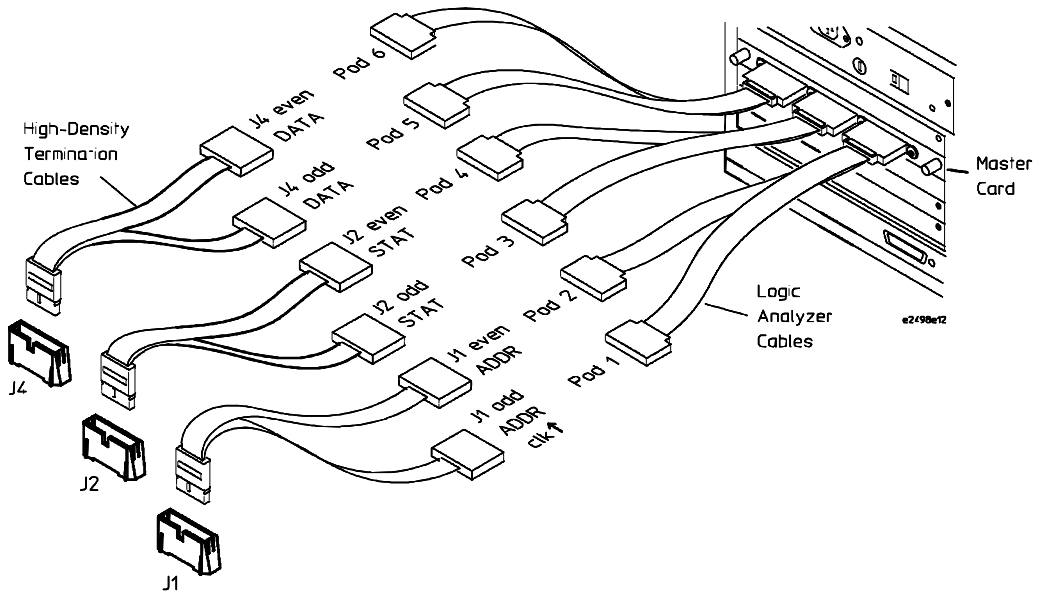
To connect to the 16554/55/56/57 (three-cards)

Use the figure below to connect the target system to the Agilent Technologies 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.



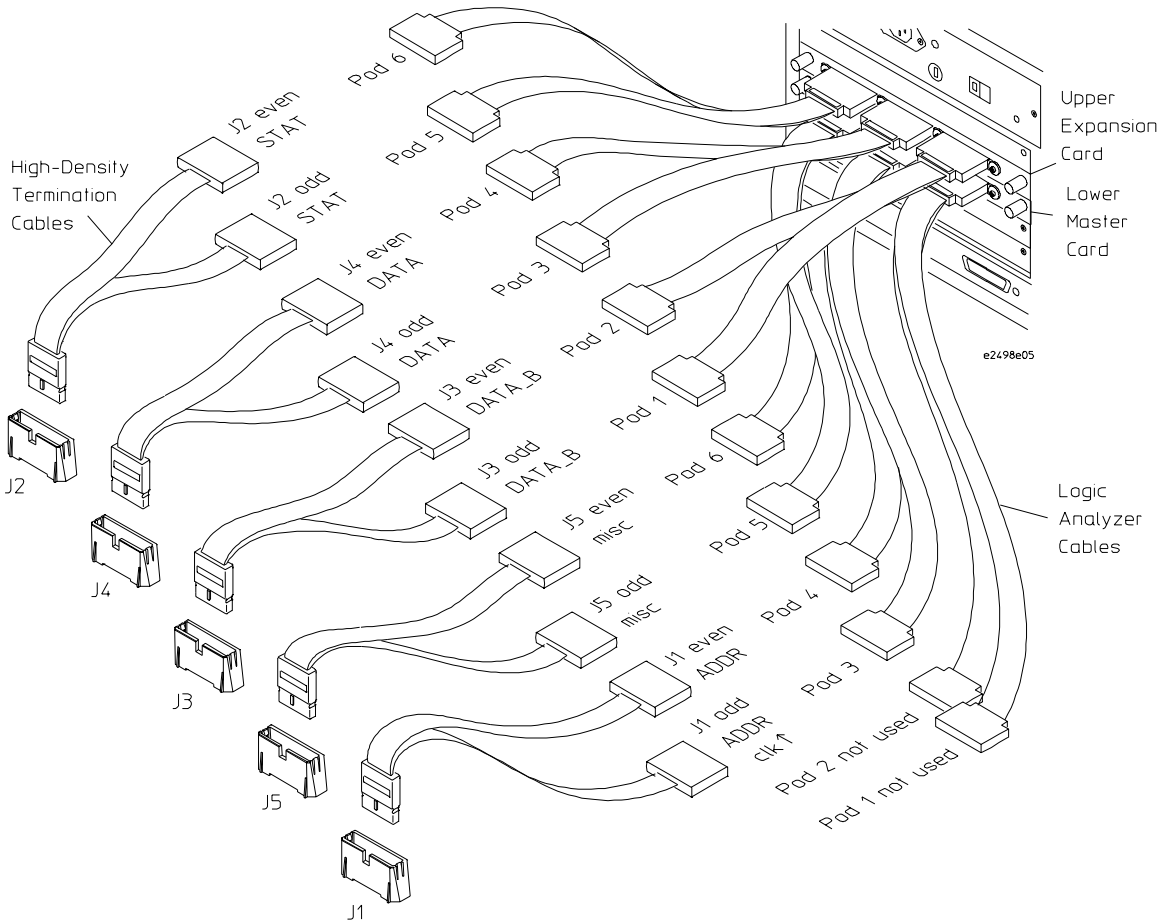
To connect to the 16550A analyzer (one card)

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



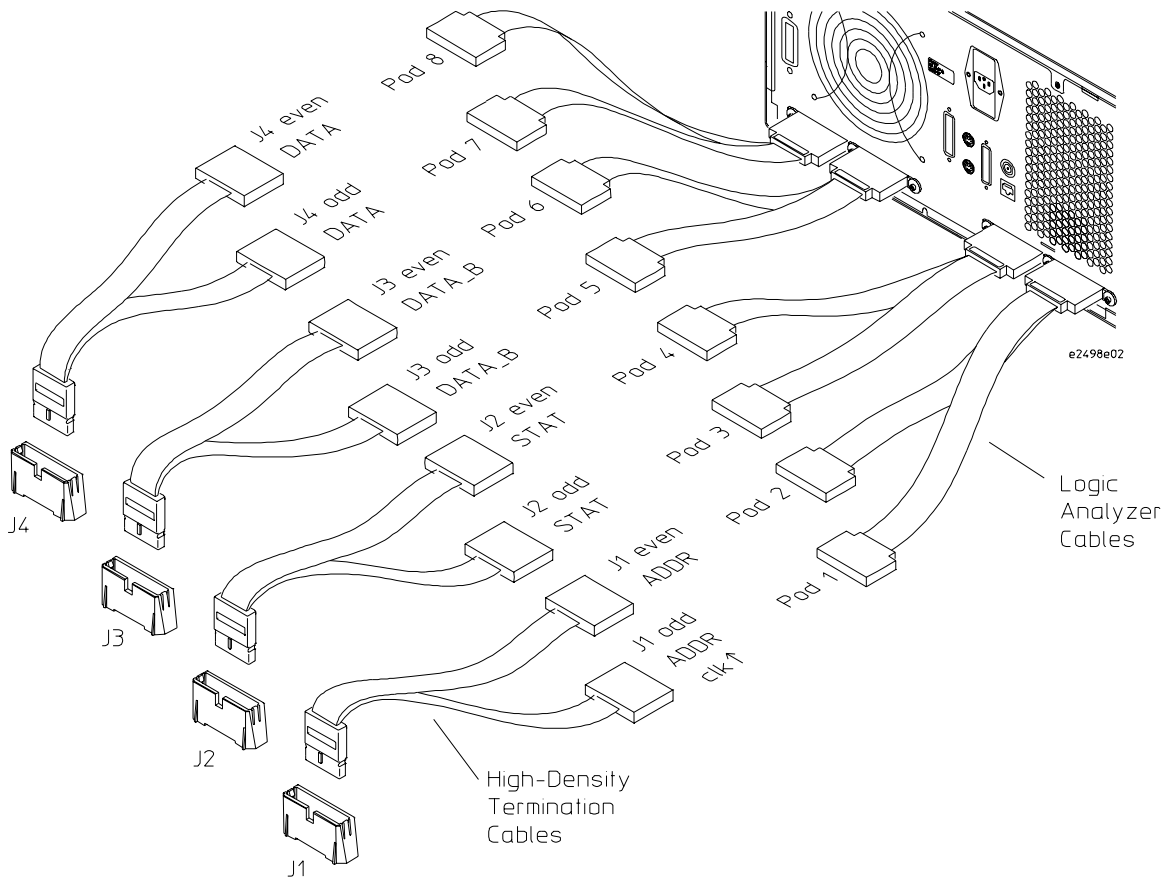
To connect to the 16550A analyzer (two cards)

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



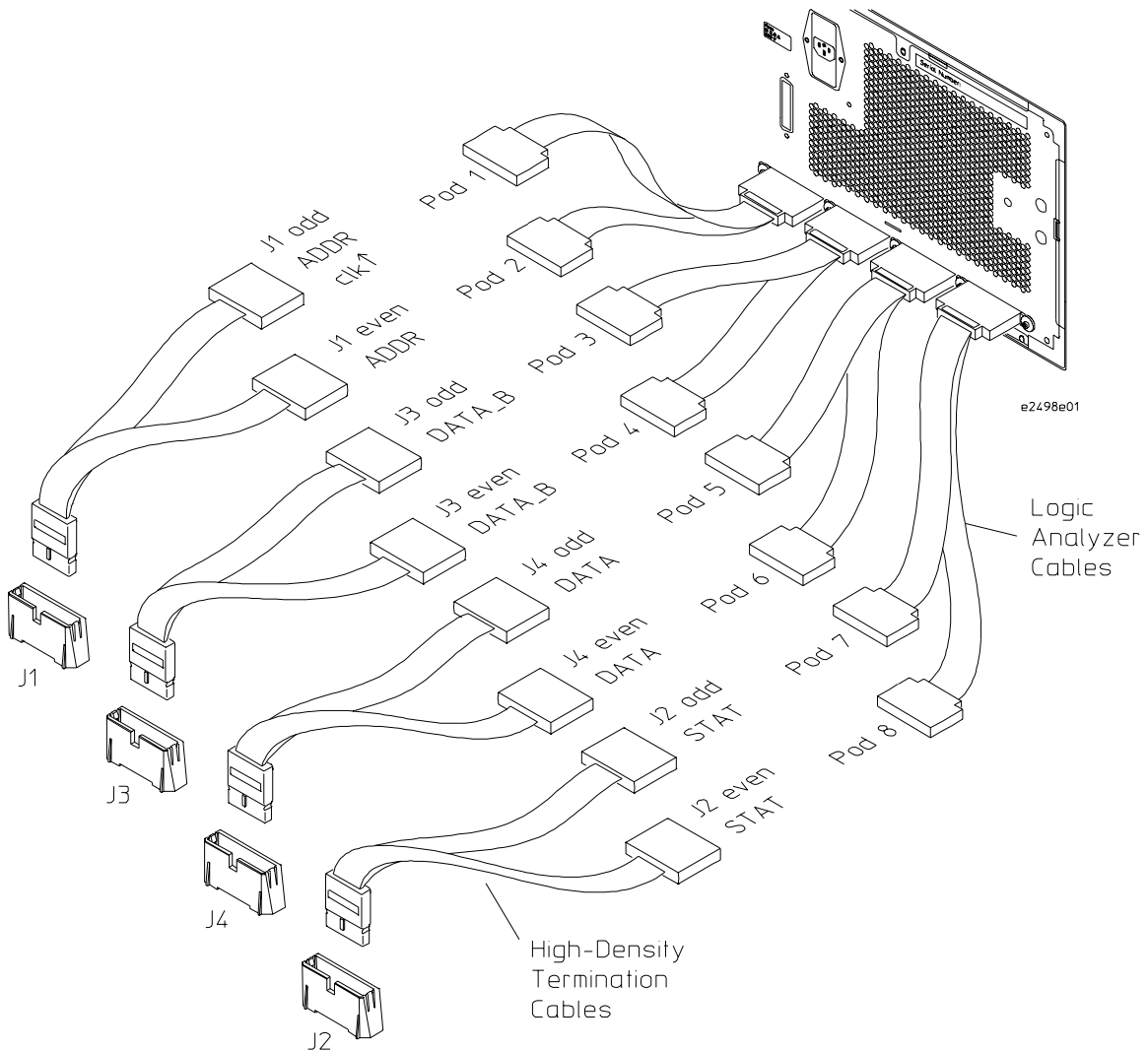
To connect to the 1670A/D logic analyzer

Use the figure below to connect the target system to the Agilent Technologies 1670A/D 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 1660A/C logic analyzers

Use the figure below to connect the target system to the Agilent Technologies 1660A/AS/C/CP/CS logic analyzers. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



**Configuring the 16600/700-Series
Logic Analyzer**

The sections of this chapter describe setting up and using the PowerPC 7XX inverse assembler using 16600/700-series logic analysis systems. If you are using 1660/1670/16500B/C-series logic analyzers, see Chapter 6, “Configuring the 1660/1670/16500B/C-Series Logic Analyzer,” beginning on page 123.

The information in this chapter is presented in the following sections:

- Loading the configuration file and the inverse assembler
- Tables showing configuration file names
- Inverse assembler modes of operation
- Inverse assembler modes of analysis
- Changing the acquisition mode
- Using the Invasm menu
- Setting inverse assembler preferences
- Symbols
- Labels
- Disabling the instruction cache for traditional inverse assembly

Configuring 16600/700-series Logic Analysis Systems

You configure the logic analyzer by loading a configuration file. Normally this is done using the Setup Assistant (see page 20). 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
- Inverse assembler file name

The configuration file you use is determined by the logic analyzer you are using and the type of analysis (64-bit data, 32-bit data, or no-data).

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

If you use Setup Assistant, it will load configuration files and the inverse assembler for you. This is the preferred method. If you did not use Setup Assistant, you can load the configuration and inverse assembler files from the logic analysis system hard disk.

If you choose to use Setup Assistant instead, it will load the configuration file for you. See page 20.

- 1 Select the File Manager icon. Use File Manager to ensure that the subdirectory `/logic/configs/hp/ppc7xx/` exists.

If the above directory does not exist, you need to install the PPC7XX Processor Support Package. Close File Manager, then use the procedure on the CD-ROM jacket to install the PPC7XX Processor Support Package before you continue. See "Installing Software" on page 49 for details.

- 2 Using File Manager, select the configuration file you want to load in the `/logic/configs/hp/ppc7xx/` 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 PPC7XX analysis by loading the appropriate PPC7XX configuration file. Loading the indicated file also automatically loads the correct inverse assembler.

The configuration file names are shown in the table on page 79.

- 3 Close File Manager.

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

If you use Setup Assistant, it will load configuration files and the inverse assembler for you. This is the preferred method. If you did not use Setup Assistant, you can load the configuration and inverse assembler files from the logic analysis system hard disk or floppy disk; however, the preferred method is to install this functionality from the CD-ROM onto the hard disk and load from the hard disk.

To install a configuration and inverse assembler file from a floppy disk:

- 1 Insert the floppy disk in the floppy drive on the 16600/700-series logic analysis system mainframe.
- 2 In the logic analysis System window, select the File Manager icon.
- 3 In the File Manager window:
 - Set Current Disk to Flexible Disk.
 - Set Target to the analyzer you wish to configure.
 - Select the name of the desired configuration file in the Contents frame. The Contents frame lists the configuration files and inverse assembler files available on the floppy disk. These may be either DOS or LIF format files. Either format can be loaded directly into the appropriate logic analyzers.

Note that the logic analyzers read both DOS and LIF formats. However, only DOS formatted floppy disks can be used to store configurations and data. LIF format floppy disks are read-only.

- 4 Select load.

The configuration file you choose will set up the logic analyzer and associated tools. You may see Information, Error, and Warning dialogs that say your configuration has been loaded, and advise you about making proper connections.
- 5 Select the Workspace window icon to see the arrangement of analysis tools in your configuration.
- 6 Select the logic analyzer icon in your configuration and choose its Setup button to see the way your configuration file defined the Sampling, Format, and Trigger options.

NOTE:

Under the Format tab, buses are labeled, and bits included in each bus are identified by an asterisk "*".

This procedure restores the configuration that was in effect when the configuration file was saved. Because the file was not saved using your system, you may receive error messages about loading the enhanced inverse assembler or about pods that were truncated. Select the Sampling, Format, and Trigger tabs and modify the configuration to satisfy your measurement desires. Then you can save your customized configuration to DOS format using the File→Save Configuration selection in any of your tool windows, or selecting the Save tab in the File Manager. For details about how to save configuration files, open the Help window.

To list software packages that are installed

- In the System Administration Tools window, select List....

Logic analyzer configuration files (16600/700-series)

The following table shows the configuration files for the supported logic analyzers.

Logic Analyzer Configuration Files

Analyzer Model	Analyzer Module Description *	Configuration File V_{I/O} = 3.3V	Configuration File V_{I/O} = 1.8V
16750/51/52 (one card)	400 MHz STATE 2 GHz TIMING ZOOM	C7XXL0	C7XX_AVCL0
16750/51/52 (two card)	400 MHz STATE 2 GHz TIMING ZOOM	C7XXL2	C7XX_AVCL2
16750/51/52A (three card)	400 MHz STATE 2 GHz TIMING ZOOM	C7XXL3	C7XX_AVCL3
16717/18/19A (one card)	333 MHz STATE 2 GHz TIMING ZOOM	C7XXL0	C7XX_AVCL0
16716A (one card)	167 MHz STATE 2 GHz TIMING ZOOM		
16715A (one card)	167 MHz STATE 667 MHz TIMING		
16717/18/19A (two cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7XXL2	C7XX_AVCL2
16716A (two cards)	167 MHz STATE 2 GHz TIMING ZOOM		
16715A (two cards)	167 MHz STATE 667 MHz TIMING		
16717/18/19A (three cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7XXL3	C7XX_AVCL3
16716A (three cards)	167 MHz STATE 2 GHz TIMING ZOOM		
16715A (three cards)	167 MHz STATE 667 MHz TIMING		
16710/11/12A (one card)	100 MHz STATE 500 MHz TIMING	C7XXF1	C7XX_AVCF1
16710/11/12A (two card)	100 MHz STATE 500 MHz TIMING	C7XXF2	C7XX_AVCF2
16603A	100/250 MHz LA	C7XXF0	C7XX_AVCF0
16602A	100/250 MHz LA	C7XXF1	C7XX_AVCF1
16601A	100/250 MHz LA	C7XXF2	C7XX_AVCF2
16600A	100/250 MHz LA	C7XXF2	C7XX_AVCF2
16557D (one card)	140/500 MHz LA	C7XXM0	C7XX_AVCM0
16557D (two card)	140/500 MHz LA	C7XXM2	C7XX_AVCM2

Chapter 5: Configuring the 16600/700-Series Logic Analyzer
Configuring 16600/700-series Logic Analysis Systems

Analyzer Model	Analyzer Module Description *	Configuration File $V_{I/O} = 3.3V$	Configuration File $V_{I/O} = 1.8V$
16557D (three card)	140/500 MHz LA	C7XXM3	C7XX_AVCM3
16556A/D (one card)	100/400 MHz LA	C7XXM0	C7XX_AVCM0
16556A/D (two card)	100/400 MHz LA	C7XXM2	C7XX_AVCM2
16556A/D (three card)	100/400 MHz LA	C7XXM3	C7XX_AVCM3
16555A/D (one card)	110/500 MHz LA	C7XXM0	C7XX_AVCM0
16555A/D (two card)	110/500 MHz LA	C7XXM2	C7XX_AVCM2
16555A/D (three card)	110/500 MHz LA	C7XXM3	C7XX_AVCM3
16554A (one card)	70/250 MHz LA	C7XXM0	C7XX_AVCM0
16554A (two card)	70/250 MHz LA	C7XXM2	C7XX_AVCM2
16554A (three card)	70/250 MHz LA	C7XXM3	C7XX_AVCM3
16550A (one card)	100 MHz STATE 500 MHz TIMING	C7XXF1	C7XX_AVCF1
16550A (two card)	100 MHz STATE 500 MHz TIMING	C7XXF2	C7XX_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 20 for instructions for using the Setup Assistant.

NOTE:

The configuration files for $V_{I/O} = 1.8V$ are for the PowerPC 745/755 processors only. The threshold voltage for these configuration files is set to 900mV instead of TTL levels.

Logic Analyzer Configuration

The following sections describe the logic analyzer configuration as set up by the configuration files.

It is strongly recommended that you do not change the setup related to the PPC7XX sampling, format, pod assignment or configuration dialogs. The configuration file (loaded by the Setup Assistant in 16600/700-series logic analysis systems) will configure the logic analyzer for making measurements.

Format menu

This section describes the organization of PPC7XX 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. The tables on the following pages show the signals used in the STAT label and the predefined symbols set up by the configuration files.

The Agilent Technologies 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.

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

The configuration software sets up the analyzer format dialog to display either eight or ten pods of data, depending on the analyzer.

Status Encoding

Each of the bits of the STAT label is described in the table below. Most of the status and control signals on the PowerPC 7XX are active low ("- suffix). To conserve display space, the "-" is omitted in many of the Format definitions.

The inverse assembler uses STAT bits TC0, TSIZE...2, TT0...3, TBST, TA, AACK, ARTRY, DRTRY, ABB, TEA, and TS. The signal-to-connector tables in the "Hardware Reference" chapter list all the PPC7XX signals probed and their corresponding analyzer channels.

Status Bit Description

Status Bit	Description
BR-	The PowerPC 7XX asserts Bus Request to indicate that it has business to conduct on the address bus
BG-	The memory system asserts Bus Grant to allow the 7XX onto the address bus
ABB-	Address Bus Busy indicates that the address bus is in use
TS-	The PowerPC 7XX asserts TS- for one cycle to commence a transaction. It also serves as the data bus request signal if the TT signals indicate a data transfer.
XATS-	XATS commences a "programmed i/o" (PIO) sequence in the extended address transfer protocol (not available on 7XXe).
DBG-	The memory system asserts Data Bus Grant to allow the 7XX onto the data bus.
DBWO-	The memory system may assert Data Bus Write Only to allow the 7XX to envelope a data write (snoop push, typically) between the address and data phases of a data read.
DBB-	Indicates Data Bus Busy.
AACK-	The memory system asserts AACK for one cycle to acknowledge an address.
ARTRY-	The memory system may assert ARTRY to cause the 7XX to back off the bus and retry the transaction.
TA-	The memory system asserts TA to acknowledge a data transaction.

Status Bit	Description
DRTRY-	The memory system may assert DRTRY to cancel the effect of a TA in the previous cycle.
TEA-	The memory system may assert TEA to indicate a transfer error, e.g. an unmapped part of the address space.
TT 0:4	The Transfer Type signals indicate the direction and purpose of a bus transaction.
Atomic(TT0)	Usually, TT0 asserted indicates an atomic (e.g., stwcx.) transfer.
R/-W (TT1)	TT1 is high for a read, low for a write.
InvlDt (TT2)	Usually, the PowerPC 7XX asserts TT2 to indicate that the corresponding cache line should be invalidated by other processors.
A Only (TT3)	TT3 high indicates that there is data associated with the current address. TT3 low usually indicates an address-only transaction.
TC 0:1	The Transfer Code outputs provide further information about the current transfer. For a read, they indicate whether instructions or operands are being fetched.
TBST-	When asserted, TBST- indicates a four-beat burst transfer of eight words.
TSIZ 0:2	Indicates the size for the data transfer in conjunction with TBST-.
WT-	Write Through indicates a write-through transaction that should be pushed through local caches to shared memory.
CI-	Cache Inhibit indicates that the 7XX will not cache a read.
GBL-	Global indicates the transaction is global, i.e., should be snooped by all other caching devices on the bus.
SRESET-	A falling-edge input causes the 7XX to undergo a soft reset.
HRESET-	Input asserted causes the 7XX to undergo a hard reset.
CKSTP-	Input asserted causes the 7XX to undergo machine check processing.

Status Bit	Description
INT-	Input indicates an external interrupt is pending.
CHECKSTOP	Output indicates that the 7XX has entered the machine check state, i.e., stopped.
QREQ-	The PowerPC 7XX asserts Quiescent Request to indicate that it wants to be, or is, in a snooze mode.

NOTE:

The PPC 745/755 has two signals which the PPC 740/750 does not have. They are BVSEL and L2VSEL. These pins are not routed on the preprocessor and they are not assigned to labels in the configuration file.

Predefined Logic Analyzer Symbols

The configuration software sets up symbol tables on the logic analyzer. The tables define a number of symbols which make several of the STAT fields easier to interpret. The following table lists the symbol descriptions.

Symbol Description

Label	Symbol	Encoding
acks	idle	1111
	ARTRY	xxx0
	DRTRY	0xxx
	TA AACK	x00x
	AACK	xx0x
	TA	x0xx
R/-W	rd	1
	wr	0
TSIZ/TBST- †	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	

Label	Symbol	Encoding
TT	Kill Block	01100
	Wr Graphics	10100
	Rd Graphics	11100
	Clean Block	00000
	Write	00010
	Wr/Kill	00110
	Read	01010
	Rd/Flush	01110
	Wr/Atomic Flush	10010
	Read Atomic	11010
	Rd/Flush Atomic	11110
	Flush Block	00100
	DSYNC	01000
	eieio	10000
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.

Trigger dialog

This section describes some PowerPC 7XX-specific considerations in triggering the analyzer. You can use the Trigger dialog to change the triggering and storage qualification to include or exclude specified cycles. The trigger specification set up by the software stores all states.

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

Qualifying Stored Data

The trigger dialog determines what will be acquired by the analyzer and when it will be acquired. The configuration file pre-configures a storage qualification term to exclude wait and idle states from the analyzer's memory.

The configuration file creates a custom default trigger term called "idle" and changes the default storing to "store if not idle". The custom term "idle" is defined as AACK, ARTRY, TA, and DTRTY all high (not asserted). The trigger sequence stores states that are not idle.

Configuring for State-per-clock mode

To configure the analyzer to store all states including wait and idle states, change the storage qualification to capture all states (state-per-clock).

Go to the trigger dialog and select the **Default Storing** tab. Change the default storing mode from **Custom** to **Anything**.

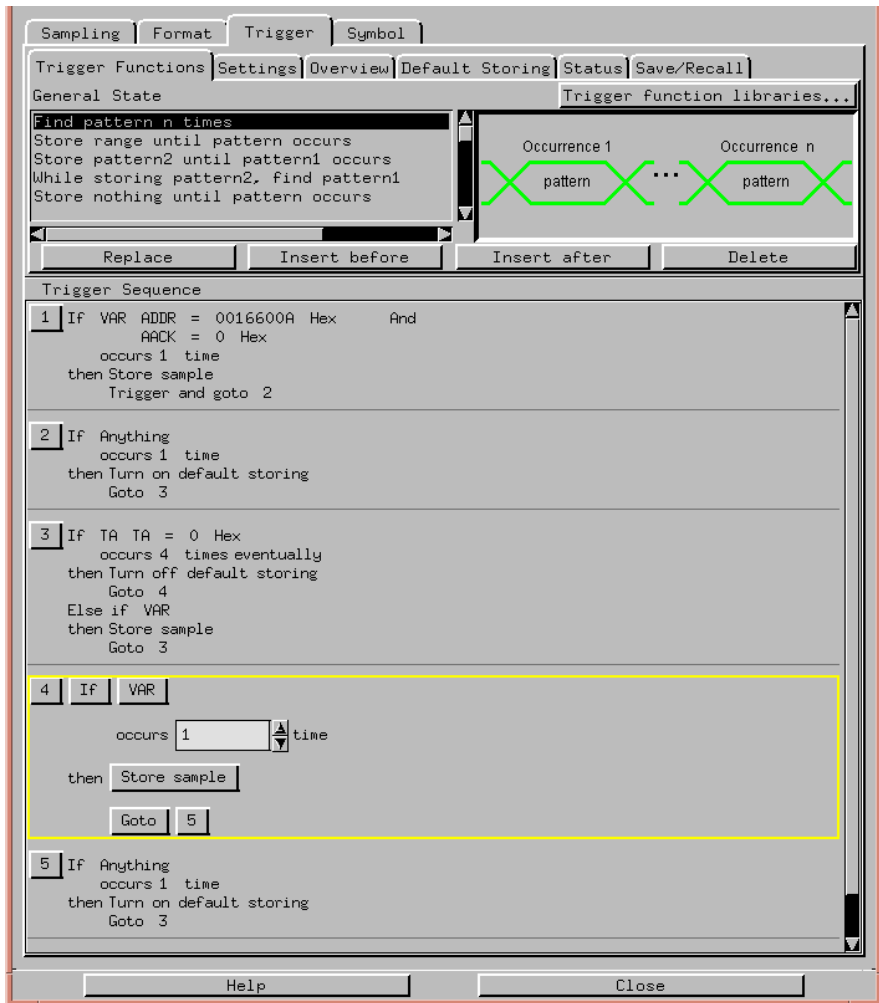
Capturing an address

To accurately trigger on a specific address, create a term with two labels: ADDR and AACK. Enter the address in the ADDR field of a trigger term and enter 0 in the AACK field of the term. This will prevent false triggering on a floating address bus.

The instruction addresses presented on the PowerPC 7XX address bus always end in hex 0 or hex 8. When the instruction cache is enabled, the 7XX will burst four data beats per address and will not update the address as it bursts. To reliably trigger on the fetch of a particular address when bursting, the least significant three bits of the address must be "don't cares." Change the base of the ADDR label to Binary to enter the 3 X's.

Capturing Isolated Addresses

The loose coupling of the address and data buses on the PowerPC 7XX makes it more difficult to trace only activities associated with a given address, such as writes to a variable. Depending on how deep the pipeline is, an address of interest may be followed by up to four data beats before the data associated with the address appears on the bus. One technique to trace writes to a variable is shown below. (The data cache is off or in write-through mode.)



Using the Inverse Assembler

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

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.

Using Cache-On Trace Reconstruction

Cache-on trace reconstruction lets you track instructions executed in the cache. The inverse assembler requires that an S-Record executable file is loaded and that the data bus is connected. This enables the logic analyzer to display the inverse assembled data in mnemonics.

The inverse assembler uses branch trace mode. For cache-on trace reconstruction, set the operating mode to State-per-ack and enable the branch trace mode by setting the MSR.BE bit 22. This BE bit enables a branch trace exception to be taken after a successful completion of a branch instruction.

The branch exception is located at 0x00000D00 for an exception prefix MSR.IP=0 or 0xFFFF00D00 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.

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:

In the **External Bus Decoding** dialog, located in the **Decoding Options** tab:

- 1 Set the cache-on mode
- 2 Set data bus connected
- 3 Provide the tracking address

In the **Opcode Source** tab

- 4 Load an S-Record executable file

To minimize effects of cache-on trace on system performance

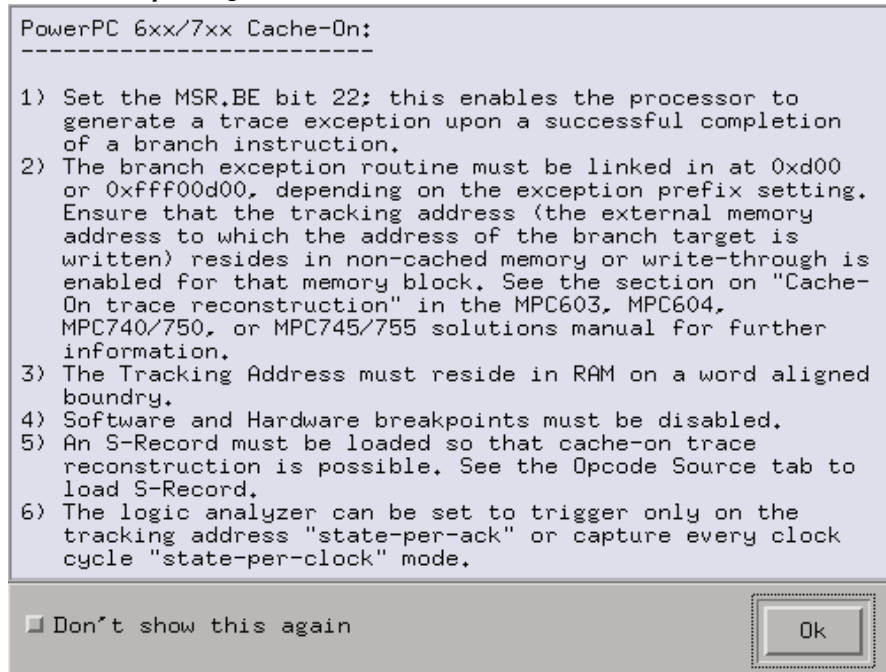
- Enable cache-on trace via the MSR.BE bit only for selected portions of code or specific tasks.
- Minimize the number of instructions in the exception handler.
 - a Dedicate registers for writing out branch messages.
 - b Do not save/restore any system registers.
- Make sure the instruction handler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

Chapter 5: Configuring the 16600/700-Series Logic Analyzer

Using the Inverse Assembler

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.

NOTE:

When using Agilent Technologies run-control the user must issue an "rstssm" (reset soft stop mode) to disable soft stop mode. Also, this command must be issued whenever registers are modified, since soft stop mode is re-enabled automatically.

Enabling branch exception disassembly

The following trace shows cache-on execution using branch trace exception disassembly. See page 88 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, tracking address 0x00000100

The screenshot shows the Inverse Assembler software interface. The main window displays a list of assembly instructions with columns for State Number, SW_ADDR, Mnemonics/Hex, and PowerPC 6xx/7xx Inverse Assembler. The instructions are as follows:

State Number	SW_ADDR	Mnemonics/Hex	PowerPC 6xx/7xx Inverse Assembler
239	ABSOLUTE 00009D90	read word	0x00000001
240	ABSOLUTE 00003B84	lwz	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 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 microprocessor memory controller configuration. The other dialogs assist in analyzing and displaying data. The following sections describe these dialogs.

Loading 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.

Setting the Inverse Assembler Preferences

Why the configuration is necessary

Because critical information about what type of data is being accessed through a memory bank is stored in internal registers, the inverse assembler needs to be given some information about how the memory system is set up.

The memory controller operates by mapping every address to one of eight memory regions. Each memory region can be set up to drive different external signals, to have different write permissions, etc. The memory regions are numbered from 0 to 7. Memory region 0 has the highest priority and region 7 has the lowest.

The base register and option register for each memory region hold information that describes the width of the memory accessed through that region and the addresses that will be accessed through that region. Since this information is not given on external signals, the inverse assembler provides a preferences window to enter this information so that the data decode can be as accurate as possible.

To set the inverse assembler preferences

To open the Preferences dialog:

- 1** Load a configuration file, if needed.
- 2** Open the Listing window.
- 3** Select **Preferences...** from the **Invasm** menu at the top of the Listing window.

To set the memory map preferences

It is necessary to configure the memory map in the Preferences dialog before using the inverse assembler.

Inverse Assembler Preferences Dialog

PowerPC 6xx/7xx (E2498A/E2449B) Preferences
File In<1>;File In<1>;Frame 10;Slot A:PPC745/755

Region Number	Base Address	End Address	Memory Width
Region 0	00000000	FFFFFFF	64 bits
Region 1	00000000	00000000	64 bits
Region 2	00000000	00000000	64 bits
Region 3	00000000	00000000	64 bits
Region 4	00000000	00000000	64 bits
Region 5	00000000	00000000	64 bits
Region 6	00000000	00000000	64 bits
Region 7	00000000	00000000	64 bits

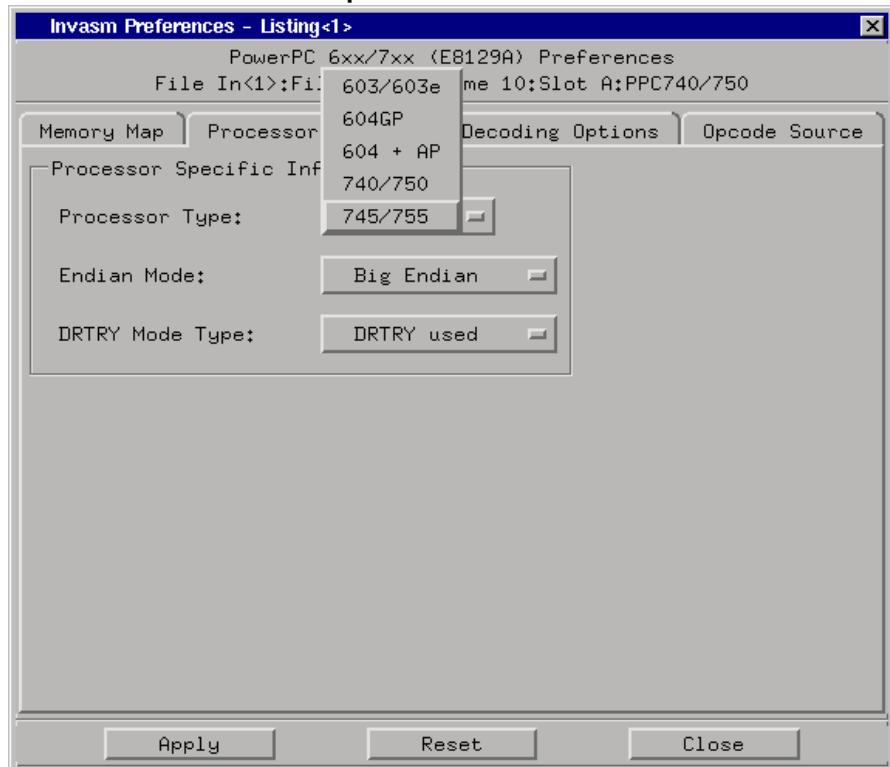
Apply Reset Close

Select Apply when you have finished configuring the memory map.

Setting the Inverse Assembler Preferences**To set the processor options preferences**

Processor Options. This inverse assembler is designed to work with the PowerPC 60X/740/750/745/755 microprocessors. Because of this wide range of support, you must specify which processor type is currently being used. Use the following table to determine how to specify the processor type.

Processor Used	Set Processor Type
PowerPC 740/750	740/750
PowerPC 745/755	745/755

Inverse Assembler Processor Options

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.

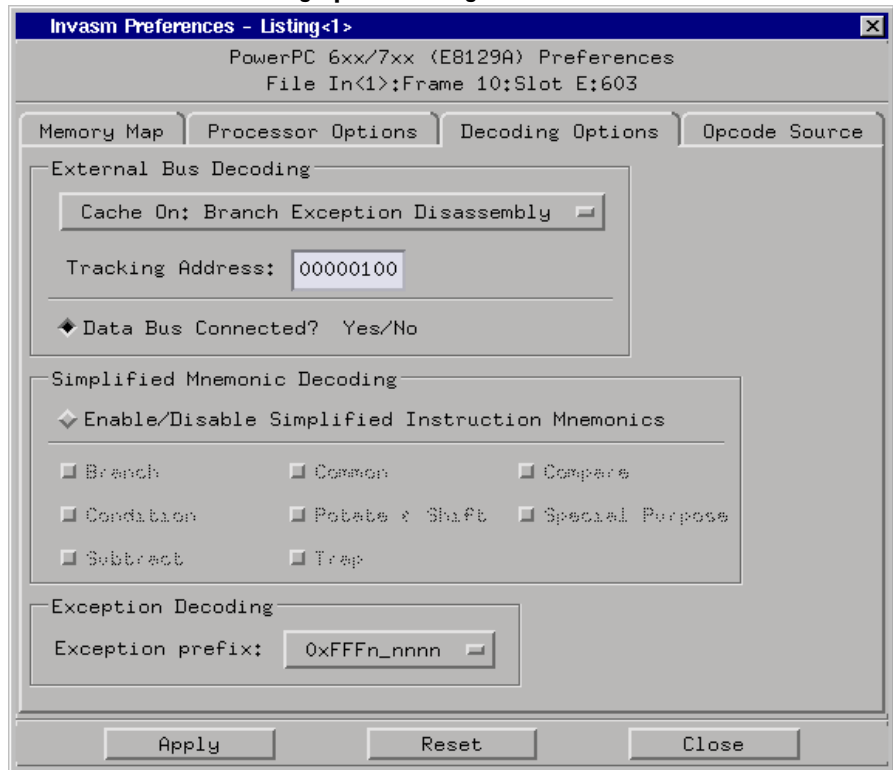
Little-endian mode causes the instruction word from DL0...31 (DATA_B label; external address xxx4) to be dispatched before the instruction word from DH0...31 (DATA label; external address xxx0). It also causes byte and half-word reads and writes to appear on the opposite side of the bus and swaps the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

DRTRY Mode

The inverse assembler also allows you to specify inclusion of the DRTRY signal in its decoding. There are certain versions of the PowerPC microprocessors that have a no-DRTRY mode. If your processor is currently running in this mode, be sure to select the no-DRTRY mode.

To set the decoding options preferences

Inverse Assembler Decoding Options Dialog



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 92

Simplified Mnemonic Decoding. PowerPC assemblers support a number of simplified mnemonics for some popular assembly language instructions, as described in Appendix F of the *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. Select 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.
- 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.

Setting the Inverse Assembler Preferences

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.

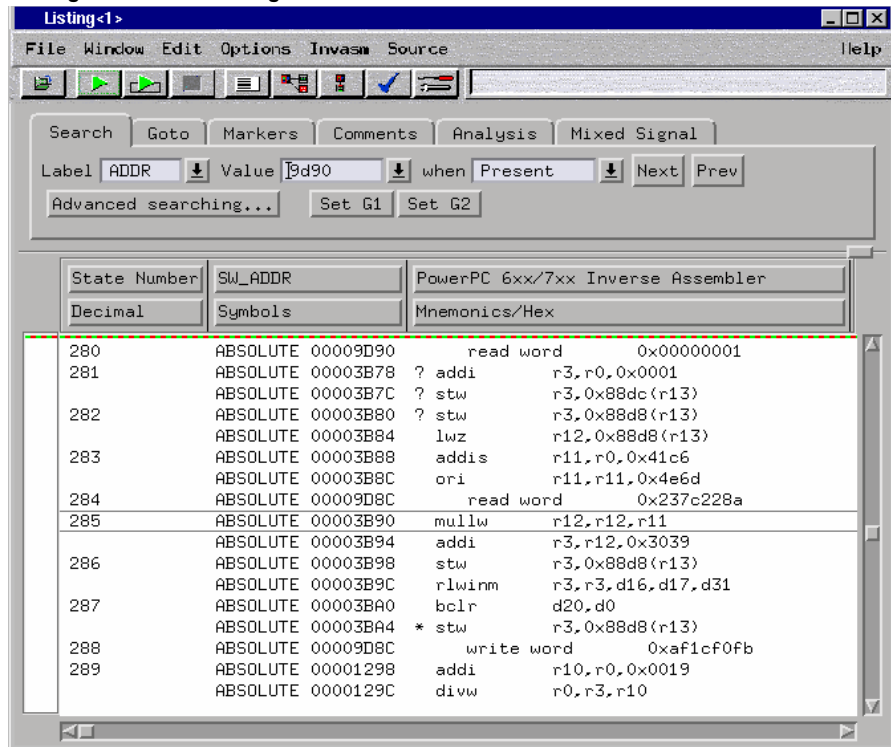
Mnemonic	Decoded As
rlwimi (rotate left word immediate then mask insert)	inslwi insert from left immediate insrwi insert from right immediate
rlwinm (rotate left word immediate then AND with mask)	rotlwirotate left immediate rotrwirotate right immediate slwishift left immediate srwishift right immediate 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

Setting the Inverse Assembler Preferences

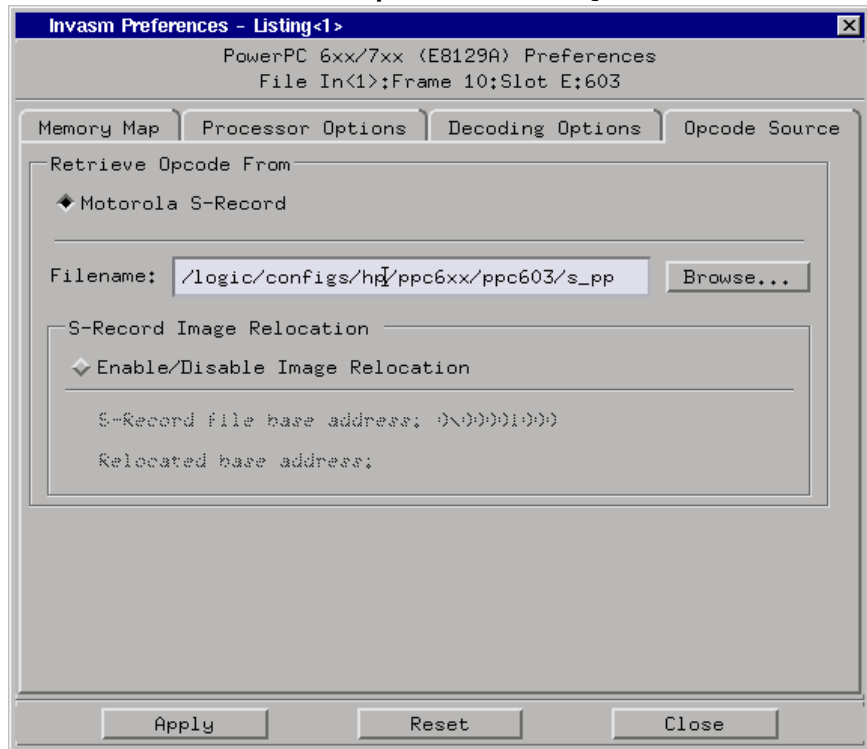
Exception Decoding. the inverse assembler can output the types of exceptions that occur. The PowerPC architecture allows for two locations of the exception vector table. You can determine which location is set up for your target by looking at the MSR.IP bit 25. This can be done by examining the initialization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

Listing Window Showing Trace with Data Bus Connected: Cache Off

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

To set the opcode source preferences

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 enable/disable the instruction cache on the PPC7XX

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 probe:

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:

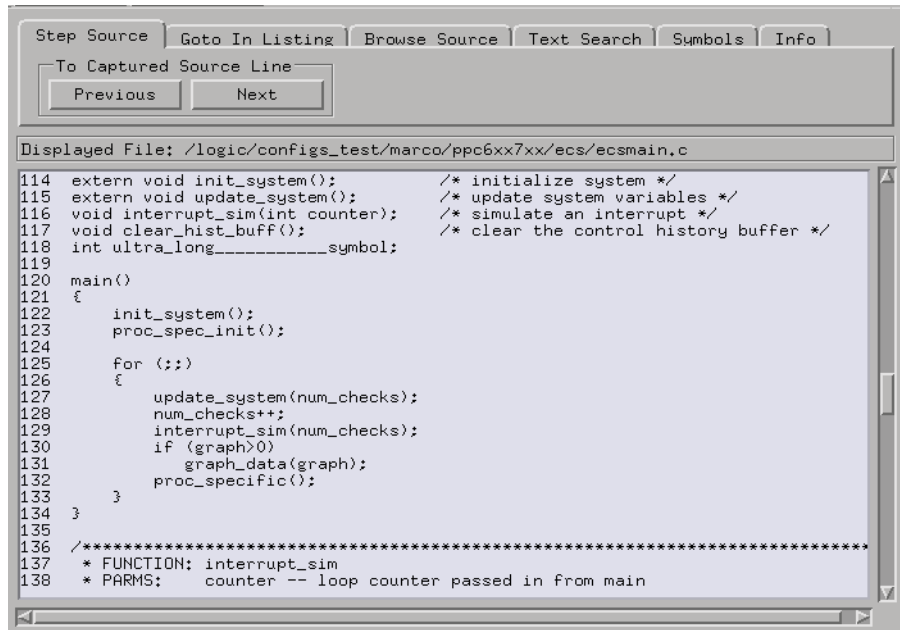
```
mf spr    r3, hid0
ori      r3, 0C00# set ICFI and DCFI
mt spr   hid0, r3
rlwinm  r3, r3, 0, 22, 19 # clear ICFI and DCFI
mt spr   hid0, r3
rlwinm  r3, r3, 0, 18, 15 # clear ICE and DCE
mt spr   hid0, r3
isync
```

- Enable the instruction cache with the following code:

```
mf spr    r3, hid0
rlwinm  r3, r3, 1, 17, 15 # set ICE
mt spr   hid0, r3
isync
```

Setting the Inverse Assembler Preferences**Displaying Data with the B4620B Source Correlation Tool Set**

Source correlation correlates the addresses from cache with the high-level code execution. The figure below shows execution of data that is correlated to the data shown on the previous page.

Source Correlation Tool Set Data


The screenshot shows a window titled "Source Correlation Tool Set Data" with a menu bar containing "Step Source", "Goto In Listing", "Browse Source", "Text Search", "Symbols", and "Info". Below the menu bar is a "To Captured Source Line" section with "Previous" and "Next" buttons. The main area displays the source code for the file "/logic/configs_test/marco/ppc6xx7xx/ecs/ecsmain.c".

```

114 extern void init_system();          /* initialize system */
115 extern void update_system();       /* update system variables */
116 void interrupt_sim(int counter);   /* simulate an interrupt */
117 void clear_hist_buff();           /* clear the control history buffer */
118 int ultra_long_____symbol;
119
120 main()
121 {
122     init_system();
123     proc_spec_init();
124
125     for (;;)
126     {
127         update_system(num_checks);
128         num_checks++;
129         interrupt_sim(num_checks);
130         if (graph>0)
131             graph_data(graph);
132         proc_specific();
133     }
134 }
135
136 /*****
137  * FUNCTION: interrupt_sim
138  * PARMS:   counter -- loop counter passed in from main

```

Symbols are more easily recognized than hexadecimal address values in logic analyzer trace displays, and they are easier to remember when setting up triggers.

You can download symbols from certain object file formats into Agilent Technologies logic analyzers.

Also, Agilent Technologies logic analyzers let you assign user-defined symbol names to particular label values.

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

After describing symbols, the rest of this chapter describes the requirements and considerations for displaying object file symbols and source code for PPC7XX address values captured by a logic analyzer.

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. You can trigger a measurement on the occurrence of a symbol, and you can have the logic analyzer show symbols in place of values in Listing displays.

Three symbol sources may be used in the logic analyzer:

- Object-file symbols
- Predefined PPC7XX symbols
- User-defined symbols

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 object files in a format that the logic analyzer doesn't understand, you can use a General-Purpose ASCII (GPA) symbol file (see Chapter 10, "General-Purpose ASCII (GPA) Symbol File Format," on page 189).

To load symbols in the Agilent Technologies 16600A/16700A-series logic analysis system:

- 1** Open the logic analyzer module's **Setup** window.
- 2** Click the **Symbol** tab.
- 3** Click the **Object File** tab.

Make sure the label is ADDR.

To view predefined symbols for the PPC7XX

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. The logic analyzer configuration files included with the PPC7XX inverse assembler contain predefined symbols for logic analyzer labels.

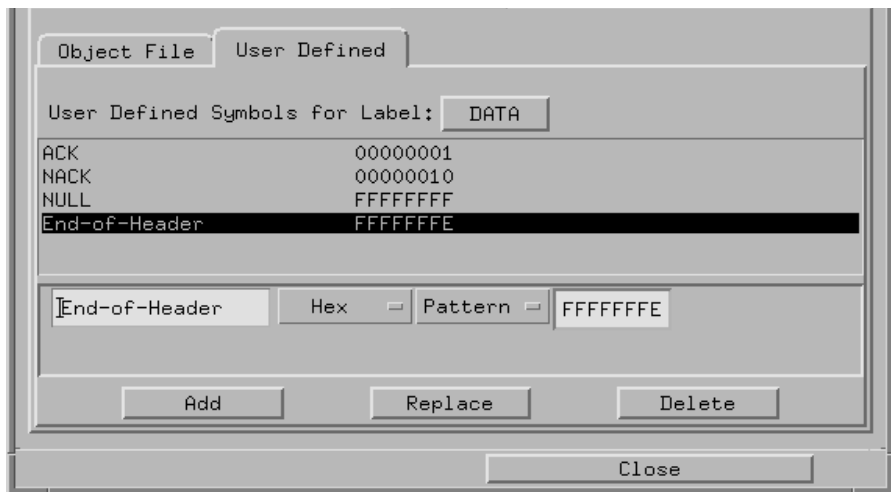
To display the predefined symbols for the PPC7XX:

- 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. The predefined PPC7XX symbols are listed on page 84.

To create user-defined symbols

User-defined symbols are symbols you create from within the logic analyzer user interface by assigning names to values that can be found on the labeled bits. Typically, you assign symbol names to address label values, but you can also define symbols for values on the data, status, or other labels as well. The screen below shows a set of symbols for values found on the DATA label. From the **Setup** dialog, select **Symbols**, and select **User Defined**.



User-defined symbols are saved with the logic analyzer configuration.

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 Agilent Technologies E2498A inverse assembler provides PPC7XX 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

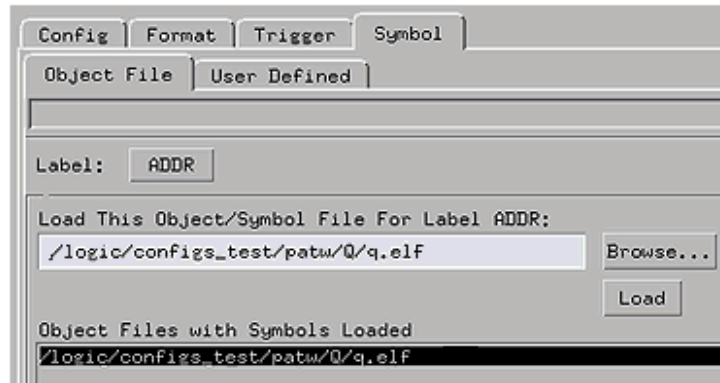
The PPC7XX inverse assembler decodes captured data into program counter (PC) addresses (also known as software addresses) and assembly language mnemonics. Refer to the previous chapter on PPC7XX inverse assembly.

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 190).

To use object file symbols in the 16600/700

To load object file symbols for address values in the 16600/700-series logic analysis system, open the logic analyzer module's Setup window and select the Symbol tab; then, select the Object File tab. Make sure the label is ADDR. From this dialog you can select object files and load their symbol information.



When you load object file symbols into a logic analyzer, a database that correlates symbols and line numbers to addresses in the object file is generated. 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.

If your language tool is not one of those listed on page 113, you can create a symbol file in the General-Purpose ASCII (GPA) file format (refer to the "General-Purpose ASCII (GPA) File Format" chapter).

See Also

Refer to your logic analyzer documentation or online help for information on how to load symbol files.

- If you have an Agilent Technologies 16600/700-series logic analysis system, refer to the online help.
- If you have another logic analyzer, refer to your logic analyzer documentation.

Compilers for PPC7XX

The following PPC7XX compilers and their ELF/DWARF format object files can be used with Agilent Technologies logic analyzers and the Agilent Technologies B4620B Source Correlation Tool Set:

Object File Formats

Language System	Format
Wind River	ELF/DWARF

In order to use symbols in the logic analyzer, file name and line number information must be present in the object file. Your compiler may have options that include or exclude this information.

Limitations: For C++ files, symbols are not demangled. Mangled names are available for use and the trace listing will still correctly correlate to the appropriate source file lines.

When compiling code, if possible, specify that code and data be put in different memory 'blocks'. A 'block' is 32 Kbytes. 32 Kbytes is the smallest area of memory that can be distinguished by each memory block.

It is also useful to put the stack in the data block.

By separating the code and data in this way, the inverse assembler can be configured to properly decode both code and data.

See Also

Contact your Agilent Technologies sales engineer to find out if there are other compilers for the PPC7XX microprocessor that can be used with Agilent logic analyzers.

Loading Symbol Information

Wind River Compiler Options

The following options should be used:

-g	Specifies to generate symbolic debugger information (same as -g2).
-WDDOBJECT=E	Specifies the ELF/DWARF file format.
-WDDENVIRON=cross	Specifies the cross development environment.
-WDDTARGET=PPC740 (or -WDDTARGET=PPC750)	Specifies the type of processor.
-Xdebug-mode=0xff	Turns off Diab Data extensions to the file format.

Wind River provides a utility that you can use to generate the compiler options you need. Enter "dctrl -t" and follow the instructions. When it is finished, it will present you with a string that you can use for the compiler options.

Please refer to the language tool supplier's documentation for more information about the options available.

More information is available on the World Wide Web at:

http://www.windriver.com/products/html/dcc_compiler.html

Inverse assembler generated PC (software address) label

In the Agilent Technologies 16600A/16700A-series logic analysis system, the PPC7XX inverse assembler generates a "PC" label. The PC 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 this line in listing" commands in the 16600A/16700A-series logic analysis system perform a pattern search on the PC 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 source code line will generate many assembly instructions. The "Goto this line in listing" commands will not find a given source code line 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 begin after the first assembly instruction of the loop has been executed. A "Goto this line in listing" command would not find the source line.

Triggering on Symbols and Source Code

When setting up trigger specifications to capture PPC7XX 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

You should use an 8-bit alignment to avoid missed triggers. The PPC7XX has a 64-bit data bus. Instructions for the PPC7XX 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 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. For the PPC7XX 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 correlate relocatable code using the address offset

You need to adjust the source correlation tool set to compensate for relocatable code segments or memory management units that produce fixed code offsets. The offset field in the trigger menu allows you to offset the symbol address. 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 8 (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 PPC7XX, 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 PPC7XX 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.

Using storage qualification

You should configure the logic analyzer's storage qualification capabilities to store only those cycles that correspond to software execution (non-idle, etc.). To set up the store qualification to store only non-idle states, first verify that the Store Qualify field is set to "Term Selection", (as opposed to "Anystate"). Next, replace the STAT label with the ACKs label. While the cursor is over the STAT label, hold down the right mouse button, select the "Replace label..." option, select the ACKs label from the list. Now change the format of the ACKs label to Binary Not Pattern. Finally, edit the pattern field to become "X1111". Only states that are not Idle will be stored.

The source correlation tool set can exhibit long responses 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.

Modes of Operation

The inverse assembler can be used in three different analysis modes: State-per-ack, State-per-clock, or Timing. The following sections describe these modes and how to configure the logic analyzer for each mode.

State-per-ack mode

In State-per-ack mode, the logic analyzer uses trigger sequencer store qualification to capture only address and data-acknowledge cycles. This is the default mode set up by the configuration files.

State-per-ack mode provides the greatest information density in the logic analyzer acquisition memory.

State-per-clock mode

In State-per-clock mode, every clock cycle is captured by the logic analyzer, including idle and wait states between and during tenures. To configure the logic analyzer for State-per-clock mode:

- 1 Select the logic analyzer icon.
- 2 Select **Setup...** from the menu.
- 3 Select the **Trigger** tab.
- 4 Change the **Store by default** mode to **Anything**.
- 5 Ensure that the trigger sequence is set to find the pattern “XXXXXXXX” one time and then trigger and fill memory.

Timing mode

In Timing mode, the logic analyzer samples the microprocessor pins asynchronously, typically with 4-ns resolution. To configure the logic analyzer for timing analysis:

- 1 Select the logic analyzer icon.
- 2 Select **Setup...** from the menu.
- 3 Select the **Sampling** tab.
- 4 Change the type option from **State Mode** to **Timing Mode**.

Modes of Analysis

The inverse assembler offers two modes of analysis for PowerPC 7XX microprocessors: traditional inverse assembly, and inverse assembly with cache-on trace reconstruction. The mode is set in the **Invasm Preferences** window using the **External Bus Decoding** dialog.

Inverse assembly analysis

The inverse assembler lets you obtain displays of PowerPC 7XX operations in PowerPC instruction mnemonics, as described in *PowerPC Microprocessor Family: The Programming Environments for 32-Bit Microprocessors*. In addition, information that is processed in cache may be displayed using the cache-on trace reconstruction feature of the inverse assembler.

The inverse assembler requires the Agilent Technologies 16600/700-series logic analyzers. It provides much more information when used together with the Agilent Technologies B4620B Source Correlation Tool Set. Source correlation performs a correlation of the addresses from cache with the high-level code execution.

Modes of Analysis

Configuring the 1660/1670/16500B/C-
Series Logic Analyzer

Chapter 6: Configuring the 1660/1670/16500B/C-Series Logic Analyzer

This chapter describes modes of operation for the E2498A Inverse Assembler. It also describes data, symbol encodings, and information about the inverse assembler.

The information in this chapter is presented in the following sections:

- Modes of operation
- Logic analyzer configuration
- Using the inverse assembler

Configuring 1660/1670/16500B/C-Series Logic Analysis Systems

The information in this chapter is specific to systems using 1660/1670 or 16500B/C-series logic analyzers. For systems using 16600/700-series logic analysis systems, see Chapter 5, “Configuring the 16600/700-Series Logic Analyzer,” beginning on page 73.

To load configuration files and the inverse assembler—1660/1670/16500B/C-series logic analyzers

If you have a 1660-series, 1670-series, or logic analyzer modules in a 16500B/C mainframe use these procedures to load the configuration file and inverse assembler.

The first time you set up the logic analyzer, make a duplicate copy of the flexible disk. For information on duplicating disks, refer to the reference manual for your logic analyzer.

For logic analyzers that have a hard disk, you might want to create a directory such as PPC7XX on the hard drive and copy the contents of the floppy onto the hard drive. You can then use the hard drive for loading files.

- 1** Insert the logic analyzer flexible disk in the front disk drive of the logic analyzer.
- 2** Select the "Flexible Disk" menu.
- 3** Configure the menu to "Load" the analyzer configuration from disk.
- 4** Select the appropriate module (such as "100/500 MHz LA" or "Analyzer") for the load.
- 5** Use the knob to select the appropriate configuration file.

Your configuration file choice depends on which analyzer you are using. See table on page 127.

- 6** Execute the load operation to load the file into the logic analyzer.

The logic analyzer is configured for PPC7XX analysis by loading the appropriate PPC7XX configuration file. Loading the indicated file also automatically loads the pipelined-systems inverse assembler. For information on the difference between the pipelined and nonpipelined inverse assemblers, refer to "To select a different inverse assembler" on page 128.

Logic analyzer configuration files (1660/1670/16500B/C-series logic analyzers)

The following table shows the configuration files for the supported logic analyzers.

Analyzer Model	Analyzer Description (modules only)	Configuration File $V_{I/O} = 3.3V$	Configuration File $V_{I/O} = 1.8V$
1660A/AS/C/CS	na	C7XXJ1	na
1670A/D	na	C7XXM2	na
16550A (one card)	100 MHz STATE 500 MHz TIMING	C7XXF1	C7XX_AVCF1
16550A (two card)	100 MHz STATE 500 MHz TIMING	C7XXF2	C7XX_AVCF2
16554A (one card)	0.5M SAMPLE 70/250 MHz LA	C7XXM0	C7XX_AVCM0
16554A (two card)	0.5M SAMPLE 70/250 MHz LA	C7XXM2	C7XX_AVCM2
16554A (three card)	0.5M SAMPLE 70/250 MHz LA	C7XXM3	C7XX_AVCM3
16555A/D (one card)	1.0M SAMPLE 100/500 MHz LA	C7XXM0	C7XX_AVCM0
16555A/D (two card)	1.0M SAMPLE 110/500 MHz LA	C7XXM2	C7XX_AVCM2
16555A/D (three card)	1.0M SAMPLE 110/500 MHz LA	C7XXM3	C7XX_AVCM3
16556A/D (one card)	1.0M SAMPLE 100/400 MHz LA	C7XXM0	C7XX_AVCM0
16556A/D (two card)	1.0M SAMPLE 110/400 MHz LA	C7XXM2	C7XX_AVCM2
16556A/D (three card)	1.0M SAMPLE 110/400 MHz LA	C7XXM3	C7XX_AVCM3
16557D (one card)	1.0M SAMPLE 135/500 MHz LA	C7XXM0	C7XX_AVCM0
16557D (two card)	1.0M SAMPLE 135/500 MHz LA	C7XXM2	C7XX_AVCM2
16557D (three card)	1.0M SAMPLE 135/500 MHz LA	C7XXM3	C7XX_AVCM3

To select a different inverse assembler

The Agilent Technologies E2498A contains two inverse assemblers, one for pipelined systems and one for non-pipelined systems.

The pipelined-systems inverse assembler (I7XXEP) looks for AACK before TA; this inverse assembler is loaded by default by the configuration file.

The non-pipelined-systems inverse assembler (IA7XXE) waits until the last TA to look for AACK. For non-pipelined systems, the inverse assembler must be loaded after the configuration file.

The "E" in the inverse assembler filenames indicates these inverse assemblers contain the enhanced feature set. The enhanced version is loaded by default if the logic analyzer has the correct software version.

It is not necessary to load a different inverse assembler file in Agilent Technologies 16600/700 logic analysis systems. The correct inverse assembler file is always loaded by default with the configuration file.

To load a different inverse assembler file in the other logic analyzers, repeat steps 1 - 6 in "To load configuration files and the inverse assembler—1660/1670/16500B/C-series logic analyzers" on page 126, but for step 5 select the inverse assembler.

Modes of Operation

The E2498A inverse assembler can be used in three different analysis modes: State-per-ack, State-per-clock, or Timing. The following sections describe these modes and how to configure the logic analyzer for each mode.

State-per-ack mode

In State-per-ack mode, the logic analyzer uses trigger sequencer store qualification to capture only address and data-acknowledge cycles. This is the default mode set up by the configuration files.

State-per-ack mode provides the greatest information density in the logic analyzer acquisition memory.

State-per-clock mode

In State-per-clock mode, every clock cycle is captured by the logic analyzer, including idle and wait states between and during tenures. To configure the logic analyzer for State-per-clock mode:

- Select the Trigger dialog and change the store qualification to “anystate”.

Timing mode

In Timing mode, the logic analyzer samples the microprocessor pins asynchronously, typically with 4-ns resolution. To configure the logic analyzer for timing analysis:

- 1** Select the Configuration menu of the logic analyzer.
- 2** Select the Type field for Analyzer 1.
- 3** Select Timing.

Logic Analyzer Configuration

The following sections describe the logic analyzer configuration as set up by the configuration files.

It is strongly recommended that you do not change the setup related to the PPC7XX sampling, format, pod assignment or configuration dialogs. The configuration file (loaded by the Setup Assistant in Agilent Technologies 16600/700-series logic analysis systems) will configure the logic analyzer for making measurements of the PPC7XX.

Format menu

This section describes the organization of PPC7XX 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. The tables on the following pages show the signals used in the STAT label and the predefined symbols set up by the configuration files.

The Agilent Technologies 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.

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

The configuration software sets up the analyzer format dialog to display either eight or ten pods of data, depending on the analyzer.

Status Encoding

Each of the bits of the STAT label is described in the table below. Most of the status and control signals on the PowerPC 7XX are active low (“-” suffix). To conserve display space, the “-” is omitted in many of the Format definitions.

The inverse assembler uses STAT bits TC0, TSIZE...2, TT0...3, TBST, TA, AACK, ARTRY, DRTRY, ABB, TEA, and TS. The signal-to-connector tables in the “Hardware Reference” chapter list all the PPC7XX signals probed and their corresponding analyzer channels.

Status Bit Description

Status Bit	Description
BR-	The PowerPC 7XX asserts Bus Request to indicate that it has business to conduct on the address bus
BG-	The memory system asserts Bus Grant to allow the 7XX onto the address bus
ABB-	Address Bus Busy indicates that the address bus is in use
TS-	The PowerPC 7XX asserts TS- for one cycle to commence a transaction. It also serves as the data bus request signal if the TT signals indicate a data transfer.
XATS-	XATS commences a "programmed i/o" (PIO) sequence in the extended address transfer protocol (not available on 7XXe).
DBG-	The memory system asserts Data Bus Grant to allow the 7XX onto the data bus.
DBWO-	The memory system may assert Data Bus Write Only to allow the 7XX to envelope a data write (snoop push, typically) between the address and data phases of a data read.
DBB-	Indicates Data Bus Busy.
AACK-	The memory system asserts AACK for one cycle to acknowledge an address.
ARTRY-	The memory system may assert ARTRY to cause the 7XX to back off the bus and retry the transaction.
TA-	The memory system asserts TA to acknowledge a data transaction.

Status Bit	Description
DRTRY-	The memory system may assert DRTRY to cancel the effect of a TA in the previous cycle.
TEA-	The memory system may assert TEA to indicate a transfer error, e.g. an unmapped part of the address space.
TT 0:4	The Transfer Type signals indicate the direction and purpose of a bus transaction.
Atomic(TT0)	Usually, TT0 asserted indicates an atomic (e.g., stwcx.) transfer.
R/-W (TT1)	TT1 is high for a read, low for a write.
InvlDt (TT2)	Usually, the PowerPC 7XX asserts TT2 to indicate that the corresponding cache line should be invalidated by other processors.
A Only (TT3)	TT3 high indicates that there is data associated with the current address. TT3 low usually indicates an address-only transaction.
TC 0:1	The Transfer Code outputs provide further information about the current transfer. For a read, they indicate whether instructions or operands are being fetched.
TBST-	When asserted, TBST- indicates a four-beat burst transfer of eight words.
TSIZ 0:2	Indicates the size for the data transfer in conjunction with TBST-.
WT-	Write Through indicates a write-through transaction that should be pushed through local caches to shared memory.
CI-	Cache Inhibit indicates that the 7XX will not cache a read.
GBL-	Global indicates the transaction is global, i.e., should be snooped by all other caching devices on the bus.
SRESET-	A falling-edge input causes the 7XX to undergo a soft reset.
HRESET-	Input asserted causes the 7XX to undergo a hard reset.
CKSTP-	Input asserted causes the 7XX to undergo machine check processing.

Chapter 6: Configuring the 1660/1670/16500B/C-Series Logic Analyzer
Logic Analyzer Configuration

Status Bit	Description
INT-	Input indicates an external interrupt is pending.
CHECKSTOP	Output indicates that the 7XX has entered the machine check state, i.e., stopped.
QREQ-	The PowerPC 7XX asserts Quiescent Request to indicate that it wants to be, or is, in a snooze mode.

Predefined Logic Analyzer Symbols

The configuration software sets up symbol tables on the logic analyzer. The tables define a number of symbols which make several of the STAT fields easier to interpret. The following table lists the symbol descriptions.

Symbol Description

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
	5byte	1011
	6byte	1101
7byte	1111	
TT/TBST- †	Kill Block	01100
	Wr Graphics	10100
	Rd Graphics	11100
	Clean Block	00000
	Write	00010
	Wr/Kill	00110
	Read	01010
	Rd/Flush	01110
	Wr/Atomic Flush	10010
	Read Atomic	11010
	Rd/Flush Atomic	11110
	Flush Block	00100
	DSYNC	01000
	eieio	10000
(reserved)	10110	
TLB Invalidate	11000	
STAT	inst fetchxxxx xxxx xxxx xxx1 xxxx 1xxx xxxx 0xxx	

Chapter 6: Configuring the 1660/1670/16500B/C-Series Logic Analyzer **Logic Analyzer Configuration**

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

Trigger dialog

This section describes some PowerPC 7XX-specific considerations in triggering the analyzer. You can use the Trigger dialog to change the triggering and storage qualification to include or exclude specified cycles. The trigger specification set up by the software stores all states.

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

Qualifying Stored Data

The trigger dialog determines what will be acquired by the analyzer and when it will be acquired. The E2498A software pre-configures a storage qualification term to exclude wait and idle states from the analyzer's memory.

The configuration software renames a pattern term to "idle" and assigns it a pattern with AACK, ARTRY, TA, and DRTRY, all high (not asserted). The sequencer is programmed to store only states \neq idle. That is, only states where one or more of these signals is asserted will be stored.

Configuring for State-per-clock mode

To configure the analyzer to store all states including wait and idle states, change the storage qualification to capture all states (state-per-clock).

Change storage qualification from \neq **idle** to **anystate**.

Capturing an address

To accurately trigger on a specific address, create a term with two labels: ADDR and AACK. Enter the address in the ADDR field of a trigger term and enter 0 in the AACK field of the term. This will prevent false triggering on a floating address bus.

The instruction addresses presented on the PowerPC 7XX address bus always end in hex 0 or hex 8. When the instruction cache is enabled, the 7XX will burst four data beats per address and will not update the address as it bursts. To reliably trigger on the fetch of a particular address when bursting, the least significant three bits of the address must be “don't cares.” Change the base of the ADDR label to Binary to enter the 3 X's.

Using the Inverse Assembler

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

Before the inverse assembler will correctly disassemble information captured on the PPC7XX address bus, you must make sure the target system's cache has been disabled.

To disable the instruction cache on the PPC7XX

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
```

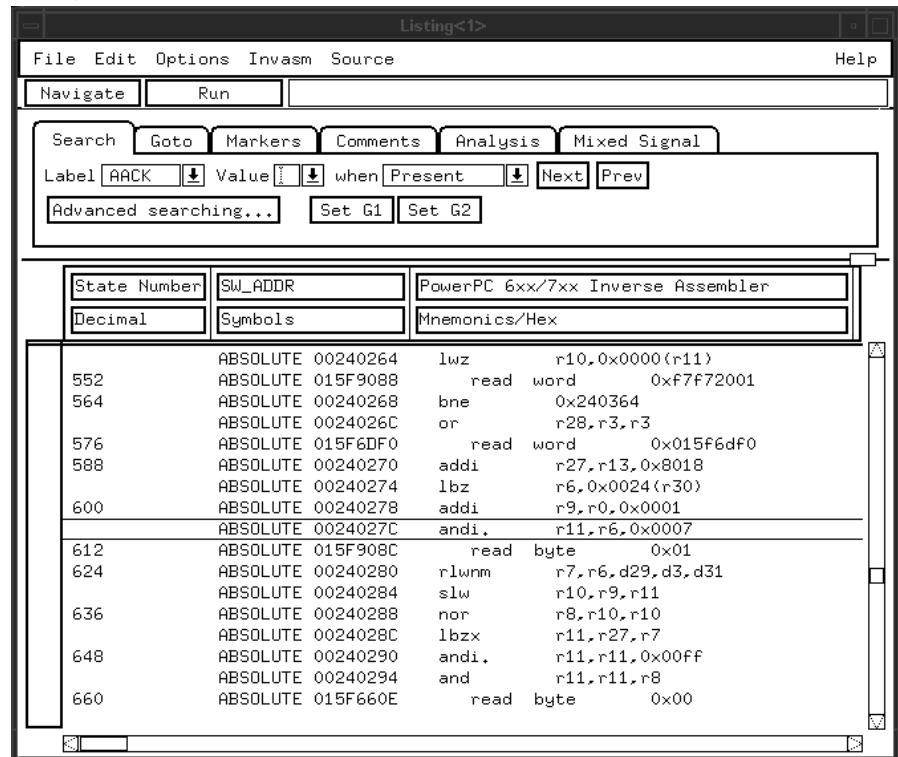
- To invalidate and disable both caches use:

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

To display captured state data

The logic analyzer displays captured state data in the Listing menu. The inverse assembler display is obtained by setting the base for the DATA label to Invasm. The following figure shows a typical Listing menu.

Listing Menu.

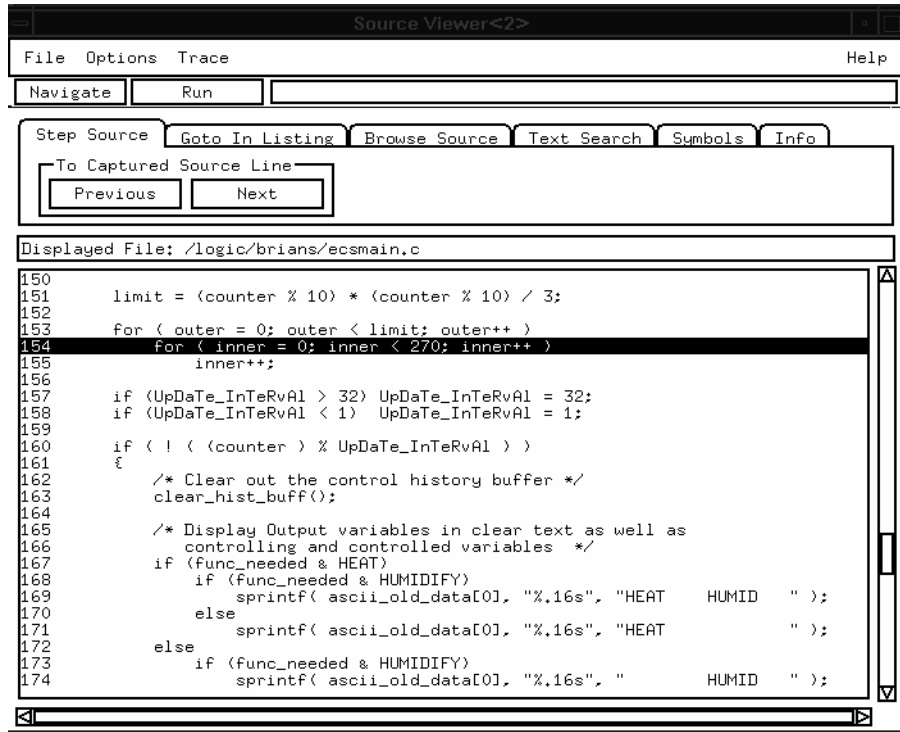


This paragraph applies only when using the Agilent Technologies 16500C system. The columns on the left of the inverse assembly data display are the least significant hexadecimal digits of an instruction or burst address. These may be useful for matching an execution trace to an assembly listing. Because the PowerPC 7XX presents one address and then reads two or eight instructions for each address, the less-significant bits are synthesized by the disassembler.

Displaying Data with the Agilent Technologies B4620B Source Correlation Tool Set

Source correlation correlates the addresses from cache with the high-level code execution. The figure below shows execution of data that is correlated to the data shown on the previous page

Source Correlation Tool Set Data.



The screenshot shows the Source Viewer application window. The title bar reads "Source Viewer<2>". The menu bar includes "File", "Options", "Trace", and "Help". Below the menu bar are "Navigate" and "Run" buttons. A toolbar contains buttons for "Step Source", "Goto In Listing", "Browse Source", "Text Search", "Symbols", and "Info". A sub-toolbar below the toolbar has a label "To Captured Source Line" and "Previous" and "Next" buttons. The main display area shows the following C code:

```
150
151     limit = (counter % 10) * (counter % 10) / 3;
152
153     for ( outer = 0; outer < limit; outer++ )
154         for ( inner = 0; inner < 270; inner++ )
155             inner++;
156
157     if (UpDaTe_InTeRvAl > 32) UpDaTe_InTeRvAl = 32;
158     if (UpDaTe_InTeRvAl < 1) UpDaTe_InTeRvAl = 1;
159
160     if ( ! ( (counter) % UpDaTe_InTeRvAl ) )
161     {
162         /* Clear out the control history buffer */
163         clear_hist_buff();
164
165         /* Display Output variables in clear text as well as
166            controlling and controlled variables */
167         if (func_needed & HEAT)
168             if (func_needed & HUMIDIFY)
169                 sprintf( ascii_old_data[0], "%.16s", "HEAT    HUMID    " );
170             else
171                 sprintf( ascii_old_data[0], "%.16s", "HEAT          " );
172         else
173             if (func_needed & HUMIDIFY)
174                 sprintf( ascii_old_data[0], "%.16s", "          HUMID    " );
```

Inverse assembler output format

The following paragraphs explain the operation of the inverse assembler and the results you can expect under certain conditions.

Interpreting Data

General purpose registers are displayed as r0, r1,..., r31. Floating point registers are displayed as f0, f1,..., f31. Condition registers are displayed as cr0, cr1,..., cr7. Special purpose registers are displayed using their mnemonic.

Most numerical data is displayed in hexadecimal, for example, "lwzr28, 0x0044(r1)."

Bit numbers and shift counts are displayed in decimal with a "d" prefix, for example, "cror d31,d31,d31."

A few instructions display their operands in binary with a "b" prefix, for example, "mtfsfi 4,b0101."

The inverse assembler decodes the full 32-bit mode PowerPC instruction set architecture. Instructions that are 64-bit mode or optional instructions not implemented on the PPC7XX are decoded as "illegal". Any instruction that does not decode to a valid opcode is shown as "unknown".

When these un-implemented opcodes are encountered, the instruction mnemonic has a "?" prefix. If a reserved bit is set in an instruction opcode field, a "?" is appended most often to the mnemonic, but in some cases to an operand.

An instruction word of 00000000 is decoded as "illegal." Otherwise, if an opcode is invalid, it is shown as "Undefined 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 7XX 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 an interrogation point "?". 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.

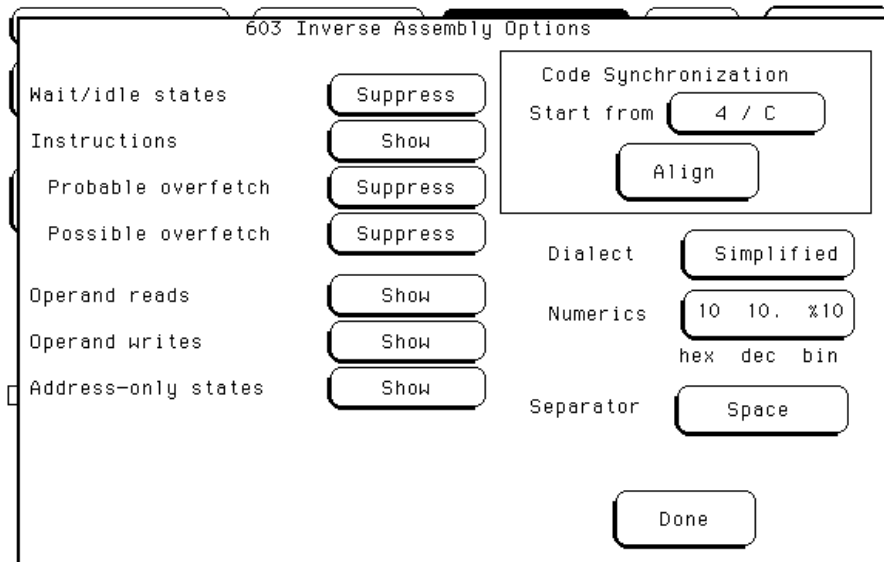
Little-Endian Mode

The inverse assembler is designed to support the native big-endian mode of operation on the PowerPC 7XX. When operating in little-endian mode, the 7XX 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.

In little-endian operation, in a given data beat, the instruction word from DL0...31 (DATA_B label; external address xxx4) will be dispatched before the instruction word from DH0...31 (DATA label; external address xxx0). You can compensate for this by exchanging the DATA and DATA_B labels in the Format menu. However, while this will correctly order 32-bit word reads on the 64-bit data bus, it will cause byte- and half- word reads and writes to appear on the opposite side of the bus, and swap the halves of double-word reads and writes.

To use the Invasm menu (1660, 1670, and 16500B/C mainframes)

The Agilent Technologies 1660, 1670, and 16500B/C mainframes provide an Inverse Assembly Options menu, which is accessed by pressing the Invasm button in the Listing window. The Inverse Assembly Options menu contains three functions: display filtering with Show/Suppress selections, Code Synchronization, and Display Options. The figure below shows the Inverse Assembly Options menu.



Display Filtering

Display filtering is similar to the description on page 164.

Code Synchronization

Code Synchronization allows you to correct the display when the inverse assembler incorrectly predicts a conditional branch as taken and incorrectly marks subsequent states as overfetch.

Extended Mnemonics

PowerPC assemblers support a number of extended mnemonics for some popular assembly language instructions as described in the PPC7XX User's Manual. The E2498A inverse assembler supports the following extensions:

- 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.
- 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 mtrf instruction field mask specifies the entire cr, it is decoded as mtr.

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.
```

Using the Inverse Assembler

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)	inslwi 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

Chapter 6: Configuring the 1660/1670/16500B/C-Series Logic Analyzer
Using the Inverse Assembler

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.

With the PPC7XX inverse assembler, the logic analyzer is configured, and labels are created (formatted) for the logic analysis channels when configuration files are loaded. See “Configuring the 16600/700-Series Logic Analyzer” on page 73 or “Configuring the 1660/1670/16500B/C-Series Logic Analyzer” on page 123, depending on which analyzer you are using.

You can load program object file symbols into the logic analyzer when configuring it (see “To load object file symbols” on page 108).

This chapter describes setting up logic analyzer triggers when using the Agilent Technologies E2498A inverse assembler and the Agilent Technologies B4620B source correlation tool set.

See Chapter 8, “Displaying Captured Processor Execution,” beginning on page 159 for information on displaying captured data.

Trigger sequence

The Trigger sequence is set up by the software to store all states.

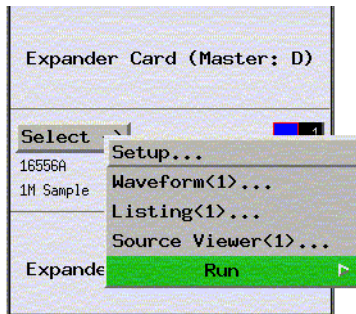
NOTE:

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

Setting Up Logic Analyzer Triggers

To set up logic analyzer triggers

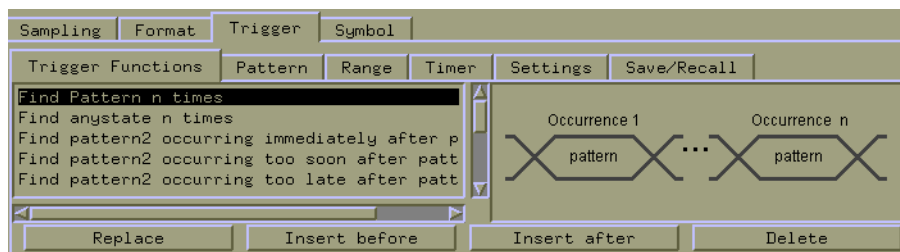
- 1 Open the logic analyzer's Setup window.



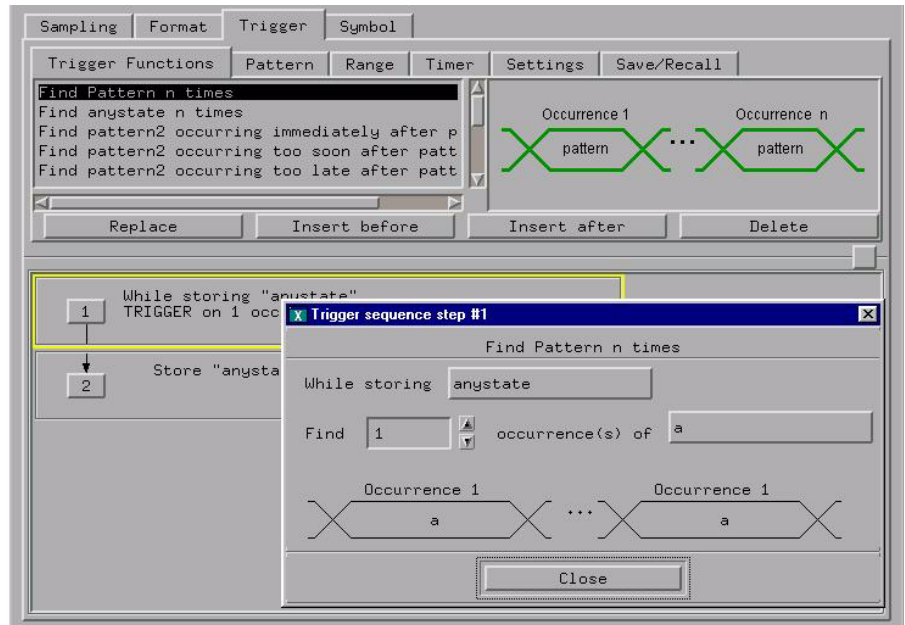
- 2 Select the Trigger tab.



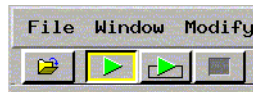
- 3 Select the trigger function that will be used in the logic analysis measurement.



4 Set up the trigger sequence.



5 Run the measurement.



See Also

The Agilent Technologies (16600/700-series logic analysis system's on-line help for more information on setting up logic analyzer triggers.

Triggering on Symbols and Source Code

When setting up trigger specifications to capture PowerPC 7XX execution:

- 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 the Address Offset

You need to adjust the source correlation tool set to compensate for relocatable code segments or memory management units that produce fixed code offsets.

The logic analyzer has an address offset field to help facilitate this.

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

Using Storage Qualification

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

The source correlation tool set can exhibit long responses 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.

NOTE:

Storage qualification can not be used when cache-on trace reconstruction is enabled. These features require that all processor cycles are stored. See "Using Cache-On Trace Reconstruction" on page 88.

To qualify stored data

The configuration file sets up the logic analyzer clock.

The logic analyzer identifies valid states based on the clock signals. The logic analyzer refers to the system clock as the K clock, and the \overline{TA} signal as the M clock.

The default clocking combination “K rising and M low” latches address and data when the system clock is rising and \overline{TA} is asserted.

The memory controller in the PowerPC 7XX will allow data to be latched on the falling edge of the system clock (K falling). If your data appears incorrect, check the processor’s memory controller registers to ensure that the clocking configurations of the processor and the logic analyzer match.

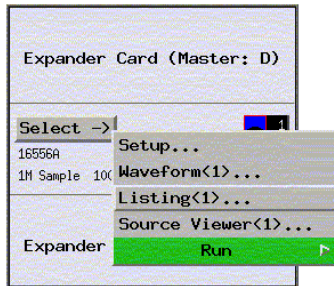
For State-per-clock acquisition, change the clock qual Q1 to Off.

Chapter 7: Capturing Processor Execution
Triggering on Symbols and Source Code

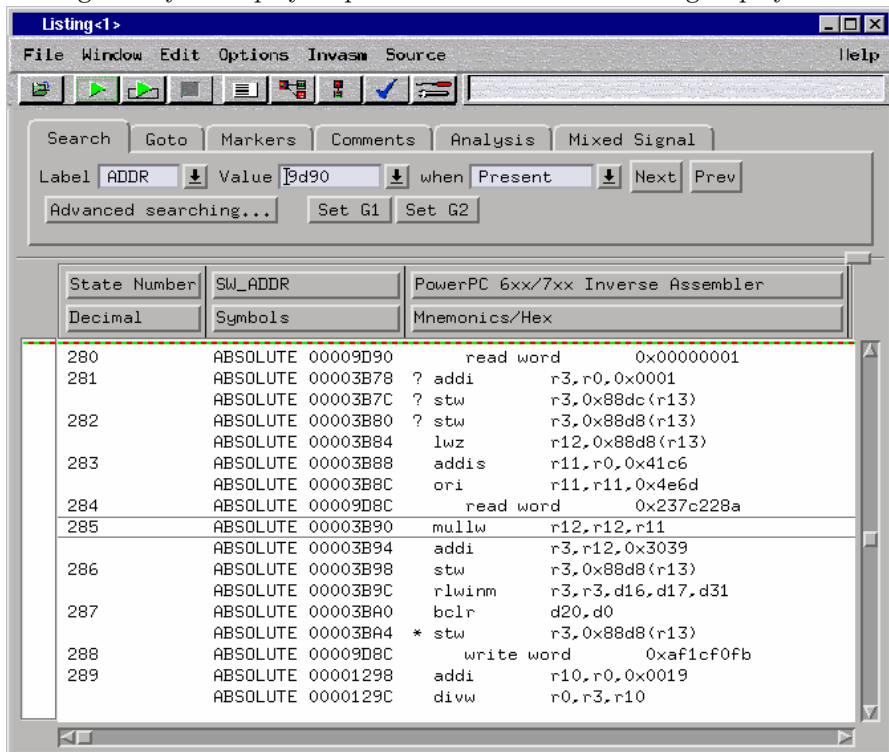
Displaying Captured Processor Execution

To display captured state data

- 1 Open the Listing display window.



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



On the 16600/700-series logic analysis systems, the entire synthesized address appears under the label “SW_ADDR”. The actual address bits presented by the PowerPC 7XX may be observed under the ADDR label.

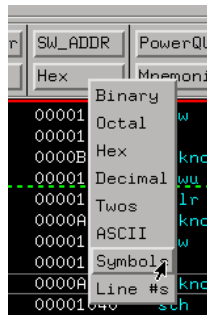
See Also

“To use the inverse assembler filters” on page 164 for information on displaying or hiding certain types of microprocessor bus cycles.

The Agilent Technologies 16600/700-series logic analysis system on-line help for information on using the Listing display.

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.

See Also

“To load object file symbols” on page 108.

To interpret the inverse assembled data

The following paragraphs explain the operation of the inverse assembler and the results you can expect under certain conditions.

Interpreting Data

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 r1,0xfff8(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 optional instructions not implemented on the PowerPC 7XX. When these 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.

SW_ADDR Label

When an Agilent Technologies 16600/700-series logic analysis system is being used, the inverse assembler generates a “SW_ADDR” field. This field is the Software Address generated by the inverse assembler.

The SW_ADDR label cannot be used exactly like other labels. For example, when loading symbols, you will notice that the SW_ADDR label is not in the list of labels that the symbols can be loaded into. Symbols should still be loaded into the ADDR label. The main purpose of the SW_ADDR label is for correlation of the listing with source code using the Agilent Technologies B4620B Source Correlation Tool Set.

Overfetch Marking

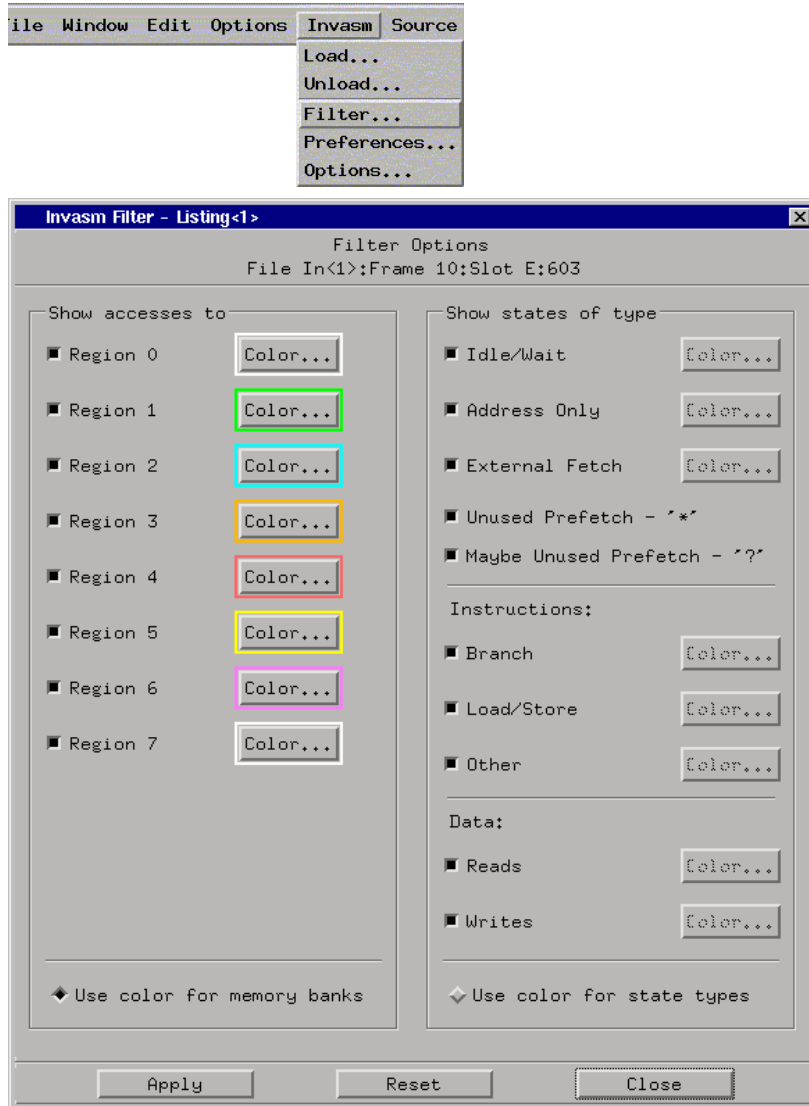
Overfetch refers to instructions which are fetched but not executed by the processor. They may arise from the following sources:

- When the 7XX 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 the inverse assembler filters

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



The inverse assembler lets you Show or Suppress several types of states. This dialog is called display filtering. States can be filtered according to what type of cycle the state is, or according to which memory bank was accessed for the cycle.

The show/suppress settings do not affect the data that is stored by the logic analyzer; they only affect whether that data is displayed or not. You can examine the same data with different settings, for different analysis requirements.

This dialog allows faster analysis in two ways. First, you can filter unneeded information out of the display. For example, suppressing idle states will show only states in which a transaction was completed.

Second, you can isolate particular operations by suppressing all other operations. For example, you can show branches, with all other states suppressed, allowing quick analysis of branch instructions.

Agilent Technologies 16600/700-series logic analysis systems provide one additional feature for analyzing data. Instead of (or in addition to) showing or suppressing states, the selected states can also be shown in color.

Color can only be used for distinguishing either memory region accesses or cycle types, but not both at the same time.

Options

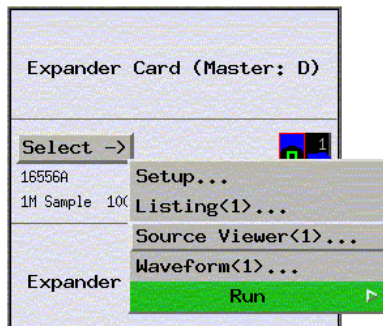
The options dialog lets you change the width of the symbols in the disassembly column. It also allows you to display symbols (globals), hex, or line numbers.

To view the source code associated with captured data

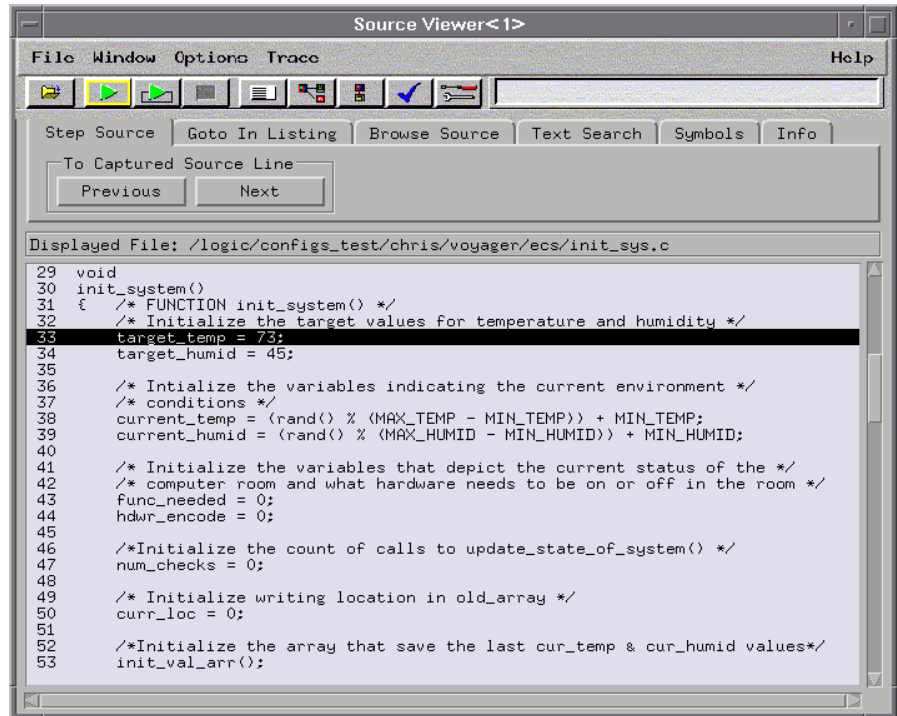
- In the Listing display window, select Source Viewer from the Source menu.



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



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 108).



Inverse Assembler Generated SW_ADDR (Software Address) Label

In the Agilent Technologies 16600/700-series logic analysis system, the PPC7XX 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 this line in listing” commands in the Agilent Technologies 16600/700-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 this line 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

begin after the first assembly instruction of the loop has been executed. A “Goto this line 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 Agilent Technologies 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 Agilent Technologies 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.

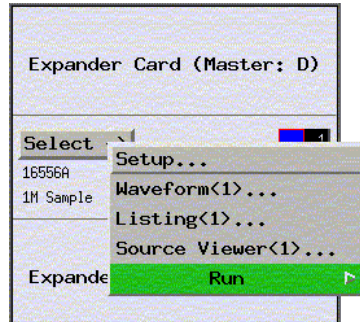
Source File Version Control. If the source code files are under a source code or version control utility, check the file names and paths carefully. These utilities can change source code file paths and file names. If these names are changed from the information contained in the symbol file, the source 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

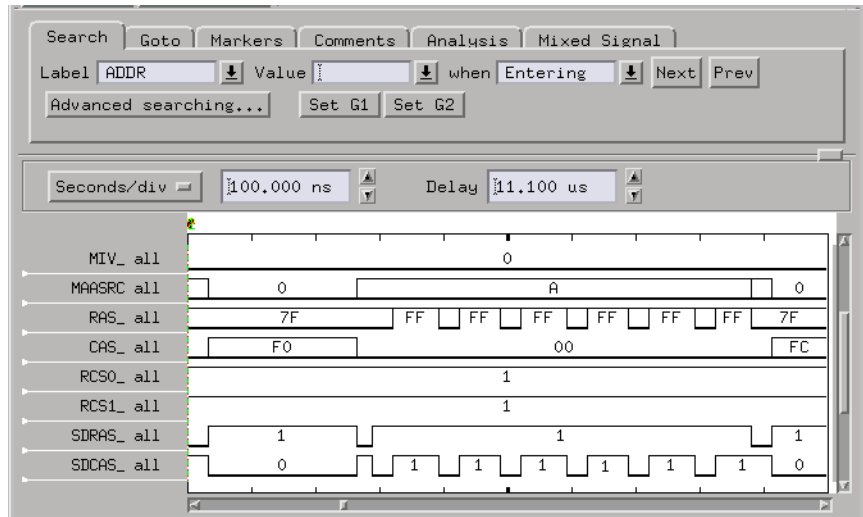
More information on configuring and using the source correlation tool set can be found in the on-line help for your logic analysis system.

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.



Coordinating Logic Analysis with
Processor Execution

This chapter describes how to use an analysis probe, an emulation module, and other features of your Agilent Technologies 16600A or 16700 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 analysis probe, to acquire data from the processor bus while it is running full-speed.
- Your emulation probe, 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 probe, and to display or change target registers and memory.
- Display tools including the Listing tool, Chart tool, and System Performance Analyzer tool to make sense of the data collected using the analysis probe.
- Your debugger, to control your target system using the emulation probe. Do not use the debugger at the same time as the Emulation Control Interface.
- The Agilent Technologies 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 probe) 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. Use the Source Viewer to set analyzer triggers. The Source Viewer window is available only if you have licensed the Agilent Technologies 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 online help contains examples of measurements which will save you time throughout the phases of system development: hardware turn-on, firmware development, software development, and system integration.

A few of the many things you can learn from 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."

Triggering the Emulation Probe from the Analyzer

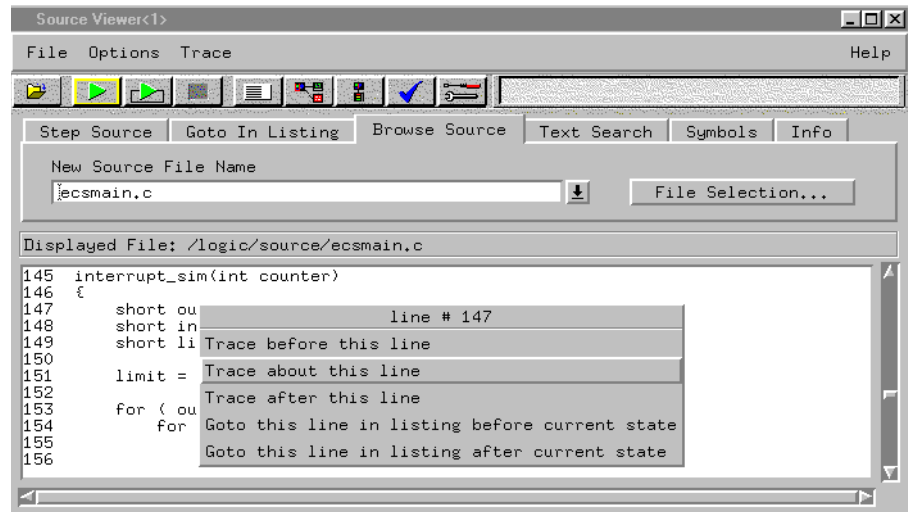
The logic analyzer may be used to signal the emulation probe to stop (break) the target processor. This is done from either the Source Viewer window or the Intermodule window. If you are using the Agilent Technologies B4620B Source Correlation Tool Set, using the Source Viewer window is the easiest method.

To stop the processor when the logic analyzer triggers on a line of source code (Source Viewer window)

If you have the Agilent Technologies B4620B Source Correlation Tool Set, you can easily stop the processor when a particular line of code is reached.

- 1 Select the logic analyzer module icon in the System window, and choose **Source Viewer...**
- 2 In the Source Viewer 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.



3 Select Trace→Enable - Break Emulator On Trigger.

The emulation probe 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**.

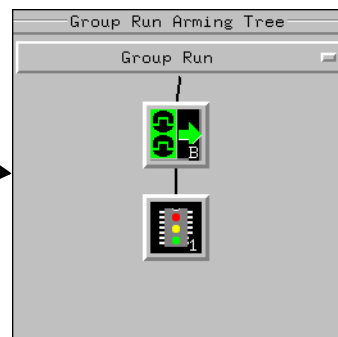
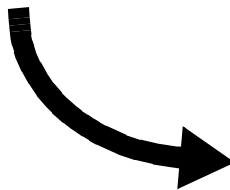
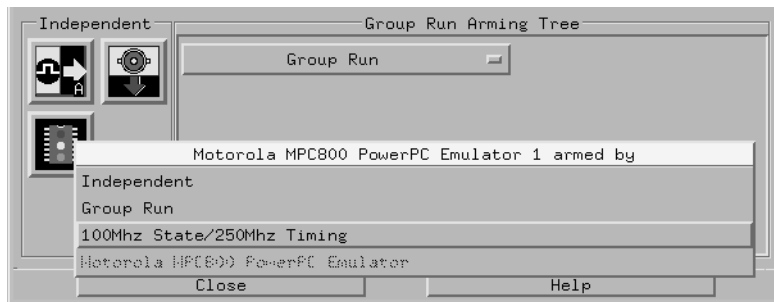
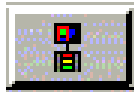
4 Select Group Run in the Source window (or other logic analyzer window).

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

To stop the processor when the logic analyzer triggers (Intermodule window)

Use the Intermodule window if you do not have the Agilent Technologies 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 Select the **Intermodule** icon in the System window.
- 3 In the Intermodule window, select the emulation module icon, then select the analyzer which is intended to trigger it.



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

- 4 Select **Group Run** in the Source window (or other logic analyzer window).
- 5 If your target system is not already running, select **Run** in the emulation Run Control window to start your target.

See Also

See the online 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 that your target can support.

The amount of skid will depend on the processor's execution speed and whether code is executing from the cache. See the *Emulation for the PowerPC7XX User's Guide* for information on how to configure the clock speed.

To stop the analyzer and view a measurement

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

NOTE:

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.

Example

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

- 1** The analyzer triggers.
 - 2** The trigger ("Break In") is sent to the emulation probe.
 - 3** The emulation probe 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.
 - 6** If the trigger position is not "End", the analyzer may continue waiting for more states.
 - 7** 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. To do this, you could set a breakpoint at the start of the function then use this measurement to see how the function is getting called.

NOTE:

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

If you have already set up an intermodule measurement, you must “undo” it by setting all components in the intermodule window to run independently.

To capture a trace before the processor halts

- 1 Set the sampling to **state mode** and the trigger condition to **Run until user stop**.
- 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 target processor halts, select **Stop** in the logic analyzer window to complete the measurement.

NOTE:

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 probe (described in the next section).

Triggering the Logic Analyzer from the Emulation Probe

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

If you are only using a state analyzer to capture the processor bus then it will be much simpler to use “Tracing until processor halts” as described on page 180.

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

The emulation probe trigger signal

The trigger signal coming from the emulation probe 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 that are 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 probe
- Emulation modules connected in series
- Emulation probe connected to Port Out

Here are some examples:

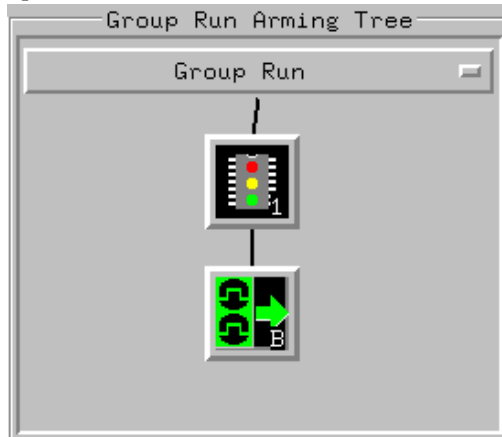
- If Group Run is armed from Port In and an emulation probe is connected to Group Run, then any Port In signal will cause the emulation probe to go into monitor. The Group Run button does not have to be selected for this to operate.
- If two emulation probes are connected together so that one triggers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

The current emulation probe 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 probe does not mean that the Group Run will Run the emulation probe.

The emulation probe 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:



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 probe 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 probe 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 that appear in the listing.

You can often distinguish these additional states because the time tags will be in the μs and ms 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 μs and ms 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 the 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 - timing mode

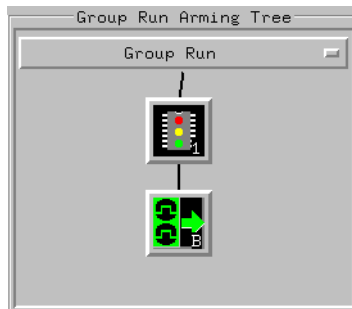
If your processor halts unexpectedly, and you would like to see timing information on your bus prior to the halt, set up this measurement.

The following example shows how to set up an Agilent Technologies 16600/16700 logic analysis system with VisiTrigger. This measurement can also be set up using Agilent Technologies 16600A/16700 logic analysis systems without VisiTrigger, and Agilent Technologies 1660/1670/16500-series logic analysis systems.

NOTE:

If you only need state information leading up to a processor halt, and timing information is not important, use the procedure called “To capture a trace before the processor halts” on page 180. It is much simpler.

- 1 In the Intermodule window, select on the logic analyzer you want to trigger and select the emulation module. A picture (similar to the one shown below) will appear in the intermodule window. This sets the logic analyzer to trigger when the processor halts.

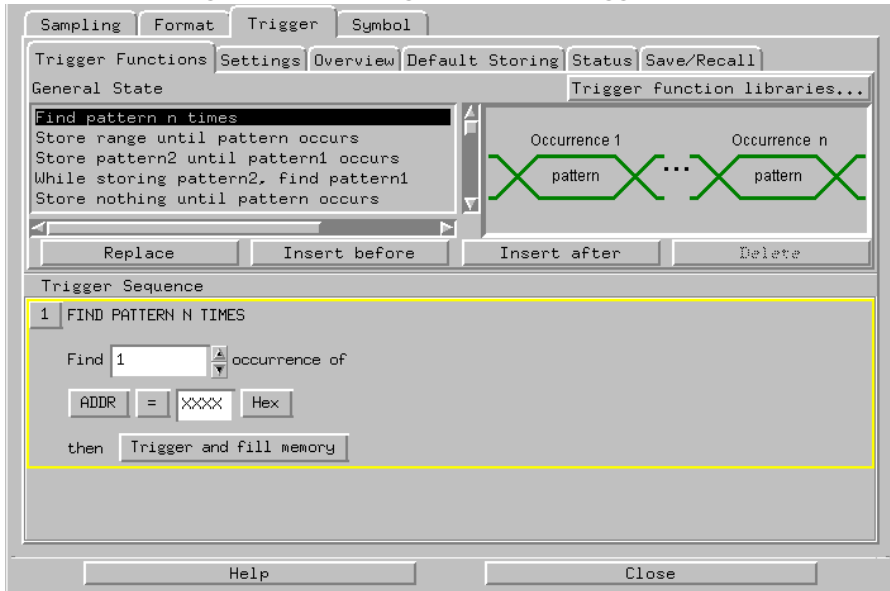


Now continue to step 2.

Chapter 9: Coordinating Logic Analysis with Processor Execution

Triggering the Logic Analyzer from the Emulation Probe

- 2 Set the sampling mode to timing and set the trigger as shown below:



- 3 Set the trigger position to **end**.
- 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.

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 halted.

The logic analyzer will store states until the processor halts.

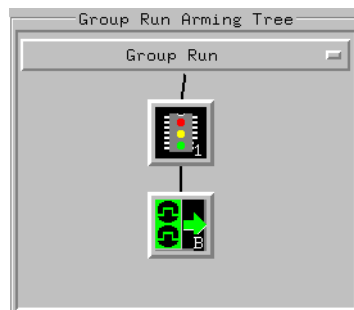
To trigger the analyzer when the processor reaches a breakpoint

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.

NOTE:

If you are only using a state analyzer to capture the processor bus then it will be much simpler to use “Tracing until processor halts” as described on page 180.

- 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 that 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.

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

General-Purpose ASCII (GPA) Symbol
File Format

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 is not one of those listed on page 113, 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

```
[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

```
[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.

Define sections first

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

```
[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

```
[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

```
[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

```
[START ADDRESS]  
address
```

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

Example

```
[START ADDRESS]  
00001000
```

Comments

```
#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.
```

Specifications and Characteristics

Operating Characteristics

The following operating characteristics are not specifications, but are typical operating characteristics for the Agilent Technologies E2498A PowerPC Inverse Assembler.

Operating Characteristics	
Microprocessor Compatibility	PowerPC 740, 745, 750, 755
Microprocessor Clock Speed	70 MHz for the Agilent 1670 and 16554A Logic Analyzers 100 MHz for the Agilent 16600A, 16601A, 16602A, 16603A, 16710/11/12, 16715/16/17/18/19, 16750/51/52, 16550A, 16554D, 16555/56/57, 1660, and 1670D Logic Analyzers
Logic Analyzers Supported	1660A/AS/C/CP/CS, 1670A/D, 16550A, 16554A/55A/56A, 16555D/56D/57D, 16600A, 16601A, 16602A, 16603A, 16710/11/12, 16715/16/17/18/19, 16750/51/52.
Probes Required	Eight 16-channel probes are required for disassembly. Two additional 16-channel probes are available.
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 the 16600-series logic analysis system, the combined setup/hold must be at least 4.5 ns (such as 0/4.5, 1.0/3.5. etc.). For 16550/54/55/56 logic analyzers, the combined setup/hold time must be at least 3.5 ns. For 16557 logic analyzers, the combined setup/hold time must be at least 3.0 ns. For 16710/11/12A logic analyzers, the combined window must be at least 4.0 ns. 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).

Operating Characteristics

Troubleshooting the Logic Analyzer

Chapter 12: Troubleshooting the Logic Analyzer

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, probes, and analysis probes. Otherwise, you may damage circuitry in the analyzer, analysis probe, 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 on the analysis probe interface 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 209 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, board connections, and analysis probe interface connections.
- ❑ Check for bent or damaged pins on the analysis probe.

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 sequence 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 analysis probe. This will allow the logic analyzer to power up. Reconnect logic analyzer cabling after power up.

Analysis Probe Problems

This section lists problems that you might encounter when using an analysis probe. If the solutions suggested here do not correct the problem, you may have a damaged analysis probe. Contact your local Agilent Technologies Sales Office if you need further assistance.

Target system will not boot up

If the target system will not boot up after connecting the analysis probe interface, the microprocessor (if socketed) or the analysis probe interface 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 analysis probe and target system.
 - a** Power up the analyzer and analysis probe.
 - b** Power up the target system.

If you power up the target system before you power up the analysis probe, interface circuitry in the analysis probe may latch up and prevent proper target system operation.

- ❑ Verify that the microprocessor and the analysis probe are properly rotated and aligned, so that the index pin on the microprocessor (pin A1) matches the index pin on the analysis probe interface.
- ❑ Verify that the microprocessor and the analysis probe interface are securely inserted into their respective sockets.
- ❑ Verify that the logic analyzer cables are in the proper sockets of the analysis probe interface and are firmly inserted.

Erratic trace measurements

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

Some analysis probe designs require a full reset to ensure correct configuration.

- ❑ Ensure that your target system meets the timing requirements of the processor with the analysis probe installed.

See “Capacitive loading” in this chapter. While analysis probe 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 by the analysis probe interface, or system lockup in the microprocessor. All analysis probe 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 analysis probe or in your target system. If you follow the suggestions in this section to ensure that you are using the analysis probe and 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 each logic analyzer pod is connected to the correct analysis probe connector.

There is not always a one-to-one correspondence between analyzer pod numbers and analysis probe cable numbers. Analysis Probes must supply address (ADDR), data (DATA), and status (STAT) information to the analyzer in a predefined order. The cable connections for each analysis probe are often altered to support that need. Thus, one analysis probe 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 not change the names of these labels or the bit assignments within the labels. Some analysis probes also require other data labels. See page 126 for more

information.

- ❑ Verify that all microprocessor caches and memory managers have been disabled.

NOTE:

This is only necessary if cache-on trace reconstruction and cache-on execution tracker are not used.

In most cases, if the microprocessor caches and memory managers remain enabled you should still get inverse assembly. It may be incorrect because a portion of the execution trace was not visible to the logic analyzer.

To determine if a cache is on or off, examine the most significant bit of the ICCST register (for the instruction cache) or the DCCST register (for data cache). If this bit is 1, the cache is on; if the bit is 0, the cache is off.

See “To enable/disable the instruction cache on the PPC7XX” on page 104 or “To disable the instruction cache on the PPC7XX” on page 139.

- ❑ Verify that storage qualification has not excluded storage of all the needed opcodes and operands.
- ❑ Verify that the data format is big-endian.

Unlike most processors, the MPC8XX can run in either little-endian or big-endian mode. The inverse assembler can only decode data in big-endian format. To verify the format of the data, the MSR or Machine State Register must be examined. The least significant bit (bit 0 according to Motorola convention, or bit 31 according to IBM convention) indicates the mode of the processor. A value of 1 indicates little-endian mode. A value of 0 indicates big-endian mode.

Inverse assembler will not load or run

You need to 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 “To install software from CD-ROM (16600/700-series logic analysis systems)” on page 50 to reinstall the software if you have renamed or deleted the inverse assembler.

If the inverse assembler cannot determine cycle sizes

Agilent Technologies 16600A and 16700 logic analysis systems only

The processor sometimes fails to put correct information on the bus.

- In the Listing window, select the Preferences menu.
Enter each memory bank address into the appropriate field.

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.

Analysis Probe Messages

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

“... Inverse Assembler Not Found”

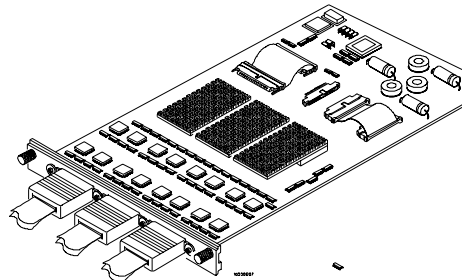
This error occurs if you rename or delete the inverse assembler file that is attached to the configuration file.

Ensure that the inverse assembler file is not renamed or deleted, and that it is located in the correct directory:

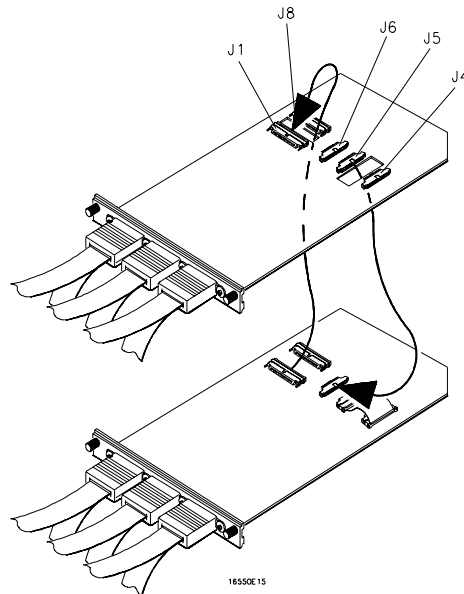
- For Agilent Technologies 16600A/700-series logic analysis systems it should be in `/logic/ia`.
- For other logic analyzers it should be in the same directory as the configuration file.

“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

See the *Agilent Technologies 16550A 100-MHz State/500-MHz Timing Logic Analyzer Service Guide* for more information.

“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 when you load the configuration file. 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 hard disk” on page 76 or “To load configuration files and the inverse assembler—1660/1670/16500B/C-series logic analyzers” on page 126.

“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 analysis probe interface. See Chapter 4, “Probing the Target System,” beginning on page 53, 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.

Returning Parts to Agilent Technologies for Service

The repair strategy for this emulation solution is board replacement.

Exchange assemblies are available when a repairable assembly is returned to Agilent Technologies. These assemblies have been set up on the “Exchange Assembly” program. This lets you exchange a faulty assembly with one that has been repaired, calibrated, and performance verified by the factory. The cost is significantly less than that of a new assembly.

To return a part to Agilent Technologies

- 1** Follow the procedures in this chapter to make sure that the problem is caused by a hardware failure, not by configuration or cabling problems.
- 2** In the U.S., call 1-800-403-0801. Outside the U.S., call your nearest Agilent Technologies sales office. Ask them for the address of the nearest Agilent Technologies service center.
- 3** Package the part and send it to the Agilent Technologies service center.

Keep any parts which you know are working. For example, if only the target interface module is broken, keep the emulation module and cables.

- 4** When the part has been replaced, it will be sent back to you.

The unit returned to you will have the same serial number as the unit you sent to Agilent Technologies.

The Agilent Technologies service center can also troubleshoot the hardware and replace the failed part. To do this, send your entire measurement system to the service center, including the logic analysis system, analysis probe, and cables.

In some parts of the world, on-site repair service is available. Ask the Agilent Technologies sales or service representative for details.

To obtain replacement parts

The following table lists some parts that may be replaced if they are damaged or lost. Contact your nearest Agilent Technologies Sales Office for further information.

Analysis Probe Replaceable Parts

Agilent Part Number	Description
E5346A	High-density Cable
E2476-66502	Analysis Probe Circuit Board
E2476-87602	BGA Extender (one pre-installed on the analysis probe board, and one supplied with the product for customer installation. Total of 2)
E2476-87606	Double Header, 19x19
E2476-87607	BGA Carrier
E5355A	BGA Probe Kit

Cleaning the Instrument

If the instrument requires cleaning:

- 1** Disconnect power from the instrument.
- 2** Clean the instrument using a soft cloth that has been moistened in a mixture of mild detergent and water.

Make sure that the instrument is completely dry before reconnecting it to a power source.

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

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, 162
- * symbol in listing, 144, 163
- ? symbol in listing, 144, 163

A

- access to source code files, 168
- active low signal, 33
- adapter, 29
- ADDR label, 108
- ADDR label, modifying, 81, 131
- address
 - triggering on a specific address, 86, 138
- addresses
 - branch target, 144, 162
 - mask, 93
 - offset, 117, 156
 - PC label, 115
 - type, 82, 132
- Agilent 1252-7431, 29
- Agilent E5346-44701 shroud, 29
- Agilent E5346-60002 adapter, 29
- Agilent Technologies B4620B
 - source correlation tool set, 152
- Agilent Technologies E5346A high-density termination adapter
 - cable, 29
- analysis mode
 - state, 169
- analysis probe
 - definition, 221
 - equipment supplied, 22
 - inverse assembly, 88, 139
 - modes of analysis, 121
 - modes of operation, 119, 129
 - operating characteristics, 200
 - power on/power off sequence, 46
 - storage qualification, 86, 137, 157

- analysis probe problems, 208
 - erratic trace measurements, 209
 - target system will not boot, 208
- analyzer problems, 205
 - capacitive loading, 209
 - intermittent data errors, 205
 - unwanted triggers, 206
- ASCII format (GPA), 190
- assistant
 - See setup assistant
- AT status bit, 82, 132

B

- b prefix, 143
- B4620B source correlation tool set,
 - 2, 3, 49
- background debug monitor, 221
- BB status bit, 82, 132
- bits
 - labels, 81, 131
 - LSB and MSB, 81, 131
 - STAT, 82, 132
- blank pins, 33
- board space, 29
- branch exception disassembly, 91
- branch instructions, 144, 162
- branches, displaying, 164
- breakpoints
 - tracing until, 187

C

- cables
 - replacing, 219
- cache
 - disabling, 104, 139
 - enabling, 104
 - trace problems and, 211
- cache-on execution tracker
 - performance hints, 89
- cache-on trace exception routine,
 - 89

- cache-on trace reconstruction, 88,
 - 92
 - caches
 - enabling and disabling, 104, 139
 - captured data, source code
 - associated with, 166
 - cards
 - See logic analyzers
 - center inline pins, 31, 33
 - checklist, setup, 19
 - chip selects, 212
 - cleaning, 220
 - clocks
 - logic analyzer, 119, 129
 - qualification, 86, 137, 157
 - qualified, and emulator, 178
 - slow, 186, 188, 216
 - colors, 164
 - comments, in GPA files, 198
 - compilers, 113
 - configuration
 - checklist, 19
 - logic analyzer, 123
 - logic analyzers, 81, 125, 131
 - configuration file
 - loading, 126
 - configuration files, 49, 78, 109,
 - 127, 152
 - installing, 73
 - loading, 125
- Configuring the 16600A/16700A-series Logic Analyzer, 73
- configuring the logic analyzer, 75
- connecting-to target, 28
- connection
 - analysis probe, 53
 - analysis probe to logic analyzer,
 - 54
 - setup checklist, 19
- connection notes
 - recommended configuration, 33
- connector board, 221
- creating GPA symbol files, 190

custom probing
 designing connectors, 53
cycle size, inverse assembler, 212

D

d prefix, 143
DATA label, modifying, 81, 131
data, displaying, 160
debug mode, 221
debug port, 221
 definition, 221
decoding
 exception, 102
 simplified mnemonic, 99
deep memory logic analyzer, 167
Diab Data
 compiler, 114
directories
 configuration files, 76
disassembly, branch exception, 91
display filtering, 164
display timing analysis mode data,
 169
displaying state data, 160

E

E9503A emulation solution, 2
elastomeric probe adapter
 definition, 221
Emulation Control Interface, 49
 when to use, 172
emulation migration
 definition, 221
emulation module, 2, 3
 definition, 221
 firmware, 49
 target system design, 44
emulation probe
 definition, 221
emulation solution, 2
enhanced inverse assembler
 logic analyzer requirements, 25

equipment required, 22
equipment supplied, 22
 analysis probe, 22
error messages
 inverse assembler, 214
Ethernet networks, 168
examples, measurement, 173
exception decoding, 102
extended mnemonics, 147
extender, 221

F

filtering, display, 164
firmware
 emulation module, 49
flexible adapter
 definition, 222
floppy disks
 duplicating, 126
flowchart, setup, 19
format menu, 81, 131
ftp, 168
FUNCTIONS in GPA format, 195

G

gateway address
 definition, 222
General-Purpose ASCII (GPA)
 symbol file, 108
General-Purpose ASCII format, 190
 address format, 190
 comments, 198
 FUNCTIONS, 195
 record format summary, 192
 record headers, 190
 SECTIONS, 194
 simple form, 190
 SOURCE LINES, 197
 START ADDRESS, 198
 VARIABLES, 196
general-purpose flexible adapter
 definition, 222

ground returns, 31

H

high-density adapter cable
 definition, 222
high-density connector
 mechanical specifications, 30
 pin assignment, 30
high-density connectors, 29
high-density termination adapter
 definition, 222
high-level source code, 166
HRESET signal, 44

I

illegal opcode, 162
information sources, 6
inline pins, 31
installation, software, 45
instruction cache
 See cache
instruction decoding, 99, 147
intermodule measurement
 creating, 176
intermodule measurement
 problems, 213
 an event wasn't captured, 213
 analyzer doesn't stop, 178
inverse assembler, 2, 49, 167
 configuration file names, 55
 definition, 222
 loading, 93
 loading files, 76, 77, 126
 operating modes, 92
 output format, 143, 162
 overview, 121
 preferences, 94
 probing for "IA-only", 53
 requirements for, 88, 139
 requirements for enhanced, 25

-
- inverse assembler problems, 210
 - failure to load or run, 212
 - incorrect inverse assembly, 210
 - no inverse assembly, 210
 - inverse assembly, 88, 139
 - cache-on, 92
 - displays, 172
 - Pods required, 24
 - traditional, 88, 92
 - IP address
 - definition, 222
 - J**
 - jumper, definition, 222
 - L**
 - labels
 - definition, 222
 - LAN protocols, 168
 - LAN system administrators, 168
 - link-level address
 - definition, 223
 - listing
 - incorrect, 210
 - Listing display window, 160
 - Listing menu, 141
 - listing windows, 172
 - Load menu, 93
 - loading configurations
 - logic analyzer, 125
 - loading configurations, vs.
 - installing, 49, 76
 - loading object file symbols, 108
 - loading symbol information, 108
 - logic analyzer
 - configuration files, 78, 109, 127
 - deep memory, 167
 - trigger setup, 154
 - triggers, 152
 - logic analyzers
 - 16550A connections, 69
 - 16554/55/56/57 connections, 66, 67
 - 16600A and 16700A-series, 20
 - 16600A connections, 65
 - 16601A connections, 64
 - 16602A connections, 63
 - 16603A connections, 62
 - 1660A/C connections, 72
 - 1670A/D connections, 71
 - 16710/11/12A connections, 60
 - 16715/16/17/18/19A connections, 57
 - clocking, 86, 137, 157
 - configuration, 81, 131
 - configuring, 76, 77, 126
 - loading configuration files, 125
 - software version requirements, 25
 - storage qualification, 86, 137, 157
 - supported, 24
 - logic analyzers-compatible, 22
 - LSB, 81, 131, 141
 - M**
 - mainframe logic analyzer
 - definition, 223
 - male-to-male header
 - definition, 223
 - measurement examples, 173
 - mechanical specifications
 - high-density connector, 30
 - memory banks, 93
 - MICTOR (Matched Impedance ConnecTOR) connectors, 29
 - MICTOR connector, definition, 223
 - mnemonics, 99, 147
 - modes
 - analysis, 121
 - operating, 119, 129
 - monitor, background debug, 221
 - MSB, 81, 131, 141
 - N**
 - network access to source files, 168
 - NFS client/server, 168
 - no-connect, 33
 - O**
 - object file symbols, 152
 - loading, 108
 - offset, address, 117, 156
 - online configuration help, 20
 - operating modes, inverse
 - assembler, 92
 - operating system, 50
 - Options menu, 93
 - P**
 - parts supplied, 22
 - passive probing, 53
 - PC label, 115
 - personality files, 49
 - plastic shroud, 29
 - Pods, logic analyzer, 223
 - power on/power off sequence, 46
 - predefined symbols, 109
 - preferences, inverse assembler, 94
 - ? prefix and suffix, 143
 - preprocessor
 - See analysis probe
 - preprocessor interface
 - See *analysis probe*
 - problems
 - analysis probe, 203
 - processor support package, 49, 50
 - processors supported, 2
 - Q**
 - question mark, 143
-

R

R/W status bit, 82, 132
recommended circuit board
 routing, 30
recommended configuration
 connection notes, 33
recommended connector layout,
 33
record format, General-Purpose
 ASCII, 192
record headers, 190
references, 6
registers
 listing format, 143, 162
repair
 analysis probe, 218
requirements
 target system, 44
run control tool
 See emulation control interface

S

search path, source code, 168
Section Format, 190
SECTIONS in GPA format, 194
service, how to obtain, 218
Setting Up the Logic Analysis
 System, 45
Setup Assistant, 20, 49
setup assistant, 20
 definition, 223
setup checklist, 19
Setup window, 109
show cycles
 disassembly, 88
shroud, 29, 30
signal ground returns, 31, 33
signal integrity, 29
skid, reducing, 177
slow clock message, 186, 188

software
 installing, 45
 list of installed, 78
 requirements, 25
software addresses, 115, 167
software requirements, 25
solution
 at a glance, 2
 definition, 223
solutions
 description of, 2
source code, 166
 associated with captured data,
 166
 displays, 172
source correlation, 23
 data display, 106, 142
source correlation tool set, 106,
 166, 168
source files
 network access to, 168
 search path, 168
 version control, 168
SOURCE LINES in GPA format,
 197
SRESET signal, 44
stand-alone logic analyzer
 definition, 224
START ADDRESS in GPA format,
 198
STAT
 encoding, 82, 132
 label, 82, 132
 modifying, 81, 131
state analysis, 224
 definition, 224
state mode, 169
state-per-ack, mode of operation,
 119, 129
state-per-clock mode, 86
state-per-transfer, mode of
 operation, 119, 129
status bits, 82, 132

status encoding, 82, 132
storage qualification, 86, 137
STS status bit, 82, 132
subnet mask
 definition, 224
? prefix and suffix, 143
support shroud, 30
supported logic analyzers, 24
surface mount connector, 30
SW_ADDR label, 162, 167
symbol file, 166
symbol files
 creating, 190
symbol information
 loading, 108
symbols
 definition, 224
 in analyzer, 156
 predefined, 84, 135
 user-defined, 110
Symbols tab, 109
symbols, displaying, 161
synchronous mode, 119, 129
system administrators, 168

T

TA status bit, 82, 132
target board adapter
 definition, 224
target control port, 224
target interface module (TIM)
 definition, 224
target system
 boot failure, 208
 power sequence, 46
 requirements for emulation, 44
TCP/IP protocol, 168
TEA status bit, 82, 132
telnet, 168
timing analysis, 224
 definition, 224

timing analysis mode
 data, displaying, 169
timing, mode of operation, 120, 130
trace
 erratic, 209
 missing display, 207
trace reconstruction, cache-on, 88
transition board
 definition, 225
trigger
 dialog, 86
 emulation module, 174
 on address, 86
 on break, 181
 sequence, 153
 source code, 116, 156
 specification, 86, 137
 unwanted, 206
trigger function, 154
Trigger menu, 86, 137
trigger sequence, 155
trigger specification
 definition, 225
trigger tool, 108
troubleshooting
 analysis probe, 203
TS status bit, 82, 132
TSIZ status bit, 82, 132

U

Undefined Opcode, 143
unknown opcode, 162
user defined signals, 33
User Defined Symbols tab, 109
user-defined symbols, 109

V

VARIABLES in GPA format, 196
version control, source file, 168
versions
 logic analyzer software, 25

W

Waveform display, 169
web sites
 Agilent logic analyzers, 6
 See Also under debugger names
wizard
 See setup assistant

X

X-Window client/server, 168

Y

Y-cable, 29

© 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.

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 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 *Solutions for the PowerPC PPC7XX User's Guide*.

Publication number
E2498-97006, September 2000
Printed in USA.

The information in this manual previously appeared in:
E2498-97005 May 2000
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.

Reflection 1 is a U.S. trademark of Walker, Richer & Quinn, Inc.

UNIX is a registered trademark of the Open Group.

Windows and MS Windows are U.S. registered trademarks of Microsoft Corp.

X/Open is a registered trademark, and the X device is a trademark of X/Open Company Ltd. in the UK and other countries.