User's Guide

Publication number E8135-97005

October 2002

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

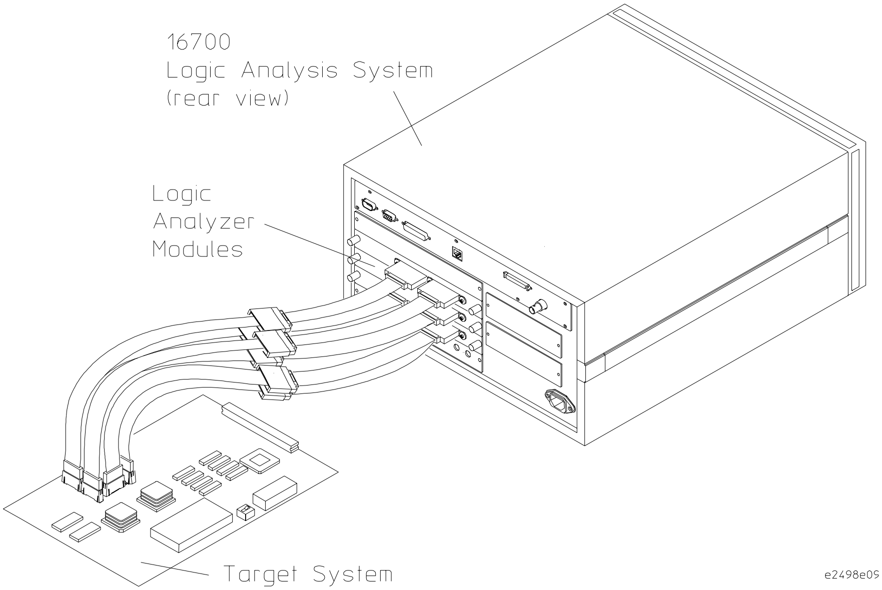
Logic Analysis Support for the Motorola MPC7410/4X/5X

Logic Analysis Support for the MPC7410/4X/5X — At a Glance

This book documents the MPC7410/4X/5X inverse assemblers.

Inverse Assembler Software

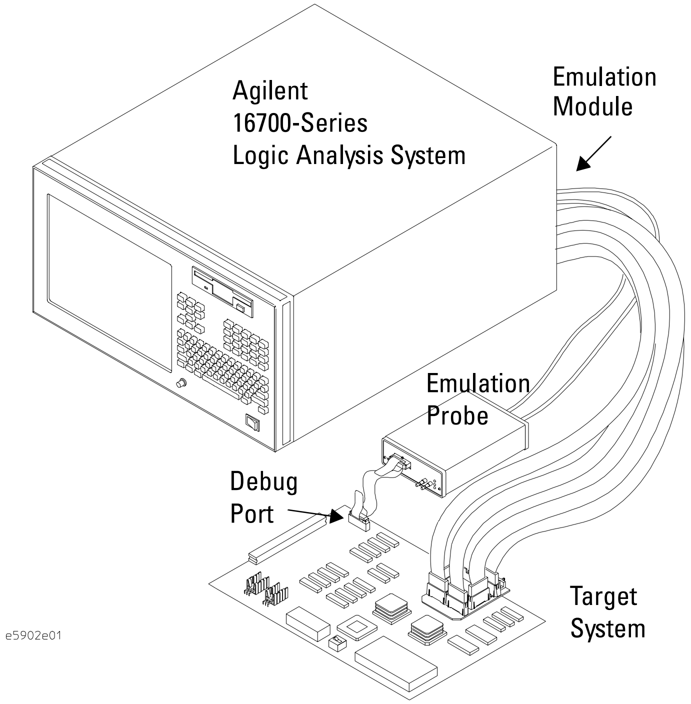
The Agilent Technologies E8170B and E8135A inverse assemblers, in conjunction with a Hewlett-Packard or Agilent Technologies logic analyzer, allow you to view MPC7410/4X/5X assembly instructions that are executing in your target system. The inverse assemblers can be configured to work with signals that are available for probing.



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

The inverse assembler package model number is Agilent Technologies E9614A Option 001. The MPC7410/4X/5X solution uses two inverse assemblers, one for 60x bus mode, and one for MPXbus mode. The 60x bus inverse assembler is identified as E8170B in the Setup Assistant, and the MPXbus inverse assembler is identified as E8135A in the Setup Assistant.

If You Purchased an Emulation Probe/Module



Emulation Probe and Emulation Module

The emulation probe connects to the emulation module and a debug port on your target system. It lets you use the target processor's built-in background debugging features, including run control and accessing registers and memory. A high-level source debugger can use the emulation probe to debug code running on the target system. The emulation module provides improved probe/analyzer cross triggering, and an easy way to set the emulation probe's IP address.

Information about using the logic analysis system with the emulation probe/module can be found in Chapter 8, “Coordinating Logic Analysis with Processor Execution,” beginning on page 151 of *this* manual.

The *Emulation for the MPC74XX User’s Guide* describes setting up and using the emulation probe and emulation module.

Source Correlation Tool Set

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

Additional Information Sources

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

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

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

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

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

In This Book

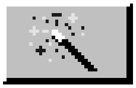
This book documents the following products:

Inverse Assembler Software		
Processors supported	Product ordered	Includes
MPC7410 MPC7440 MPC7441 MPC7445 MPC7450 MPC7451 MPC7455	E9614A Option 001	E8170B inverse assembler E8135A inverse assembler

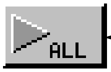
Tips To Save You Time

Use the Setup Assistant

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

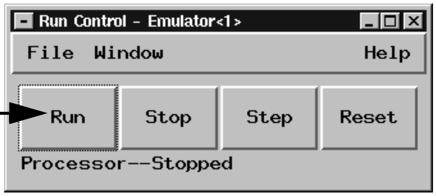


Use the appropriate Run button



Click here to start a measurement.

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



Processor--Stopped

Contents

4 Connecting the Logic Analyzer to the Target System 75

5 Configuring the Logic Analyzer 85

1

Overview

Setup Checklist

Follow these steps to connect your equipment:

If you need to install an emulation module in an Agilent Technologies 16700-series logic analysis system, see your emulation manual.

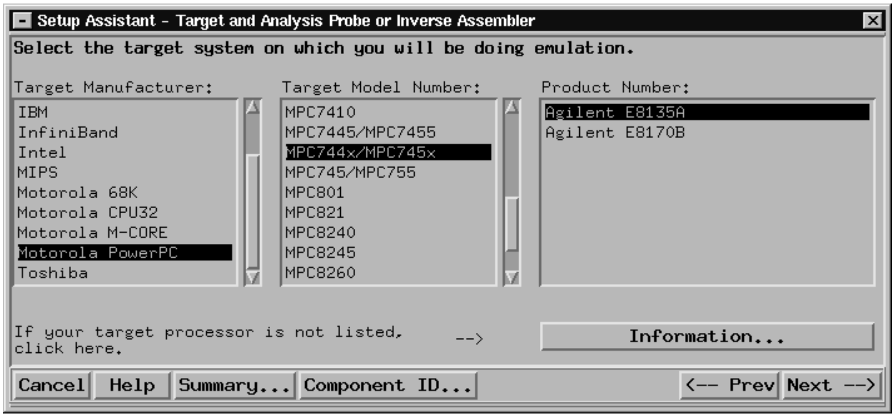
- Install the software. See page 72.
- Use the Setup Assistant to help you connect and configure your logic analysis system. See page 17.

Setup Assistant

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

This menu-driven tool will guide you through the procedures for connecting the logic analyzer to the target system.

Start the Setup Assistant by selecting in the system window. The Setup Assistant dialog will be displayed.



Setup Assistant - Target and Analysis Probe or Inverse Assembler

Select the target system on which you will be doing emulation.

Target Manufacturer: Motorola PowerPC

Target Model Number: MPC744x/MPC745x

Product Number: Agilent E8135A

If your target processor is not listed, click here. -->

The E8135A inverse assembler supports the MPX bus. The E8170B inverse assembler supports the 60x bus.

Use the MPC7410 selection for the MPC7410 processor. Use the MPC744x/MPC745x selection for MPC7440, MPC7441, MPC7450, and MPC7451 processors. Use the MPC7445/MPC7455 selection for MPC7445 and MPC7455 processors.

If you ordered this inverse assembler software with your 16700-series logic analysis system, the logic analysis system has the latest software installed, including support for this product. If you received this product after you received your logic analysis system, see “Installing and Loading Software” on page 72.

Equipment Used with Inverse Assembler Software

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

Equipment supplied

The E8170B and E8135A inverse assembler software packages consist of the following:

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

Minimum equipment required

For state and timing analysis of an MPC7410/4X/5X target system, you need all of the following items.

- The E8170B and E8135A MPC7410/4X/5X Inverse Assemblers.
- The appropriate connectors on the target system. Chapter 2, “Preparing the Target System,” contains information on designing the appropriate connectors into the target system. You could use General Purpose probes instead of connectors.
- One of the logic analyzers listed on page 19.

Additional equipment supported

B4620B Source Correlation Tool Set

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

Compatible Logic Analyzers

The following table lists the logic analyzers supported by the E8170B and E8135A inverse assembler software.

The E8170B (60x bus) inverse assembler requires eight logic analyzer pods (136 channels) for traditional inverse assembly with 64-bit data. Inverse assembly can also be done with four pods (no data). See “Connecting the Logic Analyzer to the Target System” on page 75 for details.

The E8135A (MPXbus) inverse assembler requires eight logic analyzer pods (136 channels) for traditional inverse assembly of a *single* MPC7410/4X/5X processor with 64-bit data. Inverse assembly on a *single* processor can also be done with four pods (no data). See “Connecting the Logic Analyzer to the Target System” on page 75 for details.

The E8135A (MPXbus) inverse assembler requires 10 logic analyzer pods (180 channels) for inverse assembly of *multiple* MPC7410/4X/5X processors.

The following tables show which logic analyzers are supported for use with the inverse assemblers.

Logic Analyzers Supported - MPC7410/4X/5X

Logic Analyzer	Number of Cards	Channel Count
16756A	(1 to 5 cards)	68/card
16755A	(1 to 5 cards)	68/card
16754A	(1 to 5 cards)	68/card
16753A	(1 to 5 cards)	68/card
16752A	(1 to 5 cards)	68/card
16751A	(1 to 5 cards)	68/card
16750A	(1 to 5 cards)	68/card
16742A	(1 to 5 cards)	68/card
16741A	(1 to 5 cards)	68/card
16740A	(1 to 5 cards)	68/card
16719A	(1 to 5 cards)	68/card
16718A	(1 to 5 cards)	68/card
16717A	(1 to 5 cards)	68/card
16716A	(1 to 5 cards)	68/card
16715A	(1 to 5 cards)	68/card
16557D	(1 to 5 cards)	68/card

Related Products

If you ordered an emulation probe and an emulation module, you received the *Emulation for the MPC74XX User's Guide*.

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

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

2

Preparing the Target System

Target System Requirements

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

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

The following sections contain information on designing your target system connections using these approaches. For signal-to-connector mappings that show which signals must go to which logic analyzer connectors, see page 30 (for the MPC7410) or page 49 (for the MPC744X/5X).

Direct Probing with General Purpose (GP) Probes

If you are using General Purpose probes, connect the individual probes to the signals according to the tables beginning on page 30 (for the MPC7410) or page 49 (for the MPC744X/5X).

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

Designing and Using Built-in Connectors

You can design analyzer-compatible connectors into the target board, and connect the logic analyzer cables to these connectors according to the tables beginning on page 30 (for the MPC7410) or page 49 (for the MPC744X/5X).

The primary concerns when using built-in connectors are:

- The board surface area required by the connectors
- Ensuring that the logic analyzer connection is properly terminated

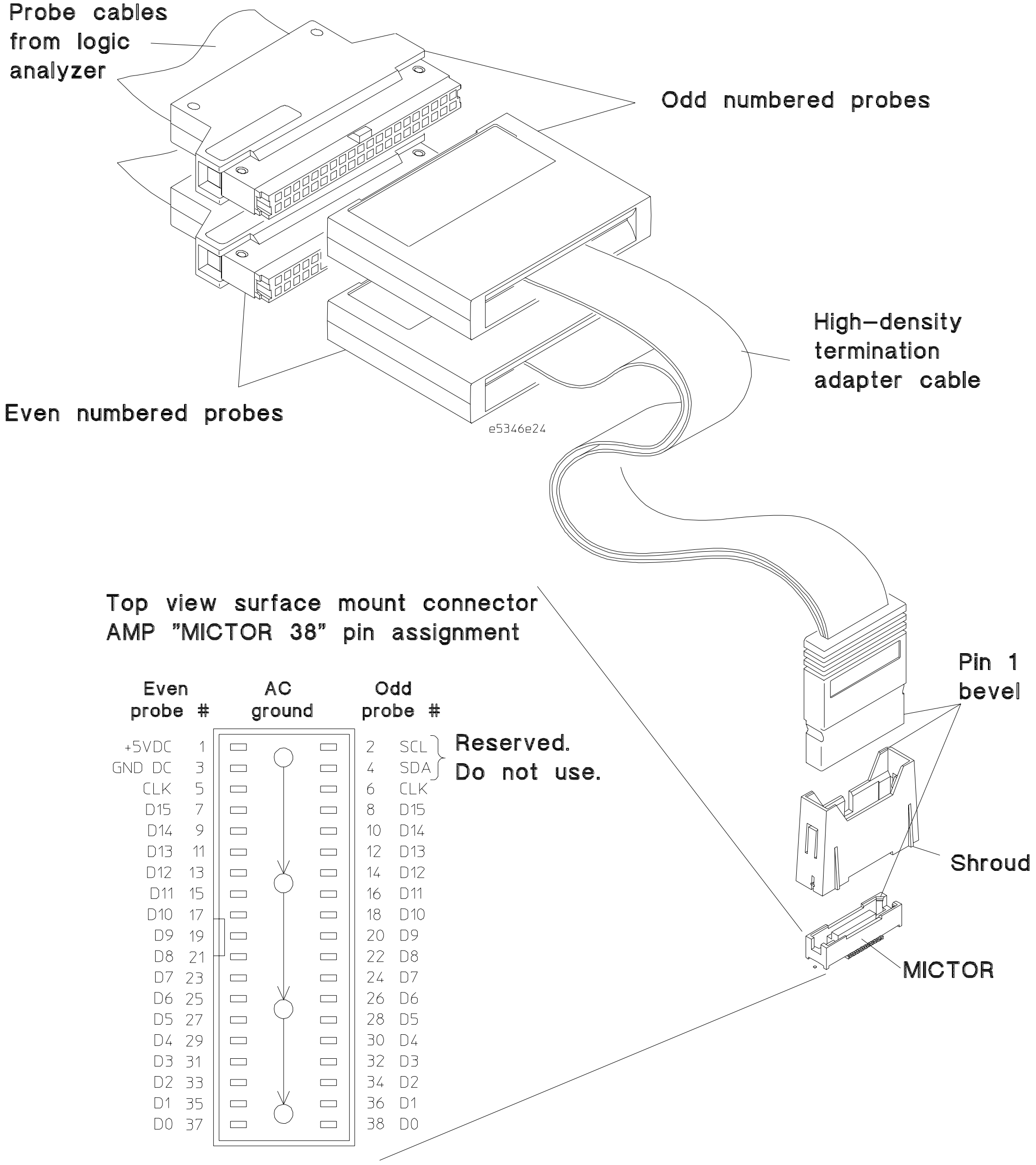
- Ensuring that the microprocessor pins connect to the proper logic analyzer probes.

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

Part Numbers for Built-in Connectors and Cables

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

Connectors, Shroud, and High-density Termination Cables


Probe cables from logic analyzer
Odd numbered probes
High–density termination adapter cable
Even numbered probes
e5346e24
Pin 1 bevel
Shroud
MICTOR

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

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

SCL, SDA: Reserved. Do not use.

AMP MICTOR 38 connectors

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

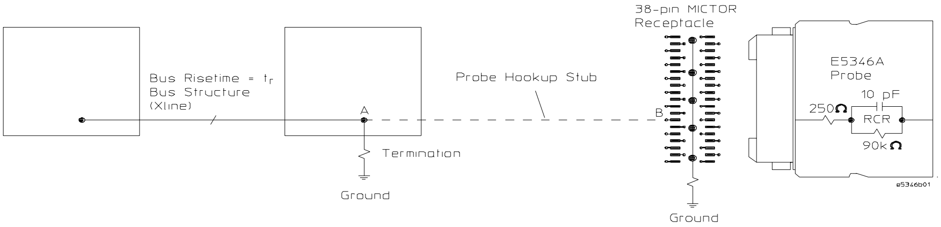
The figure on the previous page shows the pinout for a MICTOR 38 connector. Refer to the tables following page 30 or page 49 which show the microprocessor signals which should be connected to each pin. Note that the +5V pin (pin 1) is used to supply power from the logic analyzer to any active devices on an interface board. In most instances, this pin should not be used. Refer to AMP MICTOR Application Specification 114-11004 for guidelines on soldering the MICTOR connectors.

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

Design Considerations

The 2x19 header must be close enough to the target signal so that the stub length created is less than $^1/_5$ the t_r (bus risetime, see figure below). For PC board material, (er = 4.9) and Z_o in the range of 50 - 80Ω, use a propagation delay of 160 ps/inch of stub.

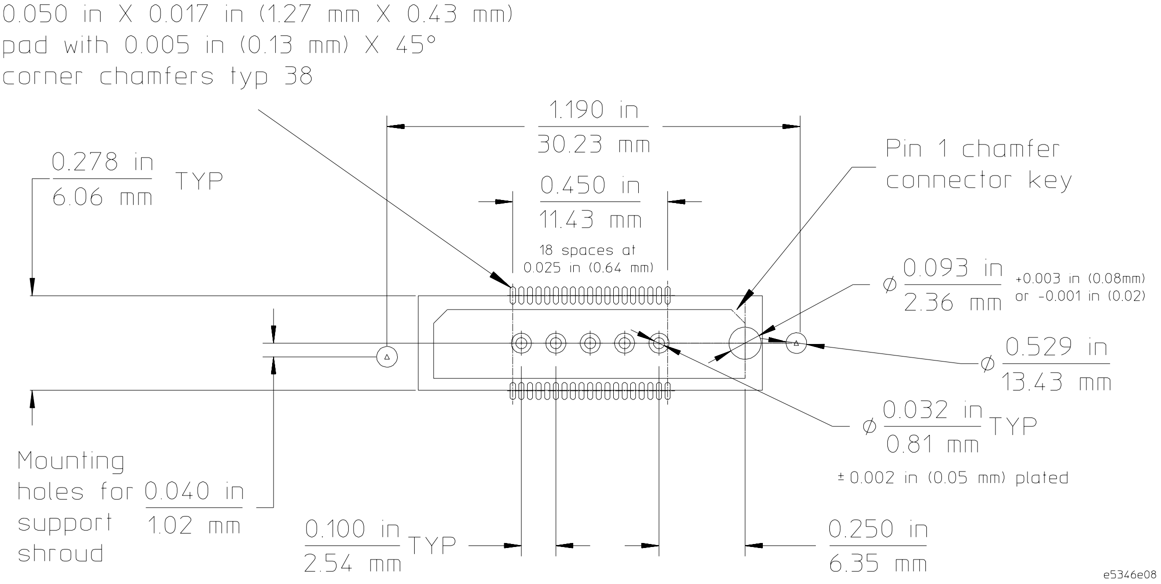
Each probed signal line must be able to supply a minimum of 600 mV to the probe tip and handle a minimum of 90 kΩ shunted by 10 pF. The maximum input voltage to the logic analyzer is ±40V peak.



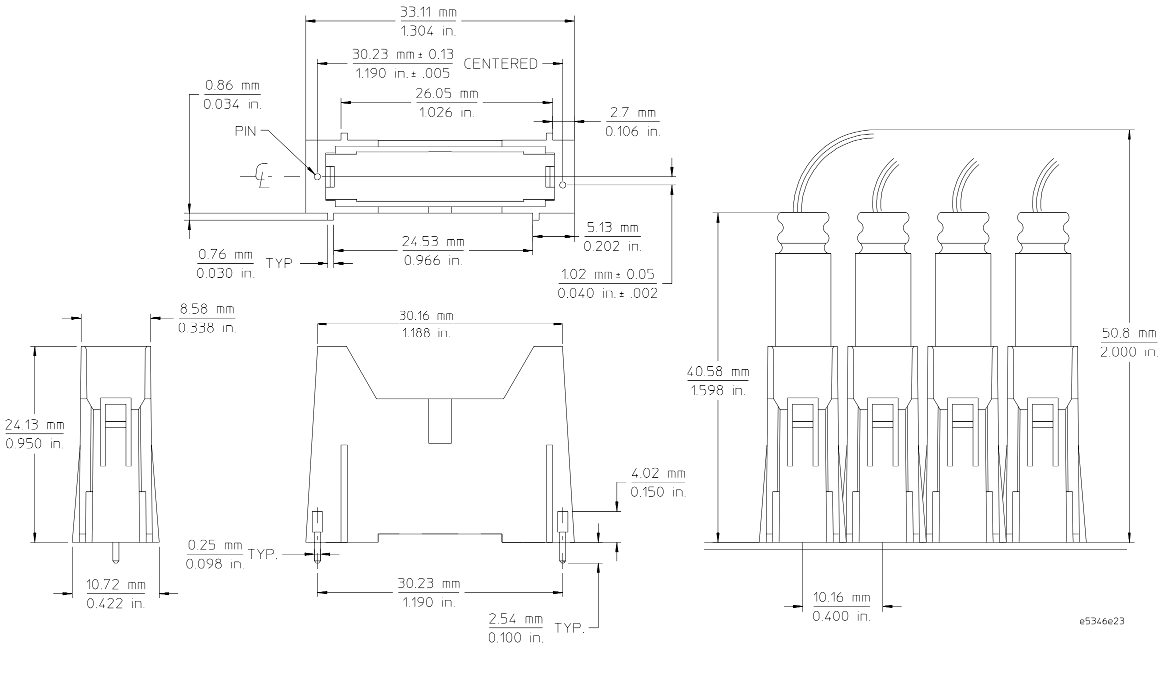
Support shroud

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

Support Shroud Mechanical Information



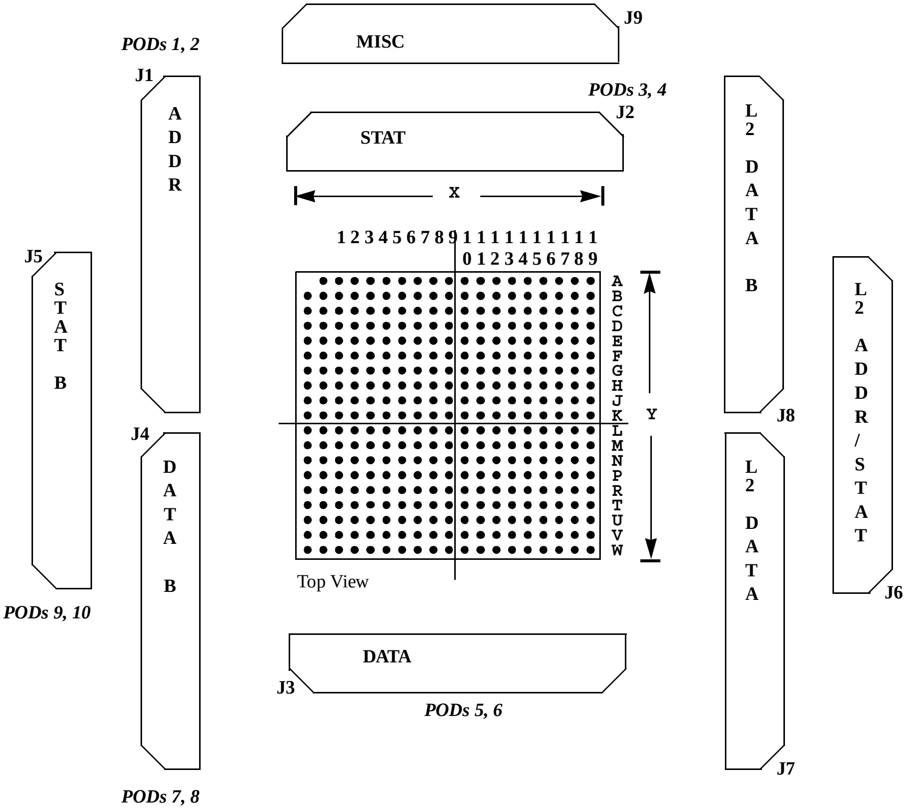
MICTOR 38 Connector Mechanical Information



Recommended Connector Layout—MPC7410

The following MICTOR placement for the MPC7410 is recommended to minimize trace lengths from the BGA to the connectors. Due to the high bus speeds on the MPC7410, even small trace lengths can affect signal integrity.

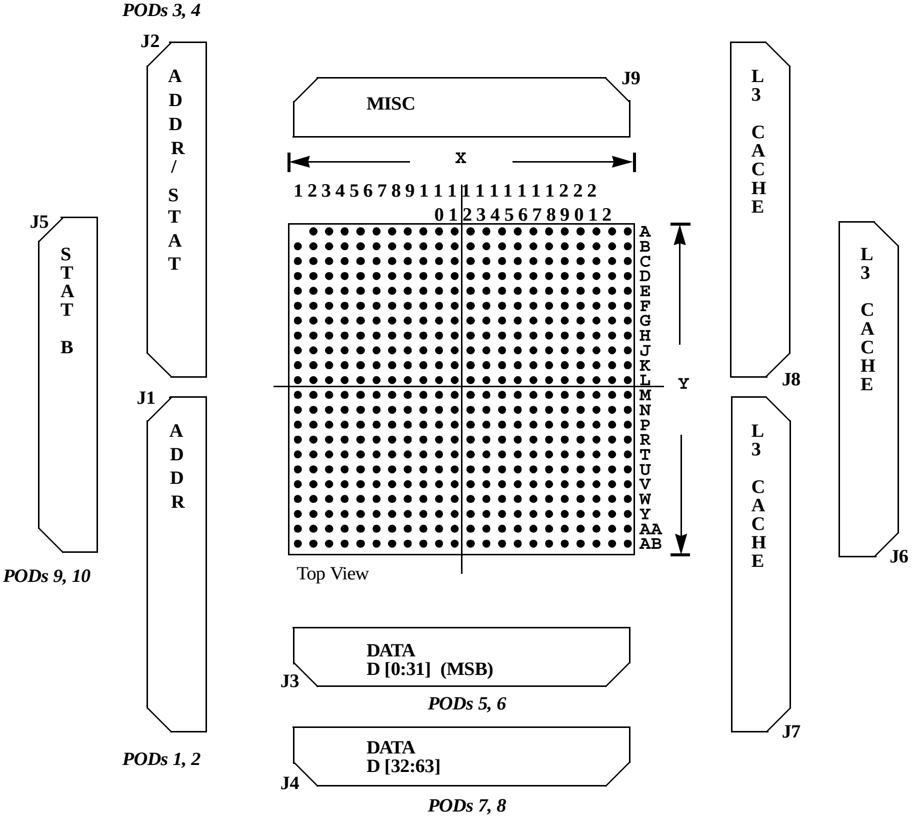
Recommended Connector Layout for the MPC7410



Recommended Connector Layout—MPC744X/5X

The following MICTOR placement for the MPC744X/5X is recommended to minimize trace lengths from the BGA to the connectors. Due to the high bus speeds on the MPC744X/5X, even small trace lengths can affect signal integrity.

Recommended Connector Layout for the MPC744X/5X



Signal-to-Connector Mapping—MPC7410

The following tables show the electrical signal-to-connector mapping required by the E8170B and E8135A inverse assemblers.

- For single processor inverse assembly, MICTOR connectors J1, J2, J3, and J4 are required.
- For multiprocessor inverse assembly using the MPXbus, J5 is also required.
- In order to see L2 cache data, MICTOR connectors J6, J7, and J8 are also required.

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.

NOTE: When using the MPXbus mode with the MPC7410/4X/5X, the inverse assembler requires an ODT signal from the system arbiter. **The ODT signal must be designed into your system arbiter.** The ODT signal is used to initialize the inverse assembler so that it can match address tenures to their respective data tenures. Please see “Using the MPXbus Tool” on page 98 for details.

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J1 Odd

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1	H9	SYSCLK	STAT
8	15	G5	A[16]	ADDR
10	14	L4	A[17]	ADDR
12	13	G4	A[18]	ADDR
14	12	J4	A[19]	ADDR
16	11	H7	A[20]	ADDR
18	10	E1	A[21]	ADDR
20	9	G2	A[22]	ADDR
22	8	F3	A[23]	ADDR
24	7	J7	A[24]	ADDR
26	6	M3	A[25]	ADDR
28	5	H3	A[26]	ADDR
30	4	J2	A[27]	ADDR
32	3	J6	A[28]	ADDR
34	2	K3	A[29]	ADDR
36	1	K2	A[30]	ADDR
38	0	L2	A[31]	ADDR

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J1 Even

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1	K7	TS#	STAT
7	15	A13	A[0]	ADDR
9	14	D2	A[1]	ADDR
11	13	H11	A[2]	ADDR
13	12	C1	A[3]	ADDR
15	11	B13	A[4]	ADDR
17	10	F2	A[5]	ADDR
19	9	C13	A[6]	ADDR
21	8	E5	A[7]	ADDR
23	7	D13	A[8]	ADDR
25	6	G7	A[9]	ADDR
27	5	F12	A[10]	ADDR
29	4	G3	A[11]	ADDR
31	3	G6	A[12]	ADDR
33	2	H2	A[13]	ADDR
35	1	E2	A[14]	ADDR
37	0	L3	A[15]	ADDR

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J2 Odd

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1	F1	TA#	STAT
8	15	A9	TSIZ[0]	STAT
10	14	B9	TSIZ[1]	STAT
12	13	C9	TSIZ[2]	STAT
14	12	C10	TT[0]	STAT
16	11	D11	TT[1]	STAT
18	10	B12	TT[2]	STAT
20	9	C12	TT[3]	STAT
22	8	F11	TT[4]	STAT
24	7	C2	CI#	
26	6	B3	SHD0#	
28	5	B4	SHD1#	
30	4	B1	GBL#	
32	3	L7	ABB# / AMON(0)#	
34	2	K5	DBB# / DMON(0)#	
36	1	E10	SRESET#	
38	0	B6	HRESET#	

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J2 Even

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1	N3	AACK#	STAT
7	15	E7	BR# - (Processor 0)	STAT
9	14	H1	BG# - (Processor 0)	STAT
11	13	K1	DBG# - (Processor 0)	STAT
13	12	B5	HIT# - (Processor 0)	STAT
15	11	K9	DRDY# - (Processor 0)	STAT
17	10	D1	DTI[0] - (Processor 0) / DBWO#	STAT
19	9	H6	DTI[1] - (Processor 0)	STAT
21	8	G1	DTI[2] - (Processor 0)	STAT
23	7	**Arbiter**	**ODT[0] - (System)**	STAT
25	6	**Arbiter**	**ODT[1] - (System)**	STAT
27	5	**Arbiter**	**ODT[2] - (System)**	STAT
29	4	**Arbiter**	**ODT[3] - (System)**	STAT
31	3	C3	WT#	STAT
33	2	J1	TEA#	STAT
35	1	L6	ARTRY#	STAT
37	0	A11	TBST#	STAT

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J3 Odd

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1			
8	15	V6	D[16]	DATA
10	14	U8	D[17]	DATA
12	13	V9	D[18]	DATA
14	12	T7	D[19]	DATA
16	11	U7	D[20]	DATA
18	10	R7	D[21]	DATA
20	9	U6	D[22]	DATA
22	8	W5	D[23]	DATA
24	7	U5	D[24]	DATA
26	6	W4	D[25]	DATA
28	5	P7	D[26]	DATA
30	4	V5	D[27]	DATA
32	3	V4	D[28]	DATA
34	2	W3	D[29]	DATA
36	1	U4	D[30]	DATA
38	0	R5	D[31]	DATA

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J3 Even

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1			
7	15	W12	D[0]	DATA
9	14	W11	D[1]	DATA
11	13	V11	D[2]	DATA
13	12	T9	D[3]	DATA
15	11	W10	D[4]	DATA
17	10	U9	D[5]	DATA
19	9	U10	D[6]	DATA
21	8	M11	D[7]	DATA
23	7	M9	D[8]	DATA
25	6	P8	D[9]	DATA
27	5	W7	D[10]	DATA
29	4	P9	D[11]	DATA
31	3	W9	D[12]	DATA
33	2	R10	D[13]	DATA
35	1	W6	D[14]	DATA
37	0	V7	D[15]	DATA

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J4 Odd

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1			
8	15	U13	D[48]	DATA_B
10	14	V10	D[49]	DATA_B
12	13	W8	D[50]	DATA_B
14	12	T11	D[51]	DATA_B
16	11	U11	D[52]	DATA_B
18	10	V12	D[53]	DATA_B
20	9	V8	D[54]	DATA_B
22	8	T1	D[55]	DATA_B
24	7	P1	D[56]	DATA_B
26	6	V1	D[57]	DATA_B
28	5	U1	D[58]	DATA_B
30	4	N1	D[59]	DATA_B
32	3	R2	D[60]	DATA_B
34	2	V3	D[61]	DATA_B
36	1	U3	D[62]	DATA_B
38	0	W2	D[63]	DATA_B

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J4 Even

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1			
7	15	M6	D[32]	DATA_B
9	14	P3	D[33]	DATA_B
11	13	N4	D[34]	DATA_B
13	12	N5	D[35]	DATA_B
15	11	R3	D[36]	DATA_B
17	10	M7	D[37]	DATA_B
19	9	T2	D[38]	DATA_B
21	8	N6	D[39]	DATA_B
23	7	U2	D[40]	DATA_B
25	6	N7	D[41]	DATA_B
27	5	P11	D[42]	DATA_B
29	4	V13	D[43]	DATA_B
31	3	U12	D[44]	DATA_B
33	2	P12	D[45]	DATA_B
35	1	T13	D[46]	DATA_B
37	0	W13	D[47]	DATA_B

**E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J5 Odd (Multi-Processor MPXbus)**

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1			
8	15		BR# - (Processor 2)	STAT_B
10	14		BG# - (Processor 2)	STAT_B
12	13		DBG# - (Processor 2)	STAT_B
14	12		HIT# - (Processor 2)	STAT_B
16	11		DRDY# - (Processor 2)	STAT_B
18	10		DTI[0] - (Processor 2)	STAT_B
20	9		DTI[1] - (Processor 2)	STAT_B
22	8		DTI[2] - (Processor 2)	STAT_B
24	7		BR# - (Processor 1)	STAT_B
26	6		BG# - (Processor 1)	STAT_B
28	5		DBG# - (Processor 1)	STAT_B
30	4		HIT# - (Processor 1)	STAT_B
32	3		DRDY# - (Processor 1)	STAT_B
34	2		DTI[0] - (Processor 1)	STAT_B
36	1		DTI[1] - (Processor 1)	STAT_B
38	0		DTI[2] - (Processor 1)	STAT_B

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J5 Even (Multi-Processor MPXbus)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1			
7	15			
9	14			
11	13			
13	12			
15	11			
17	10			
19	9			
21	8			
23	7		BR# - (Processor 3)	STAT_B
25	6		BG# - (Processor 3)	STAT_B
27	5		DBG# - (Processor 3)	STAT_B
29	4		HIT# - (Processor 3)	STAT_B
31	3		DRDY# - (Processor 3)	STAT_B
33	2		DTI[0] - (Processor 3)	STAT_B
35	1		DTI[1] - (Processor 3)	STAT_B
37	0		DTI[2] - (Processor 3)	STAT_B

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J6 Odd (L2 Cache)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1			
8	15	L19	L2_ADDR[2]	
10	14	L18	L2_ADDR[1]	
12	13	L17	L2_ADDR[0] (LSB)	
14	12	V14	L2_DATAP[0]	
16	11	U16	L2_DATAP[1]	
18	10	T19	L2_DATAP[2]	
20	9	N18	L2_DATAP[3]	
22	8	H14	L2_DATAP[4]	
24	7	F17	L2_DATAP[5]	
26	6	C19	L2_DATAP[6]	
28	5	B15	L2_DATAP[7]	
30	4	P17	L2_CE#	
32	3	N16	L2_WE#	
34	2			
36	1			
38	0			

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J6 Even (L2 Cache)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1			
7	15	W19	L2_ADDR[18] (MSB)	
9	14	K19	L2_ADDR[17]	
11	13	G18	L2_ADDR[16]	
13	12	H19	L2_ADDR[15]	
15	11	J13	L2_ADDR[14]	
17	10	J14	L2_ADDR[13]	
19	9	H17	L2_ADDR[12]	
21	8	H18	L2_ADDR[11]	
23	7	J16	L2_ADDR[10]	
25	6	J17	L2_ADDR[9]	
27	5	J18	L2_ADDR[8]	
29	4	J19	L2_ADDR[7]	
31	3	K15	L2_ADDR[6]	
33	2	K17	L2_ADDR[5]	
35	1	K18	L2_ADDR[4]	
37	0	M19	L2_ADDR[3]	

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J7 Odd (L2 Cache)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1			
8	15	T18	L2_DATA[16]	
10	14	T17	L2_DATA[17]	
12	13	R19	L2_DATA[18]	
14	12	R18	L2_DATA[19]	
16	11	R17	L2_DATA[20]	
18	10	R15	L2_DATA[21]	
20	9	P19	L2_DATA[22]	
22	8	P18	L2_DATA[23]	
24	7	P13	L2_DATA[24]	
26	6	N14	L2_DATA[25]	
28	5	N13	L2_DATA[26]	
30	4	N19	L2_DATA[27]	
32	3	N17	L2_DATA[28]	
34	2	M17	L2_DATA[29]	
36	1	M13	L2_DATA[30]	
38	0	M18	L2_DATA[31]	

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J7 Even (L2 Cache)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1			
7	15	U14	L2_DATA[0]	
9	14	R13	L2_DATA[1]	
11	13	W14	L2_DATA[2]	
13	12	W15	L2_DATA[3]	
15	11	V15	L2_DATA[4]	
17	10	U15	L2_DATA[5]	
19	9	W16	L2_DATA[6]	
21	8	V16	L2_DATA[7]	
23	7	W17	L2_DATA[8]	
25	6	V17	L2_DATA[9]	
27	5	U17	L2_DATA[10]	
29	4	W18	L2_DATA[11]	
31	3	V18	L2_DATA[12]	
33	2	U18	L2_DATA[13]	
35	1	V19	L2_DATA[14]	
37	0	U19	L2_DATA[15]	

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J8 Odd (L2 Cache)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1			
8	15	C18	L2_DATA[48]	
10	14	C17	L2_DATA[49]	
12	13	B19	L2_DATA[50]	
14	12	B18	L2_DATA[51]	
16	11	B17	L2_DATA[52]	
18	10	A18	L2_DATA[53]	
20	9	A17	L2_DATA[54]	
22	8	A16	L2_DATA[55]	
24	7	B16	L2_DATA[56]	
26	6	C16	L2_DATA[57]	
28	5	A14	L2_DATA[58]	
30	4	A15	L2_DATA[59]	
32	3	C15	L2_DATA[60]	
34	2	B14	L2_DATA[61]	
36	1	C14	L2_DATA[62]	
38	0	E13	L2_DATA[63]	

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J8 Even (L2 Cache)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1			
7	15	H13	L2_DATA[32]	
9	14	G19	L2_DATA[33]	
11	13	G16	L2_DATA[34]	
13	12	G15	L2_DATA[35]	
15	11	G14	L2_DATA[36]	
17	10	G13	L2_DATA[37]	
19	9	F19	L2_DATA[38]	
21	8	F18	L2_DATA[39]	
23	7	F13	L2_DATA[40]	
25	6	E19	L2_DATA[41]	
27	5	E18	L2_DATA[42]	
29	4	E17	L2_DATA[43]	
31	3	E15	L2_DATA[44]	
33	2	D19	L2_DATA[45]	
35	1	D18	L2_DATA[46]	
37	0	D17	L2_DATA[47]	

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J9 Odd (Miscellaneous)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
6	clk1			
8	15	A4	PLL_CFG[0]	
10	14	A5	PLL_CFG[1]	
12	13	A6	PLL_CFG[2]	
14	12	A7	PLL_CFG[3]	
16	11	A12	SMI#	
18	10	B11	MCP#	
20	9	C11	INT#	
22	8	D7	CKSTP_OUT#	
24	7	B8	CKSTP_IN#	
26	6			
28	5			
30	4			
32	3			
34	2			
36	1			
38	0			

E8135A and E8170B Inverse Assemblers for the MPC7410
Logic Analyzer Interface Signal List
Connector J9 Even (Miscellaneous)

2x19 Pin	LA bit	MPC7410 pin	MPC7410 signal	Label
5	clk1			
7	15	L1	DP[0]	
9	14	P2	DP[1]	
11	13	M2	DP[2]	
13	12	V2	DP[3]	
15	11	M1	DP[4]	
17	10	N2	DP[5]	
19	9	T3	DP[6]	
21	8	R1	DP[7]	
23	7	C4	AP[0]	
25	6	C5	AP[1]	
27	5	C6	AP[2]	
29	4	C7	AP[3]	
31	3	D3	RSRV#	
33	2	A3	EMODE#	
35	1	J3	QREQ#	
37	0	B2	QACK#	

Signal-to-Connector Mapping—MPC744X/5X

The following tables show the electrical signal-to-connector mapping required by the E8170B and E8135A inverse assembler software.

- For single processor inverse assembly, MICTOR connectors J1, J2, J3, and J4 are required.
- For multiprocessor inverse assembly using the MPXbus, J5 is also required.
- In order to see L3 cache data, MICTOR connectors J6, J7, and J8 are also required.

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.

NOTE:

When using the MPXbus mode with the MPC7410/4X/5X, the inverse assembler requires an ODT signal from the system arbiter. **The ODT signal must be designed into your system arbiter.** The ODT signal is used to initialize the inverse assembler so that it can match address tenures to their respective data tenures. Please see “Using the MPXbus Tool” on page 98 for details.

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J1 Odd

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1	A10	D6	SYSCLK	STAT
8	15	J2	P1	A[20]	ADDR
10	14	K4	P4	A[21]	ADDR
12	13	N4	R6	A[22]	ADDR
14	12	J3	M7	A[23]	ADDR
16	11	M5	N7	A[24]	ADDR
18	10	P5	AA3	A[25]	ADDR
20	9	N3	U4	A[26]	ADDR
22	8	T1	W2	A[27]	ADDR
24	7	V2	W1	A[28]	ADDR
26	6	U1	W3	A[29]	ADDR
28	5	N5	V4	A[30]	ADDR
30	4	W1	AA1	A[31]	ADDR
32	3	B12	D10	A[32]	ADDR
34	2	C4	J4	A[33]	ADDR
36	1	G10	G10	A[34]	ADDR
38	0	B11	D9	A[35]	ADDR

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J1 Even

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1	L4	P5	TS#	STAT
7	15	F10	C8	A[4]	ADDR
9	14	L2	R2	A[5]	ADDR
11	13	D11	A7	A[6]	ADDR
13	12	D1	M2	A[7]	ADDR
15	11	C10	A6	A[8]	ADDR
17	10	G2	M1	A[9]	ADDR
19	9	D12	A10	A[10]	ADDR
21	8	L3	U2	A[11]	ADDR
23	7	G4	N2	A[12]	ADDR
25	6	T2	P8	A[13]	ADDR
27	5	F4	M8	A[14]	ADDR
29	4	V1	W4	A[15]	ADDR
31	3	J4	N6	A[16]	ADDR
33	2	R2	U6	A[17]	ADDR
35	1	K5	R5	A[18]	ADDR
37	0	W2	Y4	A[19]	ADDR

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J2 Odd

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1	K6	N8	TA#	STAT
8	15	L1	T1	TEA#	STAT
10	14	G6	L1	TSIZ[0]	STAT
12	13	F7	H3	TSIZ[1]	STAT
14	12	E7	D1	TSIZ[2]	STAT
16	11	D10	A9	DX#	STAT
18	10	E5	F1	TT[0]	STAT
20	9	E6	F4	TT[1]	STAT
22	8	F6	K8	TT[2]	STAT
24	7	E9	A5	TT[3]	STAT
26	6	C5	E1	TT[4]	STAT
28	5	F11	B7	TBST#	STAT
30	4	D3	L2	WT#	STAT
32	3	E11	E10	A[0]	ADDR_B
34	2	H1	N4	A[1]	ADDR_B
36	1	C11	E8	A[2]	ADDR_B
38	0	G3	N5	A[3]	ADDR_B

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J2 Even

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1	R1	U1	AACK#	STAT
7	15	D2	K1	BR# - (Processor 0)	STAT
9	14	M1	R3	BG# - (Processor 0)	STAT
11	13	M2	V1	DBG# - (Processor 0)	STAT
13	12	B2	K2	HIT# - (Processor 0)	STAT
15	11	R3	T6	DRDY# - (Processor 0)	STAT
17	10	G1	P2	DTI[0] - (Processor 0)	STAT
19	9	K1	T5	DTI[1] - (Processor 0)	STAT
21	8	P1	U3	DTI[2] - (Processor 0)	STAT
23	7	N1	P6	DTI[3] - (Processor 0)	STAT
25	6	Future	Future	DTI[4] - (Processor 0)	STAT
27	5	**Arbiter**	**Arbiter**	**ODT[0] - (System)**	STAT
29	4	**Arbiter**	**Arbiter**	**ODT[1] - (System)**	STAT
31	3	**Arbiter**	**Arbiter**	**ODT[2] - (System)**	STAT
33	2	**Arbiter**	**Arbiter**	**ODT[3] - (System)**	STAT
35	1	**Arbiter**	**Future**	**ODT[4] - (System)**	STAT
37	0	N2	T2	ARTRY#	STAT

**E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J3 Odd**

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1				
8	15	W12	AB12	D[16]	DATA
10	14	V12	R12	D[17]	DATA
12	13	N11	AA13	D[18]	DATA
14	12	N10	AB11	D[19]	DATA
16	11	R11	Y12	D[20]	DATA
18	10	U11	V11	D[21]	DATA
20	9	W11	T11	D[22]	DATA
22	8	T11	R11	D[23]	DATA
24	7	R10	W10	D[24]	DATA
26	6	N9	T10	D[25]	DATA
28	5	P10	W11	D[26]	DATA
30	4	U10	V10	D[27]	DATA
32	3	R9	R10	D[28]	DATA
34	2	W10	U10	D[29]	DATA
36	1	U9	AA10	D[30]	DATA
38	0	V9	U9	D[31]	DATA

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J3 Even

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1				
7	15	R15	AB15	D[0]	DATA
9	14	W15	T14	D[1]	DATA
11	13	T14	R14	D[2]	DATA
13	12	V16	AB13	D[3]	DATA
15	11	W16	V14	D[4]	DATA
17	10	T15	U14	D[5]	DATA
19	9	U15	AB14	D[6]	DATA
21	8	P14	W16	D[7]	DATA
23	7	V13	AA11	D[8]	DATA
25	6	W13	Y11	D[9]	DATA
27	5	T13	W2	D[10]	DATA
29	4	P13	W13	D[11]	DATA
31	3	U14	Y14	D[12]	DATA
33	2	W14	U13	D[13]	DATA
35	1	R12	T12	D[14]	DATA
37	0	T12	W12	D[15]	DATA

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J4 Odd

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1				
8	15	U18	AA18	D[48]	DATA_B
10	14	W17	W14	D[49]	DATA_B
12	13	W18	R13	D[50]	DATA_B
14	12	T16	W15	D[51]	DATA_B
16	11	T18	AA14	D[52]	DATA_B
18	10	T17	V16	D[53]	DATA_B
20	9	W3	W6	D[54]	DATA_B
22	8	V17	AA12	D[55]	DATA_B
24	7	U4	V6	D[56]	DATA_B
26	6	U8	AB9	D[57]	DATA_B
28	5	U7	AB6	D[58]	DATA_B
30	4	R7	R7	D[59]	DATA_B
32	3	P6	R9	D[60]	DATA_B
34	2	R8	AA9	D[61]	DATA_B
36	1	W8	AB8	D[62]	DATA_B
38	0	T8	W9	D[63]	DATA_B

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J4 Even

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1				
7	15	W5	V7	D[32]	DATA_B
9	14	U6	T8	D[33]	DATA_B
11	13	T5	AB4	D[34]	DATA_B
13	12	U5	Y6	D[35]	DATA_B
15	11	W7	AB7	D[36]	DATA_B
17	10	R6	AA6	D[37]	DATA_B
19	9	P7	Y8	D[38]	DATA_B
21	8	V6	AA7	D[39]	DATA_B
23	7	P17	W8	D[40]	DATA_B
25	6	R19	AB10	D[41]	DATA_B
27	5	V18	AA16	D[42]	DATA_B
29	4	R18	AB16	D[43]	DATA_B
31	3	V19	AB17	D[44]	DATA_B
33	2	T19	Y18	D[45]	DATA_B
35	1	U19	AB18	D[46]	DATA_B
37	0	W19	Y16	D[47]	DATA_B

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J5 Odd (Multi-Processor MPXbus)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1				
8	15			DRDY# - (Processor 2)	STAT_B
10	14			DTI[0] - (Processor 2)	STAT_B
12	13			DTI[1] - (Processor 2)	STAT_B
14	12			DTI[2] - (Processor 2)	STAT_B
16	11			DTI[3] - (Processor 2)	STAT_B
18	10	Future	Future	DTI[4] - (Processor 2)	STAT_B
20	9			BR# - (Processor 1)	STAT_B
22	8			BG# - (Processor 1)	STAT_B
24	7			DBG# - (Processor 1)	STAT_B
26	6			HIT# - (Processor 1)	STAT_B
28	5			DRDY# - (Processor 1)	STAT_B
30	4			DTI[0] - (Processor 1)	STAT_B
32	3			DTI[1] - (Processor 1)	STAT_B
34	2			DTI[2] - (Processor 1)	STAT_B
36	1			DTI[3] - (Processor 1)	STAT_B
38	0	Future	Future	DTI[4] - (Processor 1)	STAT_B

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J5 Even (Multi-Processor MPXbus)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1				
7	15				
9	14				
11	13			BR# - (Processor 3)	STAT_B
13	12			BG# - (Processor 3)	STAT_B
15	11			DBG# - (Processor 3)	STAT_B
17	10			HIT# - (Processor 3)	STAT_B
19	9			DRDY# - (Processor 3)	STAT_B
21	8			DTI[0] - (Processor 3)	STAT_B
23	7			DTI[1] - (Processor 3)	STAT_B
25	6			DTI[2] - (Processor 3)	STAT_B
27	5			DTI[3] - (Processor 3)	STAT_B
29	4	Future	Future	DTI[4] - (Processor 3)	STAT_B
31	3			BR# - (Processor 2)	STAT_B
33	2			BG# - (Processor 2)	STAT_B
35	1			DBG# - (Processor 2)	STAT_B
37	0			HIT# - (Processor 2)	STAT_B

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J6 Odd (L3 Cache)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1		V22	L3_CLK[0]	
8	15	D17	E22	L3_ADDR[15]	L3_ADDR
10	14	E16	H18	L3_ADDR[14]	L3_ADDR
12	13	G19	G20	L3_ADDR[13]	L3_ADDR
14	12	F15	F22	L3_ADDR[12]	L3_ADDR
16	11	D19	G22	L3_ADDR[11]	L3_ADDR
18	10	G16	H20	L3_ADDR[10]	L3_ADDR
20	9	F16	K16	L3_ADDR[9]	L3_ADDR
22	8	D18	J18	L3_ADDR[8]	L3_ADDR
24	7	E19	H22	L3_ADDR[7]	L3_ADDR
26	6	H16	J20	L3_ADDR[6]	L3_ADDR
28	5	H17	J22	L3_ADDR[5]	L3_ADDR
30	4	H18	K18	L3_ADDR[4]	L3_ADDR
32	3	H19	K20	L3_ADDR[3]	L3_ADDR
34	2	F19	L16	L3_ADDR[2]	L3_ADDR
36	1	F17	K22	L3_ADDR[1]	L3_ADDR
38	0	F18	L18	L3_ADDR[0] (LSB)	

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J6 even (L3 Cache)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1		C17	L3_CLK[1]	
7	15		L20	L3_CNTL[0]	L3_STAT
9	14		L22	L3_CNTL[1]	L3_STAT
11	13		V18	L3_ECHO_CLK[0]	L3_MISC
13	12		P20	L3_ECHO_CLK[1]	L3_MISC
15	11		E18	L3_ECHO_CLK[2]	L3_MISC
17	10		E14	L3_ECHO_CLK[3]	L3_MISC
19	9	N12	AB19	L3_DATAP[0]	L3_MISC
21	8	N18	AA22	L3_DATAP[1]	L3_MISC
23	7	K17	P22	L3_DATAP[2]	L3_MISC
25	6	N19	P16	L3_DATAP[3]	L3_MISC
27	5	B18	C20	L3_DATAP[4]	L3_MISC
29	4	E12	E16	L3_DATAP[5]	L3_MISC
31	3	B13	A15	L3_DATAP[6]	L3_MISC
33	2	B14	A12	L3_DATAP[7]	L3_MISC
35	1		F20	L3_ADDR[17](MSB)	L3_ADDR
37	0	D16	J16	L3_ADDR[16]	L3_ADDR

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J7 Odd (L3 Cache)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1				
8	15	K15	N18	L3_DATA[16]	L3_DATA
10	14	J14	N20	L3_DATA[17]	L3_DATA
12	13	J18	N16	L3_DATA[18]	L3_DATA
14	12	J19	N22	L3_DATA[19]	L3_DATA
16	11	J15	M16	L3_DATA[20]	L3_DATA
18	10	K19	M18	L3_DATA[21]	L3_DATA
20	9	J16	M20	L3_DATA[22]	L3_DATA
22	8	H15	M22	L3_DATA[23]	L3_DATA
24	7	L16	R18	L3_DATA[24]	L3_DATA
26	6	P16	T20	L3_DATA[25]	L3_DATA
28	5	M18	U22	L3_DATA[26]	L3_DATA
30	4	L19	T22	L3_DATA[27]	L3_DATA
32	3	L18	R20	L3_DATA[28]	L3_DATA
34	2	K18	P18	L3_DATA[29]	L3_DATA
36	1	J17	R22	L3_DATA[30]	L3_DATA
38	0	K16	M15	L3_DATA[31]	L3_DATA

**E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J7 Even (L3 Cache)**

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1				
7	15	P15	AA19	L3_DATA[0]	L3_DATA
9	14	L15	AB20	L3_DATA[1]	L3_DATA
11	13	N15	U16	L3_DATA[2]	L3_DATA
13	12	P18	W18	L3_DATA[3]	L3_DATA
15	11	N14	AA20	L3_DATA[4]	L3_DATA
17	10	M14	AB21	L3_DATA[5]	L3_DATA
19	9	M17	AA21	L3_DATA[6]	L3_DATA
21	8	N13	T16	L3_DATA[7]	L3_DATA
23	7	N16	W20	L3_DATA[8]	L3_DATA
25	6	M19	U18	L3_DATA[9]	L3_DATA
27	5	M16	Y22	L3_DATA[10]	L3_DATA
29	4	P19	R16	L3_DATA[11]	L3_DATA
31	3	N17	V20	L3_DATA[12]	L3_DATA
33	2	M15	W22	L3_DATA[13]	L3_DATA
35	1	L17	T18	L3_DATA[14]	L3_DATA
37	0	L14	U20	L3_DATA[15]	L3_DATA

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J8 Odd (L3 Cache)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1				
8	15	G14	A18	L3_DATA[48]	L3_DATA_B
10	14	C15	G14	L3_DATA[49]	L3_DATA_B
12	13	A17	E15	L3_DATA[50]	L3_DATA_B
14	12	G12	C16	L3_DATA[51]	L3_DATA_B
16	11	F14	A17	L3_DATA[52]	L3_DATA_B
18	10	F13	A16	L3_DATA[53]	L3_DATA_B
20	9	E13	C15	L3_DATA[54]	L3_DATA_B
22	8	B16	G13	L3_DATA[55]	L3_DATA_B
24	7	A15	C14	L3_DATA[56]	L3_DATA_B
26	6	C14	A14	L3_DATA[57]	L3_DATA_B
28	5	A18	E13	L3_DATA[58]	L3_DATA_B
30	4	A13	C13	L3_DATA[59]	L3_DATA_B
32	3	F12	G12	L3_DATA[60]	L3_DATA_B
34	2	A14	A13	L3_DATA[61]	L3_DATA_B
36	1	G11	E12	L3_DATA[62]	L3_DATA_B
38	0	C13	C12	L3_DATA[63]	L3_DATA_B

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J8 Even (L3 Cache)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1				
7	15	C19	G18	L3_DATA[32]	L3_DATA_B
9	14	D15	D22	L3_DATA[33]	L3_DATA_B
11	13	G15	E20	L3_DATA[34]	L3_DATA_B
13	12	C18	H16	L3_DATA[35]	L3_DATA_B
15	11	A16	C22	L3_DATA[36]	L3_DATA_B
17	10	B19	F18	L3_DATA[37]	L3_DATA_B
19	9	A19	D20	L3_DATA[38]	L3_DATA_B
21	8	D14	B22	L3_DATA[39]	L3_DATA_B
23	7	E15	G16	L3_DATA[40]	L3_DATA_B
25	6	B15	A21	L3_DATA[41]	L3_DATA_B
27	5	B17	G15	L3_DATA[42]	L3_DATA_B
29	4	C17	E17	L3_DATA[43]	L3_DATA_B
31	3	C16	A20	L3_DATA[44]	L3_DATA_B
33	2	G13	C19	L3_DATA[45]	L3_DATA_B
35	1	E14	C18	L3_DATA[46]	L3_DATA_B
37	0	H14	A19	L3_DATA[47]	L3_DATA_B

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J9 Odd (Miscellaneous)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
6	clk1	A2	G1	SRESET#	MISC
8	15	E4	L4	SHD0#	MISC
10	14	H5	L8	SHD1#	MISC
12	13	E2	M4	GBL#	MISC
14	12	T3	AA2	DP[0]	MISC
16	11	W4	AB3	DP[1]	MISC
18	10	T4	AB2	DP[2]	MISC
20	9	W9	AA8	DP[3]	MISC
22	8	M6	R8	DP[4]	MISC
24	7	V3	W5	DP[5]	MISC
26	6	N8	U8	DP[6]	MISC
28	5	W6	AB5	DP[7]	MISC
30	4	C1	L5	AP[0]	MISC
32	3	E3	L6	AP[1]	MISC
34	2	H6	J1	AP[2]	MISC
36	1	F5	H2	AP[3]	MISC
38	0	G7	G5	AP[4]	MISC

E8135A Inverse Assembler for the MPC744X/5X
Logic Analyzer Interface Signal List
Connector J9 Even (Miscellaneous)

2x19 Pin	LA bit	MPC7440 pin	MPC7450 pin	MPC7440/50 signal	Label
5	clk1	D8	A3	HRESET#	MISC
7	15	P4	Y1	QREQ#	MISC
9	14	G5	K7	QACK#	MISC
11	13	A9	B4	PMON_OUT#	MISC
13	12	D9	E6	PMON_IN#	MISC
15	11	B8	A2	PLL_CFG[0]	MISC
17	10	C8	F7	PLL_CFG[1]	MISC
19	9	C7	C2	PLL_CFG[2]	MISC
21	8	D7	D4	PLL_CFG[3]	MISC
23	7	F9	G8	SMI#	MISC
25	6	C9	B8	MCP#	MISC
27	5	D4	J6	INT#	MISC
29	4	B1	K6	CKSTP_OUT#	MISC
31	3	A3	F3	CKSTP_IN#	MISC
33	2	J1	R1	CI#	MISC
35	1	G9	C6	BMODE0#	MISC
37	0	F8	C4	BMODE1#	MISC

Designing a JTAG Connector into Your Target System

If you are using an emulation probe/module with your logic analysis system, you will need to install a JTAG connector on your target system. For information on designing a JTAG connector into your target system, see the emulation manual supplied with your emulation probe/module.

3

Setting Up the Logic Analysis System

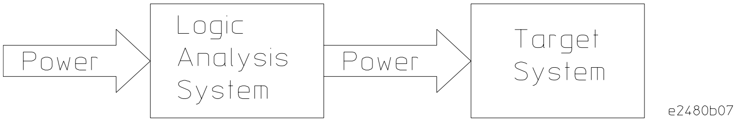
Power-on/Power-off Sequence

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

To power on 16700-series logic analysis systems

Ensure the target system is powered off.

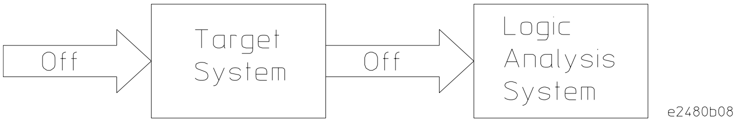
1. Turn on the logic analyzer. The Setup Assistant will guide you through the process of connecting and configuring the logic analyzer.
2. When the target system is connected to the logic analyzer, and everything is configured, turn on your target system.



To power off

Turn off power to your system in the following order:

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



Installing Logic Analyzer Modules

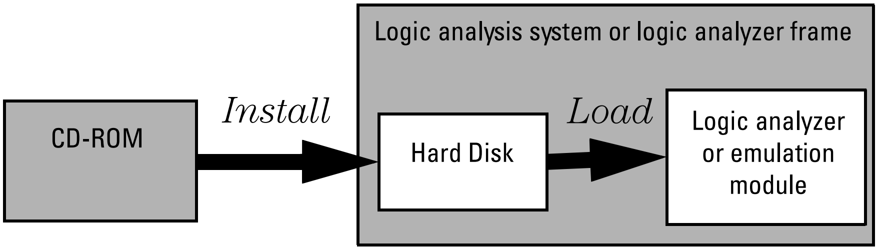
You should install logic analyzer, oscilloscope, or pattern generator modules in your logic analysis system before you install software.

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

Refer to your logic analysis systemÕs *Installation Guide* for instructions on installing logic analyzer modules.

Installing and Loading Software

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


Logic analysis system or logic analyzer frame
CD-ROM
Install
Hard Disk
Load
Logic analyzer or emulation module

What needs to be installed

NOTE: If you ordered an inverse assembler 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 assemblers (automatically loaded with the configuration files)
- Personality files for the Setup Assistant

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

To install the software from CD-ROM

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

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

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

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

2. Insert the CD-ROM in the drive.

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

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

5. Select **Install...**.

   Change the media type to "**CD-ROM**" if necessary.

6. Select **Apply**.

7. From the list of types of packages, double-click "**PROC-SUPPORT**."

NOTE: For touch screen systems, double select the "**PROC-SUPPORT"** line by quickly touching it twice.

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

8. Select the **"POWERPC74XX"** 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.

NOTE: When installing the E8135A (MPXbus) Inverse Assembler, a MPXbus Tool is also installed. You will need to re-start your logic analysis session in order to use the MPXbus Tool or the E8135A Inverse Assembler. See “Using the MPXbus Tool” on page 98 for more information.

The configuration files are stored in /logic/configs/hp/mpc74xx/74xx. The inverse assemblers are stored in /logic/ia.

See Also Refer to the instructions printed on the CD-ROM package for a summary of the installation instructions.

See Also Refer to the online help for more information on installing, licensing, and removing software.

To list software packages which are installed

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

4

Connecting the Logic Analyzer to the Target System

Connecting the Logic Analyzer to the Target System

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

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

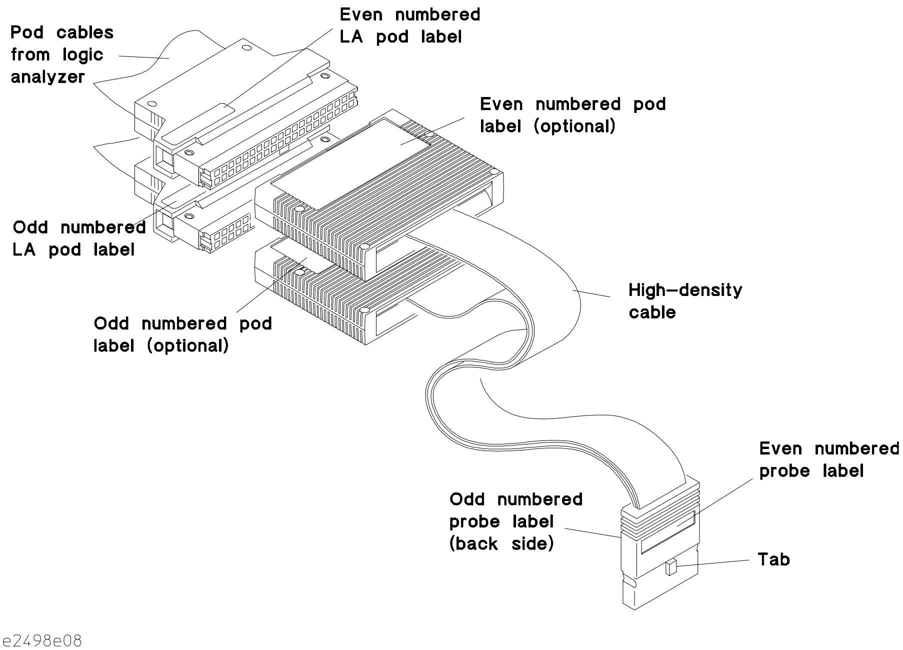
If you are not using the Setup Assistant, follow the instructions given in this chapter. This chapter covers the following tasks in the recommended order:

- Check that the target system meets the necessary requirements (see page 22)
- Read the power on/power off sequence (see page 70)
- Connect the target system to the logic analyzer (see page 77)
- Configure the logic analyzer (see page 85)

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

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

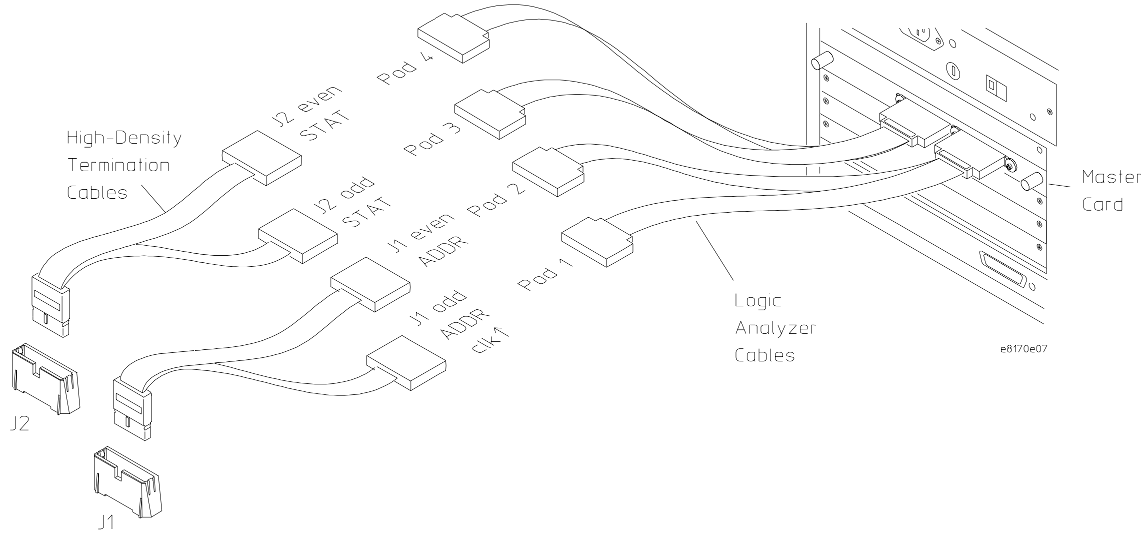
E5346A Cable Numbering


Pod cables from logic analyzer
Even numbered LA pod label
Even numbered pod label (optional)
Odd numbered LA pod label
High–density cable
Odd numbered pod label (optional)
Even numbered probe label
Odd numbered probe label (back side)
Tab

e2498e08

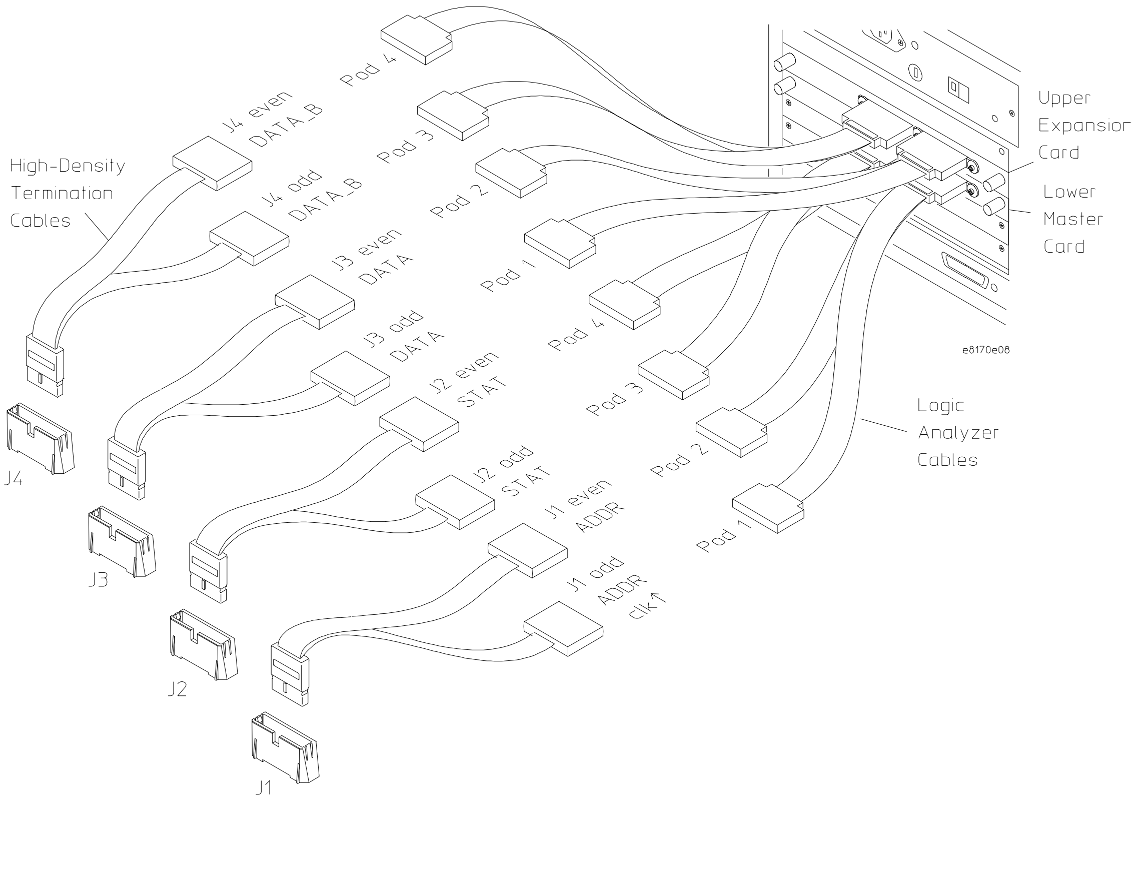
To connect to the 16715/16/17/18/19/40/41/42/50/51/52/53/54/55/56A logic analyzer (one card)

Use the figure below to connect the target system to the 16715/16/17/18/19/40/41/42/50/51/52/53/54/55/56A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



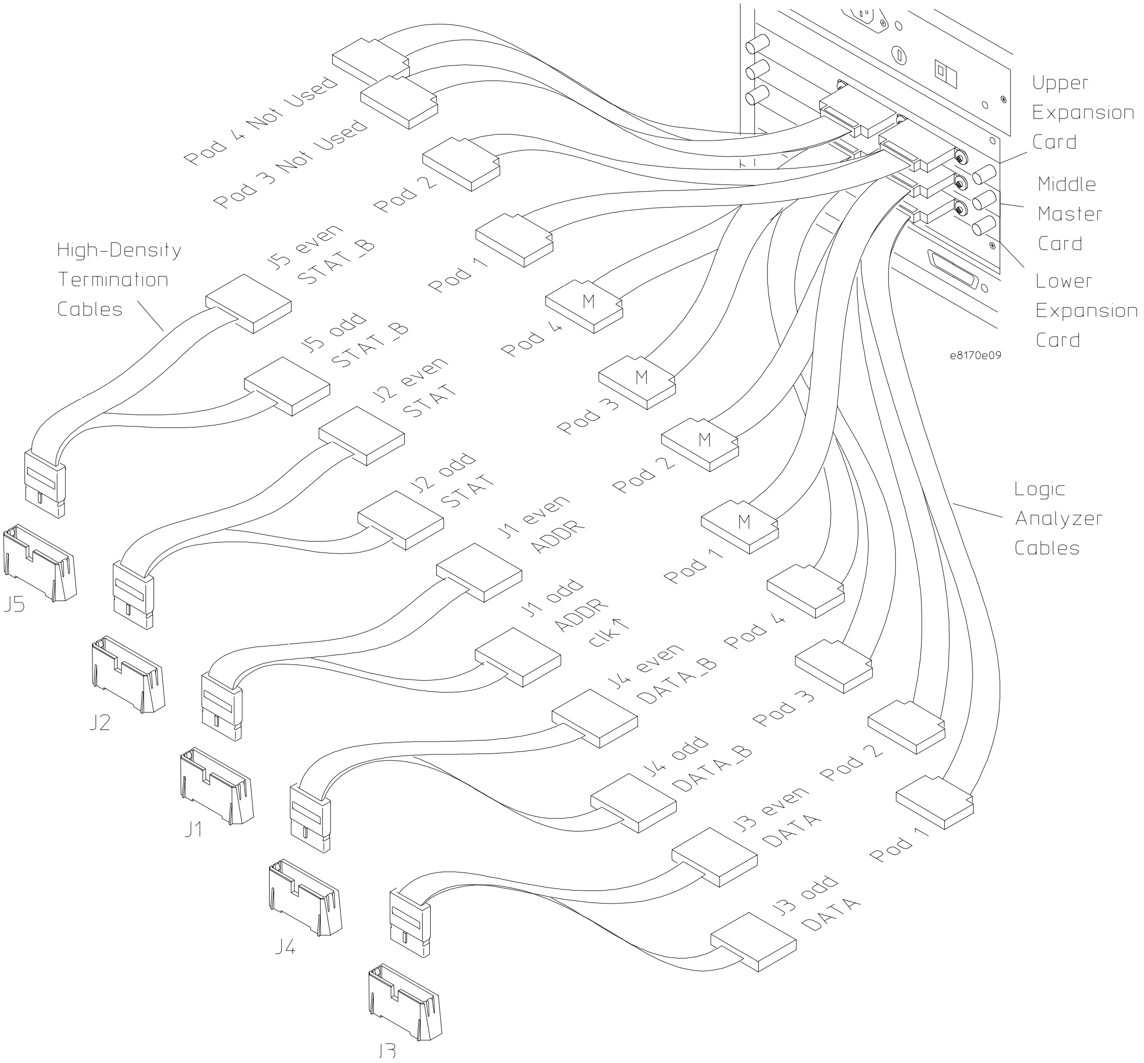
To connect to the 16715/16/17/18/19/40/41/42/50/51/52/53/54/55/56A logic analyzer (two cards)

Use the figure below to connect the target system to the 16715/16/17/18/19/40/41/42/50/51/52/53/54/55/56A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



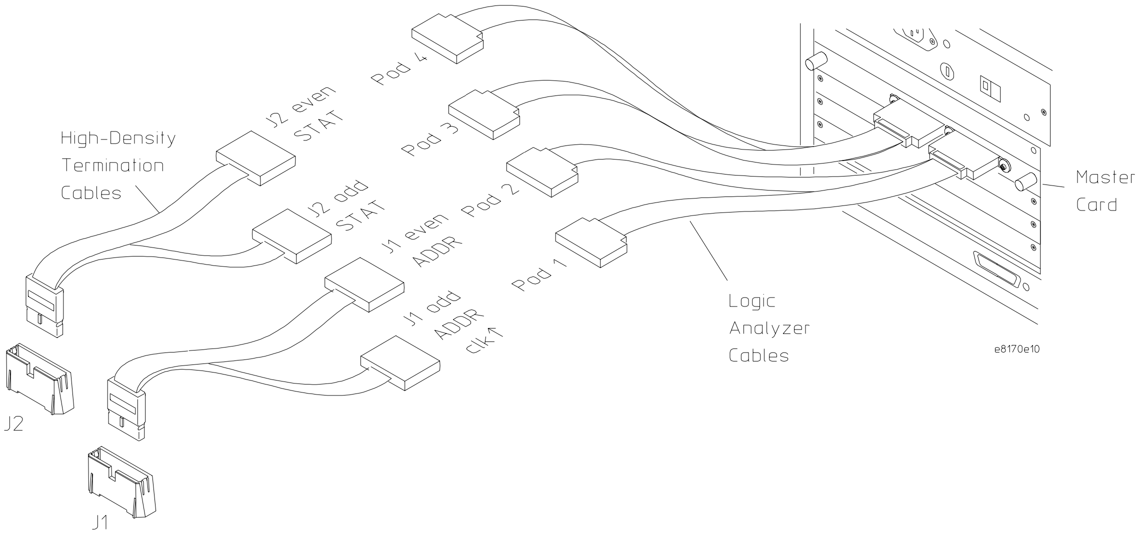
To connect to the 16715/16/17/18/19/40/41/42/50/51/52/53/54/55/56A logic analyzer (three cards)

Use the figure below to connect the target system to the 16715/16/17/18/19/40/41/42/50/51/52/53/54/55/56A logic analyzer. Use the labels that were shipped with the high-density cables to identify the connections.



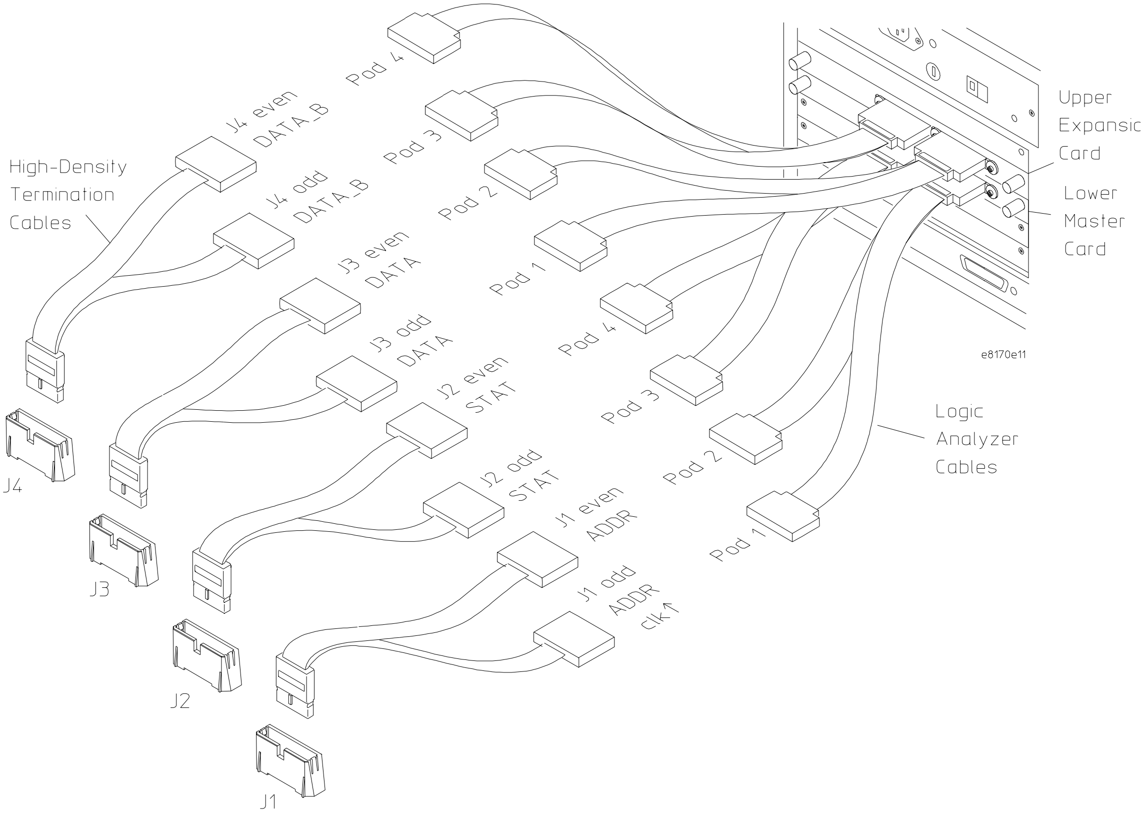
To connect to the 16557 logic analyzer (one-card)

Use the figure below to connect the target system to the two-card 16557 logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



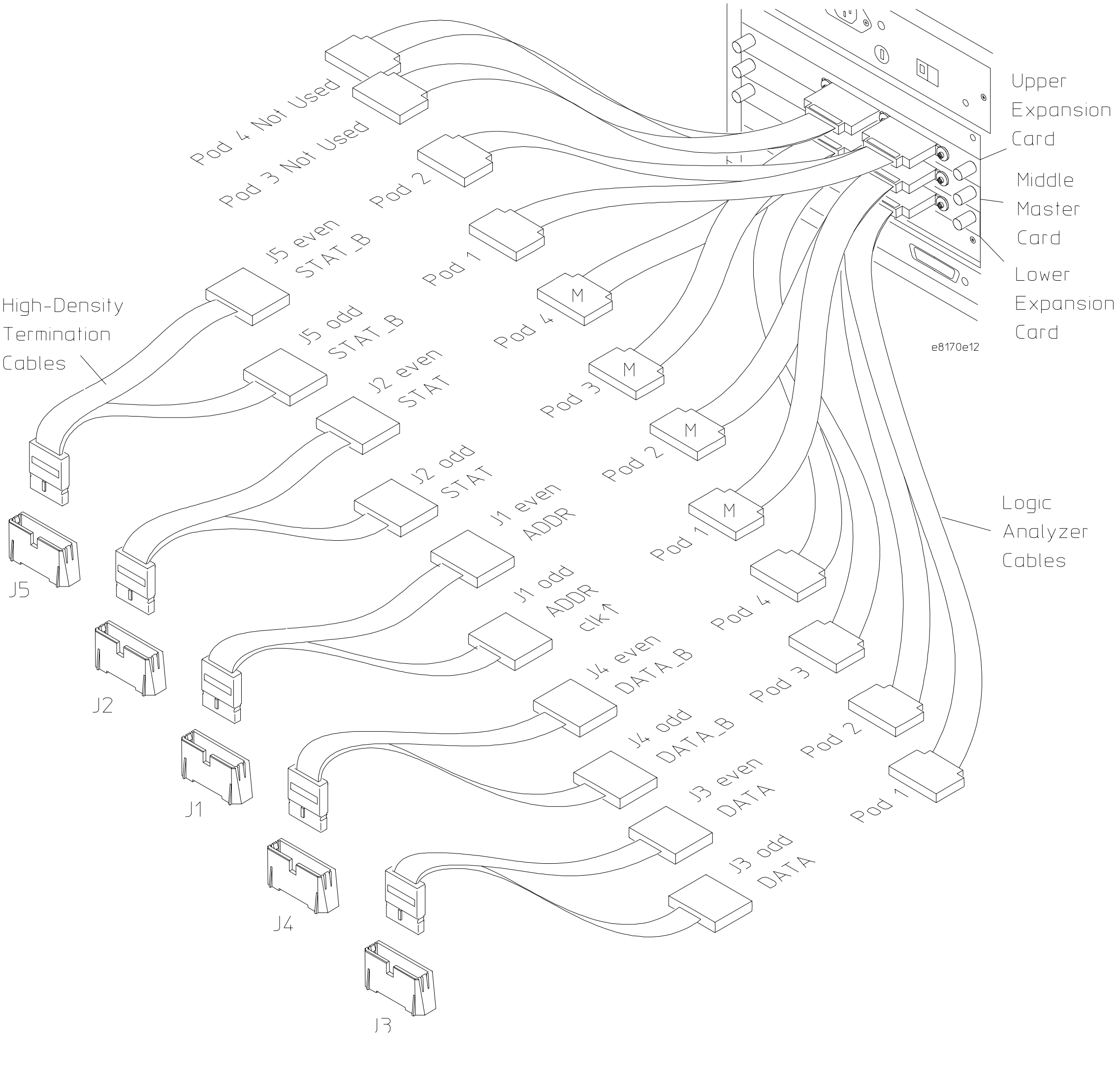
To connect to the 16557 logic analyzer (two-cards)

Use the figure below to connect the target system to the two-card 16557 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 16557 logic analyzer (three-cards)

Use the figure below to connect the target system to the 16557 logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



5

Configuring the Logic Analyzer

This chapter describes setting up and using the MPC7410/4X/5X inverse assemblers.

The information in this chapter includes:

- Loading configuration files and inverse assemblers
- Using the format menu
- Using the MPXbus Tool
- Using symbols

Configuring 16700-series Logic Analysis Systems

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

The information in the configuration file includes:

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

The configuration file you use is determined by the logic analyzer you are using and which bus mode you are using (60x bus or MPXbus).

> It is strongly recommended that you do not change the setup related to the MPC7410/40/50 sampling, format, pod assignment, or configuration dialogs. The configuration file (loaded by the Setup Assistant in 16700-series logic analysis systems) will configure the logic analyzer for making measurements of the MPC7410/40/50.

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

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

1 Click on the File Manager icon. Use File Manager to ensure that the subdirectory /logic/configs/hp/mpc74xx/ exists.

> If the above directory does not exist, you need to install the MPC74XX Processor Support Package. Close File Manager, then use the procedure on the CD-ROM jacket to install the MPC74XX Processor Support Package before you continue.

2 Using File Manager, select the configuration file that you want to load from the /logic/configs/hp/mpc74xx 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 MPC7410/4X/5X analysis by loading the appropriate MPC7410/4X/5X configuration file. Loading the indicated file also automatically loads the correct inverse assembler.

3 Close File Manager.

MPC7410 Logic Analyzer Configuration Files

The following table lists the configuration files for the MPC7410 for each supported logic analyzer card configuration.

MPC7410 Logic Analyzer Configuration Files

Analyzer Model	Analyzer Description*	60x bus Configuration File	PMC Card (60x bus) Configuration File	MPXbus Configuration File
16753/54/55/56A (two cards)	600 MHz STATE 4GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL2
16753/54/55/56A (three+ cards)	600 MHz STATE 4GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL3
16750/51/52A (two cards)	400 MHz STATE 2 GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL2
16750/51/52A (three+ cards)	400 MHz STATE 2 GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL3
16740/41/42A (two cards)	200 MHz STATE 2 GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL2
16740/41/42A (three+ cards)	200 MHz STATE 2 GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL3
16715/16/17/18/19A (two cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL2
16715/16/17/18/19A (three+ cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7410_60XL2	C7410_PMCL2	C7410_MPXL3
16557D (two cards)	140 MHz STATE 500 MHz TIMING	C7410_60XM2	C7410_PMCM2	C7410_MPXM2
16557D (three+ cards)	140 MHz STATE 500 MHz TIMING	C7410_60XM2	C7410_PMCM2	C7410_MPXM3

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

The configuration files in the PMC Card column are for the Motorola PMC evaluation board which has different signal-to-connector mappings than the ones recommended in this manual.

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

MPC744X/5X Logic Analyzer Configuration Files

The following tables list the configuration files for the MPC744X/5X for each supported logic analyzer card configuration.

MPC744X/5X Logic Analyzer Configuration Files

Analyzer Model	Analyzer Description*	60x bus Configuration File	PMC Card (60x bus) Configuration File	MPXbus Configuration File
16753/54/55/56A (two cards)	600 MHz STATE 4GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL2
16753/54/55/56A (three+ cards)	600 MHz STATE 4GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL3
16750/51/52A (two cards)	400 MHz STATE 2 GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL2
16750/51/52A (three+ cards)	400 MHz STATE 2 GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL3
16740/41/42A (two cards)	200 MHz STATE 2 GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL2
16740/41/42A (three+ cards)	200 MHz STATE 2 GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL3
16715/16/17/18/19A (two cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL2
16715/16/17/18/19A (three+ cards)	333 MHz STATE 2 GHz TIMING ZOOM	C7440_50_60XL2	C7450_PMCL2	C7440_50_MPXL3
16557D (two cards)	140 MHz STATE 500 MHz TIMING	C7440_50_60XM2	C7450_PMCM2	C7440_50_MPXM2
16557D (three+ cards)	140 MHz STATE 500 MHz TIMING	C7440_50_60XM2	C7450_PMCM2	C7440_50_MPXM3

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

The configuration files in the PMC Card column are for the Motorola PMC evaluation board which has different signal-to-connector mappings than the ones recommended in this manual.

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

Using the Format Menu

This section describes the organization of MPC7410/4X/5X signals in the logic analyzer's Format menu.

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

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

Bit ordering conventions

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

Most Significant	**Least Significant**	
A0	**A31**	***PowerPC***
ADDR31	**ADDR0**	***Logic Analyzer***

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

MPC7410 Label Bit Definitions

The following table shows the signals for each of the main logic analyzer labels (ADDR, ADDR_B, DATA, DATA_B, STAT, STAT_B, and CLK).

NOTE: When using the MPXbus mode with the MPC7410/4X/5X, the inverse assembler requires an ODT signal from the system arbiter. **The ODT signal must be designed into your system arbiter.** The ODT signal is used to initialize the inverse assembler so that it can match address tenures to their respective data tenures. Please see "Using the MPXbus Tool" on page 98 for details.

MPC7410 Label Bit Definitions

Label Name	Bit Number	Signal Name
ADDR	Bit 0	A[31] (LSB)
ADDR	..	...
ADDR	Bit 31	A[0] (MSB)
DATA	Bit 0	D[31]
DATA	...	...
DATA	Bit 31	D[0] (MSB)
DATA_B	Bit 0	D[63] (LSB)
DATA_B	...	...
DATA_B	Bit 31	D[32]
STAT	Bit 0	TT[4]
STAT	Bit 1	TT[3]
STAT	Bit 2	TT[2]
STAT	Bit 3	TT[1]
STAT	Bit 4	TT[0]
STAT	Bit 5	TSIZ[2]
STAT	Bit 6	TSIZ[1]
STAT	Bit 7	TSIZ[0]
STAT	Bit 8	TBST#
STAT	Bit 9	ARTRY#
STAT	Bit 10	TEA#
STAT	Bit 11	WT#
STAT	Bit 12	ODT[3]
STAT	Bit 13	ODT[2]
STAT	Bit 14	ODT[1]
STAT	Bit 15	ODT[0]

Label Name	Bit Number	Signal Name
STAT	Bit 16	DTI[2] - (Processor 0)
STAT	Bit 17	DTI[1] - (Processor 0)
STAT	Bit 18	DTI[0] - (Processor 0)
STAT	Bit 19	DRDY# - (Processor 0)
STAT	Bit 20	HIT# - (Processor 0)
STAT	Bit 21	DBG# - (Processor 0)
STAT	Bit 22	BG# - (Processor 0)
STAT	Bit 23	BR# - (Processor 0)
STAT	Bit 24	SYSCLK
STAT	Bit 25	TS#
STAT	Bit 26	TA#
STAT	Bit 27	AACK#
STAT_B	Bit 0	DTI[2] - (Processor 1)
STAT_B	Bit 1	DTI[1] - (Processor 1)
STAT_B	Bit 2	DTI[0] - (Processor 1)
STAT_B	Bit 3	DRDY# - (Processor 1)
STAT_B	Bit 4	HIT# - (Processor 1)
STAT_B	Bit 5	DBG# - (Processor 1)
STAT_B	Bit 6	BG# - (Processor 1)
STAT_B	Bit 7	BR# - (Processor 1)
STAT_B	Bit 8	DTI[2] - (Processor 2)
STAT_B	Bit 9	DTI[1] - (Processor 2)
STAT_B	Bit 10	DTI[0] - (Processor 2)
STAT_B	Bit 11	DRDY# - (Processor 2)
STAT_B	Bit 12	HIT# - (Processor 2)
STAT_B	Bit 13	DBG# - (Processor 2)
STAT_B	Bit 14	BG# - (Processor 2)
STAT_B	Bit 15	BR# - (Processor 2)
STAT_B	Bit 16	DTI[2] - (Processor 3)
STAT_B	Bit 17	DTI[1] - (Processor 3)
STAT_B	Bit 18	DTI[0] - (Processor 3)
STAT_B	Bit 19	DRDY# - (Processor 3)
STAT_B	Bit 20	HIT# - (Processor 3)
STAT_B	Bit 21	DBG# - (Processor 3)
STAT_B	Bit 22	BG# - (Processor 3)
STAT_B	Bit 23	BR# - (Processor 3)
CLK	J	SYSCLK

MPC744X/5X Label Bit Definitions

The following table shows the signals for each of the main logic analyzer labels (ADDR, ADDR_B, DATA, DATA_B, STAT, STAT_B, and CLK).

NOTE:

When using the MPXbus mode with the MPC7410/4X/5X, the inverse assembler requires an ODT signal from the system arbiter. **The ODT signal must be designed into your system arbiter.** The ODT signal is used to initialize the inverse assembler so that it can match address tenures to their respective data tenures. Please see "Using the MPXbus Tool" on page 98 for details.

MPC744X/5X Label Bit Definitions

Label Name	Bit Number	Signal Name
ADDR	Bit 0	A[35] (LSB)
ADDR	..	...
ADDR	Bit 31	A[4]
ADDR_B	Bit 0	A[3]
ADDR_B	Bit 1	A[2]
ADDR_B	Bit 2	A[1]
ADDR_B	Bit 3	A[0] (MSB)
DATA	Bit 0	D[31]
DATA	...	...
DATA	Bit 31	D[0] (MSB)
DATA_B	Bit 0	D[63] (LSB)
DATA_B	...	...
DATA_B	Bit 31	D[32]
STAT	Bit 0	WT#
STAT	Bit 1	TBST#
STAT	Bit 2	TT[4]
STAT	Bit 3	TT[3]
STAT	Bit 4	TT[2]
STAT	Bit 5	TT[1]
STAT	Bit 6	TT[0]
STAT	Bit 7	DX#
STAT	Bit 8	TSIZ[2]
STAT	Bit 9	TSIZ[1]
STAT	Bit 10	TSIZ[0]
STAT	Bit 11	TEA#

Label Name	Bit Number	Signal Name
STAT	Bit 12	ARTRY#
STAT	Bit 13	ODT[4]
STAT	Bit 14	ODT[3]
STAT	Bit 15	ODT[2]
STAT	Bit 16	ODT[1]
STAT	Bit 17	ODT[0]
STAT	Bit 18	DTI[4] - (Processor 0)
STAT	Bit 19	DTI[3] - (Processor 0)
STAT	Bit 20	DTI[2] - (Processor 0)
STAT	Bit 21	DTI[1] - (Processor 0)
STAT	Bit 22	DTI[0] - (Processor 0)
STAT	Bit 23	DRDY# - (Processor 0)
STAT	Bit 24	HIT# - (Processor 0)
STAT	Bit 25	DBG# - (Processor 0)
STAT	Bit 26	BG# - (Processor 0)
STAT	Bit 27	BR# - (Processor 0)
STAT	Bit 28	SYSCLK
STAT	Bit 29	TS#
STAT	Bit 30	TA#
STAT	Bit 31	AACK#
STAT_B	Bit 0	DTI[4] - (Processor 1)
STAT_B	Bit 1	DTI[3] - (Processor 1)
STAT_B	Bit 2	DTI[2] - (Processor 1)
STAT_B	Bit 3	DTI[1] - (Processor 1)
STAT_B	Bit 4	DTI[0] - (Processor 1)
STAT_B	Bit 5	DRDY# - (Processor 1)
STAT_B	Bit 6	HIT# - (Processor 1)
STAT_B	Bit 7	DBG# - (Processor 1)
STAT_B	Bit 8	BG# - (Processor 1)
STAT_B	Bit 9	BR# - (Processor 1)
STAT_B	Bit 10	DTI[4] - (Processor 2)
STAT_B	Bit 11	DTI[3] - (Processor 2)
STAT_B	Bit 12	DTI[2] - (Processor 2)
STAT_B	Bit 13	DTI[1] - (Processor 2)
STAT_B	Bit 14	DTI[0] - (Processor 2)
STAT_B	Bit 15	DRDY# - (Processor 2)
STAT_B	Bit 16	HIT# - (Processor 2)

Label Name	Bit Number	Signal Name
STAT_B	Bit 17	DBG# - (Processor 2)
STAT_B	Bit 18	BG# - (Processor 2)
STAT_B	Bit 19	BR# - (Processor 2)
STAT_B	Bit 20	DTI[4] - (Processor 3)
STAT_B	Bit 21	DTI[3] - (Processor 3)
STAT_B	Bit 22	DTI[2] - (Processor 3)
STAT_B	Bit 23	DTI[1] - (Processor 3)
STAT_B	Bit 24	DTI[0] - (Processor 3)
STAT_B	Bit 25	DRDY# - (Processor 3)
STAT_B	Bit 26	HIT# - (Processor 3)
STAT_B	Bit 27	DBG# - (Processor 3)
STAT_B	Bit 28	BG# - (Processor 3)
STAT_B	Bit 29	BR# - (Processor 3)
CLK	J	SYSCLK

Using the MPXbus Tool

The MPC7410/4X/5X MPXbus inverse assembler (E8135A) requires the use of the MPXbus Tool. The MPXbus Tool is used to align address tenures to the associated data tenures. This is done using the Outstanding Data Tenures (ODT) signal, **which must be designed into your system arbiter**.

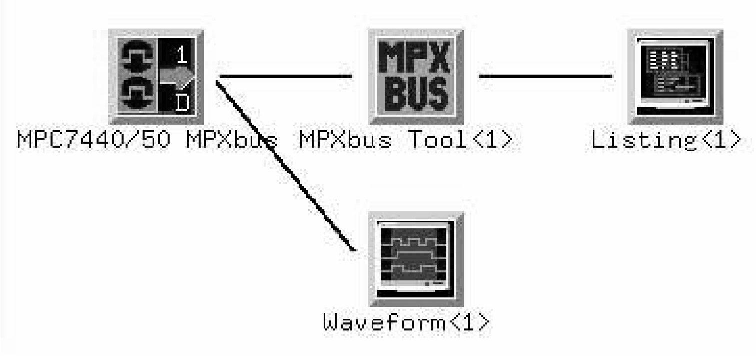
The ODT signal is used to create a logic analyzer version of the MPC7410/4X/5X processor's internal Data Tenure Queue (DTQ). This synthesized queue is then used in the same way as the MPC7410/4X/5X processor's internal DTQ—saving the address tenure data and retrieving it from the queue using the Data Transaction Index (DTI) signal.

NOTE: The ODT signal is provided in the MPX bus specification to assist an inverse assembler in correlating address tenures with their corresponding data tenures. The ODT signal is an attribute driven by the arbiter whose only sink is a logic analyzer. The ODT attribute is driven by an arbiter the same cycle as bus grant is asserted to a processor. The ODT value is equal to the number of data tenures outstanding for the processor receiving the bus grant. The ODT value is 'zero' for zero outstanding data tenures, 'one' for one outstanding data tenure, etc., up to the maximum number of outstanding data tenures supported by the system. (With this approach, we need to be able to indicate values of 0-8 for an 8-entry DTQ - which requires 4 bits.)

See Also See the "System Design Information" section of Motorola's *MPC74XX RISC Microprocessor Hardware Specifications* for more detailed information about implementing the ODT signal in your system.

Loading an MPC7410/4X/5X MPXbus configuration file will cause the logic analyzer to automatically load the MPXbus Tool onto the workspace along with the instrument, a Listing display, and a Waveform display. The MPC7410/4X/5X MPXbus inverse assembler will not load without the MPXbus tool.

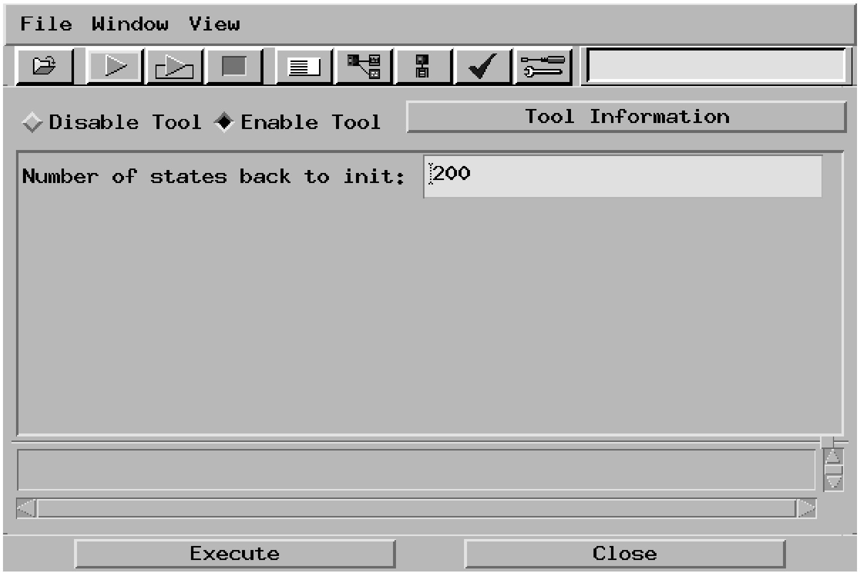
NOTE: The MPXbus Tool creates two new labels, STAT_R and ADDR_R, containing the information needed to associate the address tenures with the data tenures. These labels are passed into the Listing window for use in the MPXbus inverse assembler. The MPXbus inverse assembler requires these signals and will not load without the MPXbus tool present.



Setting the MPXbus tool preferences

The MPXbus Tool is designed to operate transparently, without user intervention; however, the MPXbus Tool does have preferences that you may modify if you desire.

You can access the MPXbus Tool window by selecting the MPXbus Tool icon and selecting Display....



Number of states back to init: 200

Number of states back to init. The MPXbus Tool must initialize the logic analyzer's synthesized Data Tenure Queue (DTQ) with default (invalid) data at the start of each execution of the tool (each page fetch). To minimize the number of unknown address states that appear in the Listing display, the MPXbus Tool goes back some number of states before the beginning of the page in order to initialize the DTQ. The Number of states back to init preference should be set to the average maximum number of states that it takes to service an outstanding data tenure. This will allow the MPXbus tool to fetch out all of the invalid addresses from the synthesized DTQ before disassembling the states that are going to be displayed in the Listing window. If you set the Number of states back to init too small, you may see a lot of unknown address states. These states will appear as a 'data tenure' state with no address and no mnemonic disassembly. Setting the Number of states back to init too high can impede the MPXbus Tool's performance by significantly increasing the number of states that must be disassembled per page fault.

Loading Symbol Information

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

The Agilent Technologies 16700-series logic analysis system lets you assign user-defined symbol names to particular label values, or you can download symbols from certain object file formats.

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

Three symbol sources may be used in the logic analyzer:

- Predefined MPC7410/4X/5X symbols
- User-defined symbols
- Object-file symbols

To view predefined MPC7410/4X/5X symbols

The MPC7410/4X/5X logic analyzer configuration files include predefined symbols. These symbols appear along with the user-defined symbols in the logic analyzer.

To view the predefined symbols:

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

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

To create user defined symbols

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

User-defined symbols are saved with logic analyzer configurations.

To create user-defined symbols:

1 Open the logic analyzer's Setup window.

2 Select the **Symbol** tab.

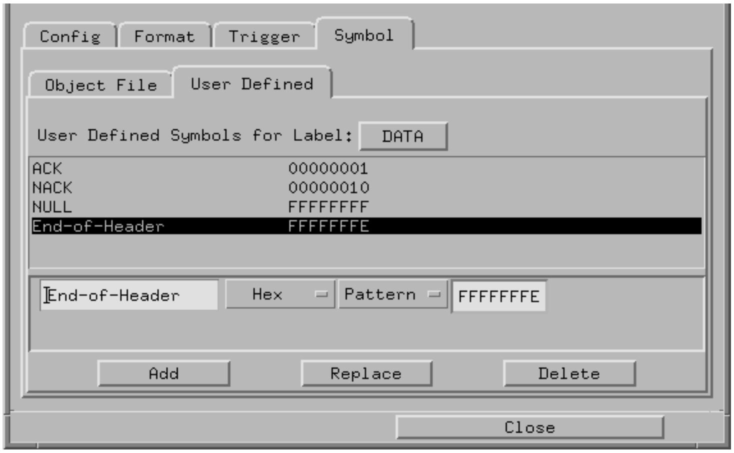
3 Select the **User Defined** tab.

4 Choose a label name from the **Label** list.

5 Enter the new symbol name and value.

6 Select **Add**.

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



User Defined Symbols for Label: DATA

```
ACK                  00000001
NACK                 00000010
NULL                 FFFFFFFF
End-of-Header        FFFFFFFE
```

End-of-Header   Hex   Pattern   FFFFFFFE

To load object file symbols

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

If your compiler generates files in a format that the logic analyzer doesn't understand, you can use a General-Purpose ASCII (GPA) symbol file (see Chapter 9, "General-Purpose ASCII (GPA) Symbol File Format," beginning on page 167).

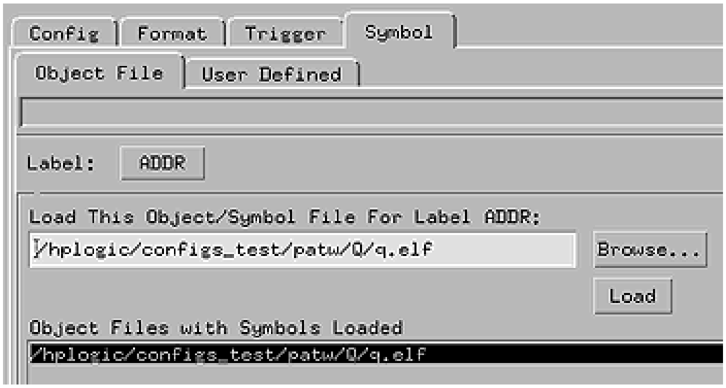
To load symbols in the 16700-series logic analysis system:

1 Open the logic analyzer module's **Setup** window.

2 Select the **Symbol** tab.

3 Select the **Object File** tab.

4 Make sure the label is ADDR, then select object files and load their symbol information.



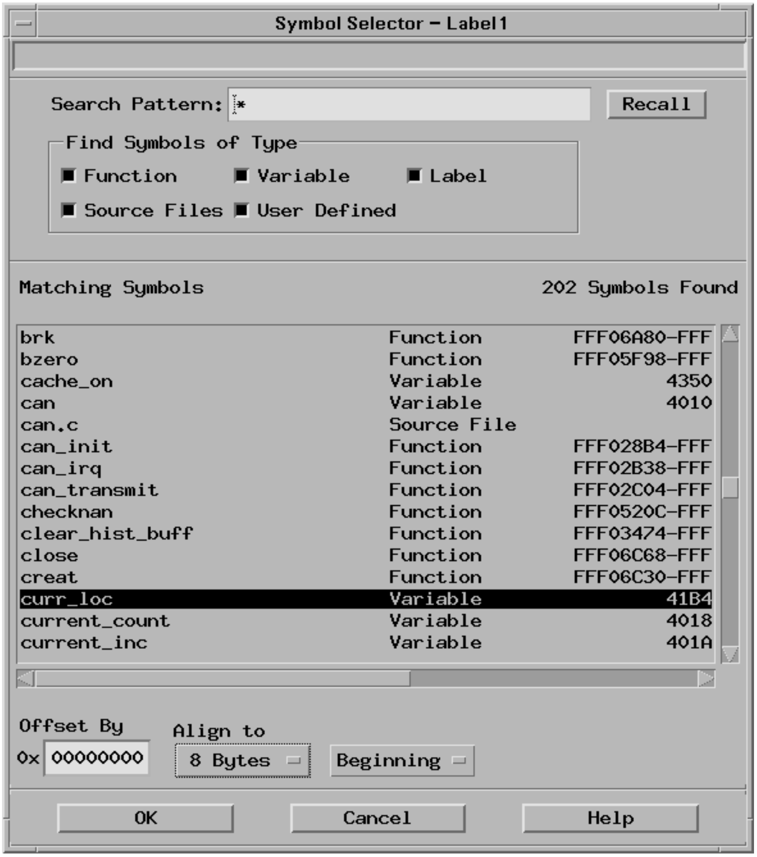
Label: ADDR

Load This Object/Symbol File For Label ADDR:
/hplogic/configs_test/patw/Q/q.elf

Object Files with Symbols Loaded
/hplogic/configs_test/patw/Q/q.elf

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

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



Symbol Selector – Label1

Search Pattern: *

Find Symbols of Type

■ Function ■ Variable ■ Label

■ Source Files ■ User Defined

Matching Symbols 202 Symbols Found

brk	Function	FFF06A80-FFF
bzero	Function	FFF05F98-FFF
cache_on	Variable	4350
can	Variable	4010
can.c	Source File	
can_init	Function	FFF028B4-FFF
can_irq	Function	FFF02B38-FFF
can_transmit	Function	FFF02C04-FFF
checknan	Function	FFF0520C-FFF
clear_hist_buff	Function	FFF03474-FFF
close	Function	FFF06C68-FFF
creat	Function	FFF06C30-FFF
curr_loc	Variable	41B4
current_count	Variable	4018
current_inc	Variable	401A

Offset By 0x 00000000

Align to 8 Bytes Beginning

See Also See “To compensate for relocated code” on page 113 for more information.

Symbol use requirements

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

An accurate bus trace

The E8170B and E8135A inverse assemblers provides MPC7410/4X/5X 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

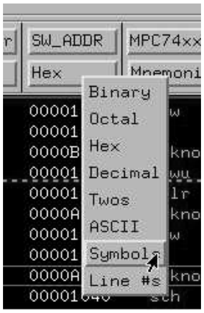
The MPC7410/4X/5X inverse assembler decodes captured data into program counter (PC) addresses (also known as software addresses) and assembly language mnemonics.

A symbol file

You need an object file containing symbolic debug information in a format the logic analyzer understands. Alternatively, you can use a General Purpose ASCII (GPA) symbol file (see page 167).

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.

.

6

Capturing Processor Execution

The normal steps in using the logic analyzer are:

- Configure the logic analyzer.

  See Chapter 5, “Configuring the Logic Analyzer,” beginning on page 85.

- Format labels for the logic analyzer channels (that is, map logic analyzer channels to target system signal names).

  The logic analyzer is configured and labels are created (formatted) for the logic analysis channels when configuration files are loaded. See Chapter 5, “Configuring the Logic Analyzer,” beginning on page 85.

- Load symbols from the program’s object file.

  You can load program object file symbols into the logic analyzer when configuring it. See “To load object file symbols” on page 103.

- Set up the trigger, and run the measurement.

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

- Display the captured data.

  See Chapter 7, “Displaying Captured MPC7410/4X/5X Execution,” beginning on page 119 for information on displaying captured data.

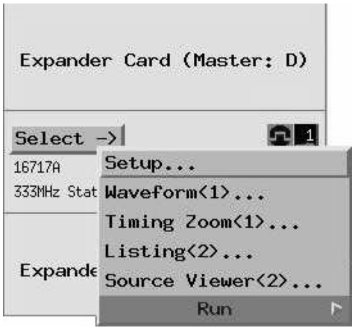
To Set Up Logic Analyzer Triggers

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

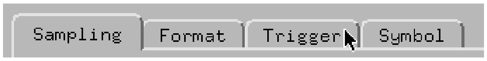
You can also specify which states that are stored in the logic analyzer. The Trigger sequence is set up by the software to store all states.

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

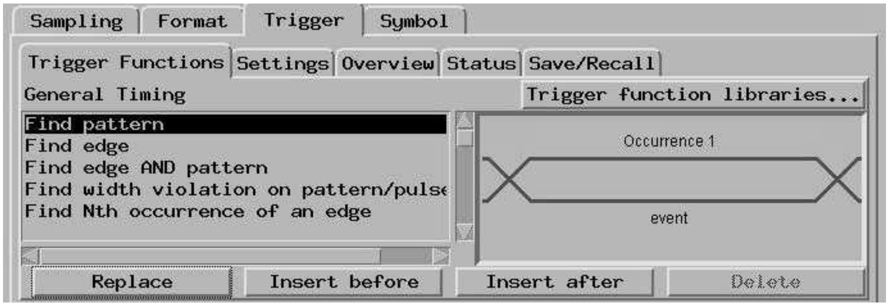
1 Open the logic analyzer's **Setup** window.



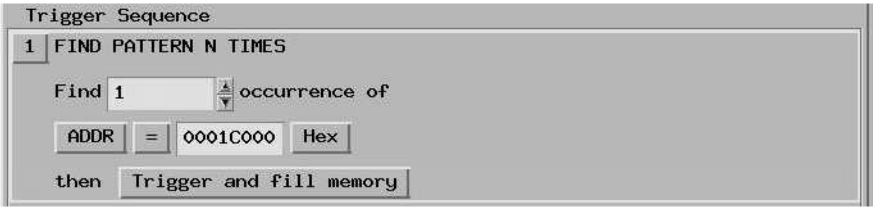
2 Select the **Trigger** tab.



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



4 Set up the trigger sequence.



Trigger Sequence

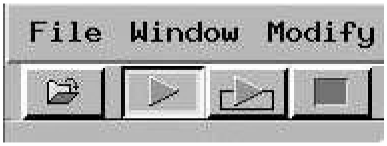
1 FIND PATTERN N TIMES

Find 1 occurrence of

ADDR = 0001C000 Hex

then Trigger and fill memory

5 Run the measurement.



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

To Setup Trigger Alignment and Offset for Symbols and Source Code

When setting up trigger specifications to capture MPC7410/4X/5X execution:

- Use the logic analyzer trigger alignment to avoid missed triggers.
- Use the logic analyzer address offset to compensate for relocated code.

Using trigger alignment

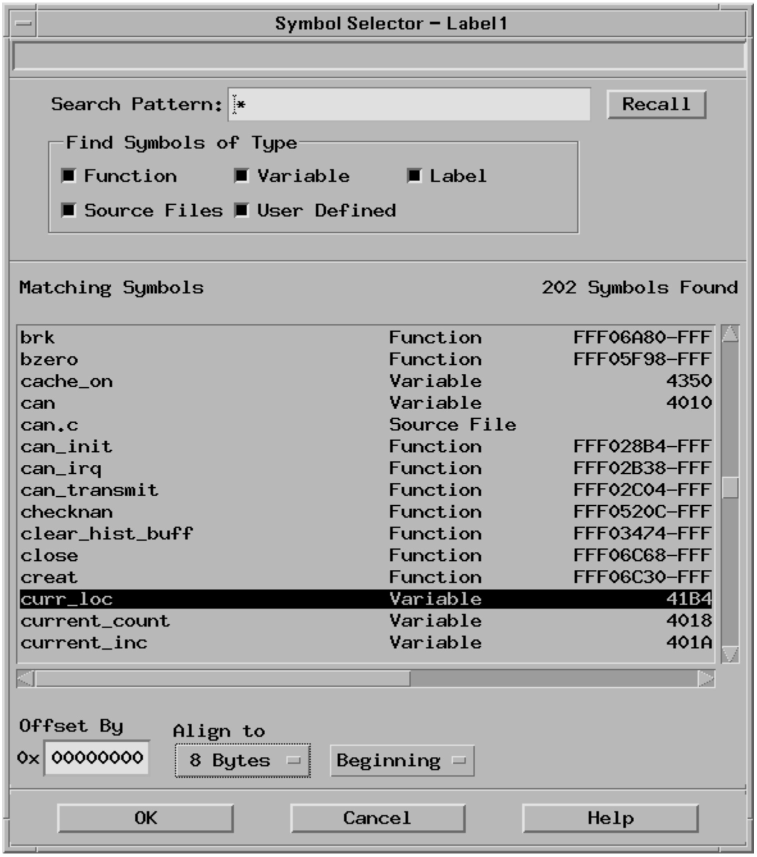
The MPC7410/4X/5X 64-bit data bus can cause missed triggers on some instructions. You should use an 8-byte alignment to avoid missed triggers. Instructions for the MPC7410/4X/5X are 32 bits long and must be located on even address boundaries. This means that an instruction will often be fetched as the 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 using the Align To menu in the Symbol Selector dialog.



Symbol Selector – Label1

Search Pattern: *

Find Symbols of Type

■ Function ■ Variable ■ Label
■ Source Files ■ User Defined

Matching Symbols 202 Symbols Found

brk	Function	FFF06A80-FFF
bzero	Function	FFF05F98-FFF
cache_on	Variable	4350
can	Variable	4010
can.c	Source File	
can_init	Function	FFF028B4-FFF
can_irq	Function	FFF02B38-FFF
can_transmit	Function	FFF02C04-FFF
checknan	Function	FFF0520C-FFF
clear_hist_buff	Function	FFF03474-FFF
close	Function	FFF06C68-FFF
creat	Function	FFF06C30-FFF
curr_loc	Variable	41B4
current_count	Variable	4018
current_inc	Variable	401A

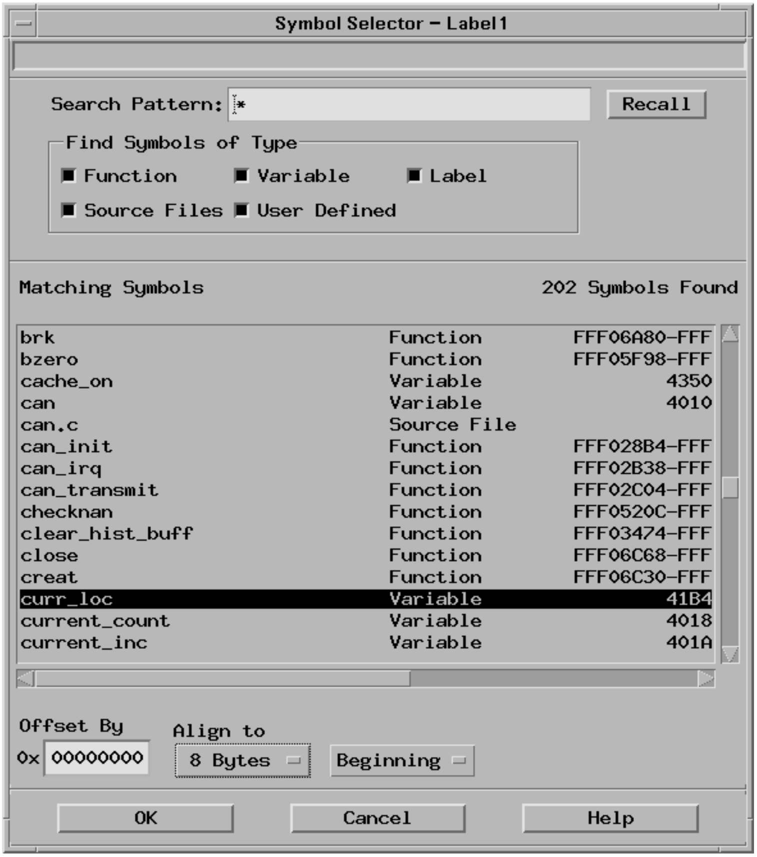
Offset By 0x 00000000

Align to 8 Bytes Beginning

For the MPC7410/4X/5X with 64-bit (8-byte) wide program memory, use 8-byte alignment. Eight-byte alignment will change the least significant three bits ($2^3 = 8$) of the trigger. Address 84 with 8-byte alignment results in a trigger address range of 80 through 87 for some logic analyzers (3 don't care bits), or an address of 80 on the other analyzers (three 0 bits). Note that either of these triggers would catch line #14 in the example above.

To compensate for relocated code

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



Symbol Selector – Label1

Search Pattern: *

Find Symbols of Type

■ Function ■ Variable ■ Label

■ Source Files ■ User Defined

Matching Symbols 202 Symbols Found

brk	Function	FFF06A80-FFF
bzero	Function	FFF05F98-FFF
cache_on	Variable	4350
can	Variable	4010
can.c	Source File	
can_init	Function	FFF028B4-FFF
can_irq	Function	FFF02B38-FFF
can_transmit	Function	FFF02C04-FFF
checknan	Function	FFF0520C-FFF
clear_hist_buff	Function	FFF03474-FFF
close	Function	FFF06C68-FFF
creat	Function	FFF06C30-FFF
curr_loc	Variable	41B4
current_count	Variable	4018
current_inc	Variable	401A

Offset By 0x 00000000

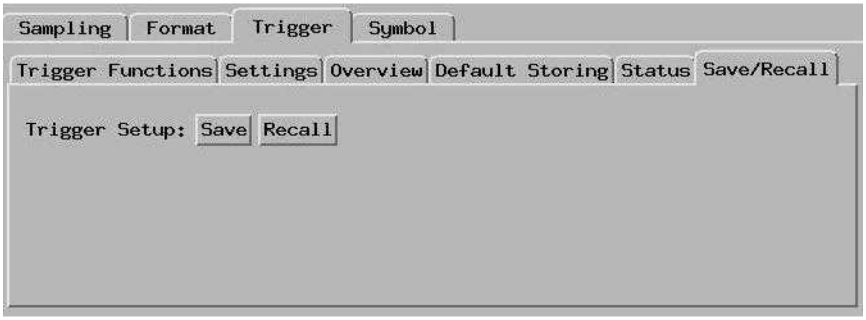
Align to 8 Bytes Beginning

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

To adjust for prefetches, use a trigger offset of 0x8 (prefetch queue depth) to avoid triggering on prefetched instructions. Note that this is not a foolproof scheme, since this may result in a missed trigger if a branch takes place between the base address and the offset address. For the MPC7410/4X/5X, an offset of 8 is large enough to overcome the prefetch queue.

Using the Saved Trigger Specifications for MPXbus

Logic analyzer configuration files for the MPC7410/4X/5X MPXbus inverse assembler contain saved trigger specifications. You can recall these trigger specifications using the Save/Recall tab under the Setup window's Trigger tab.



Anystate

This is the default trigger specification. Any captured state will trigger the logic analyzer and any state after that is stored. It gives you and easy way to return to the default.

Address Trigger (any processor)

This trigger specification will trigger on AACK=0 and the specified address, independent of which processor has control of the bus. Replace the ADDR value with the address you wish to trigger on.

Address Trigger (specific processor) - 3 or more cards required

This trigger specification will trigger on AACK=0 and the specified address, based upon which processor has control of the bus. In trigger level 1, replace the BG (Bus Grant) with the BG for the processor you would like to trigger on. In trigger level 2, replace the ADDR value with the address that you want to trigger on.

Data Trigger (any processor)

This trigger specification will trigger on TA=0 and the specified data, independent of which processor has control of the bus. Replace the DATA and/or DATA_B values with the data you wish to trigger on.

Data Trigger (specific processor) - 3 or more cards required

This trigger specification will trigger on TA=0 and the specified data, based upon which processor has control of the bus. In trigger level 1, replace the DBG (Data Bus Grant) with the DBG for the processor you are interested in triggering on. In trigger level 2, replace the DATA and/or DATA_B values with the data that you want to trigger on.

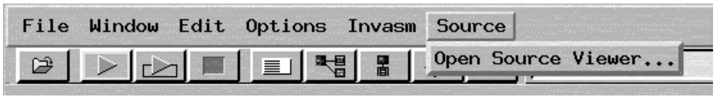
Hit Trigger (specific processor) - 3 or more cards required

This trigger specification will trigger on a HIT signal from the specified processor. Replace the HIT label with the HIT label for the processor you are interested in triggering on.

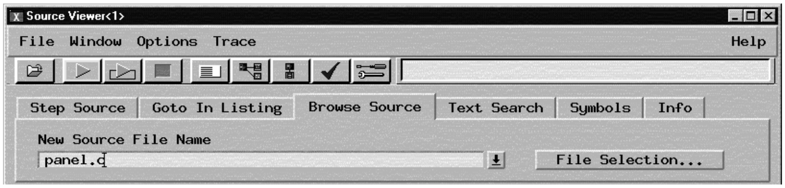
To Trigger on Source Code

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

1 Open the Source Viewer window.

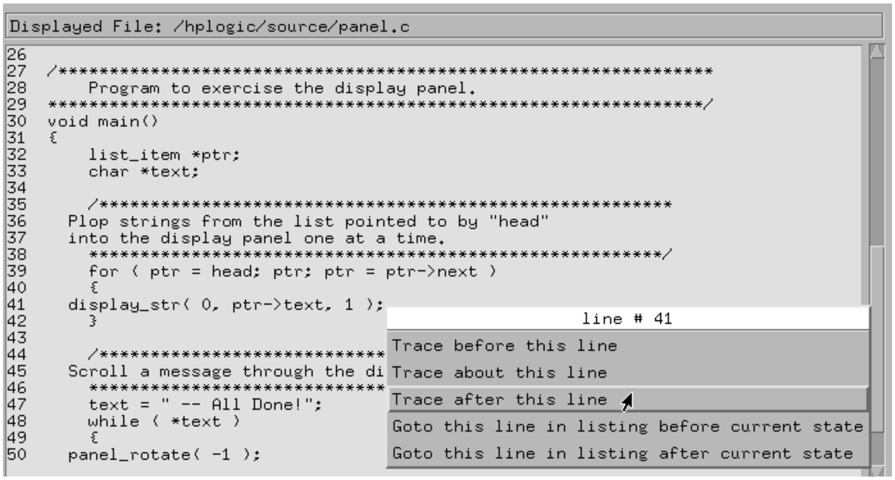


2 Browse the source file that contains the code you want to trigger on.



New Source File Name
panel.c

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

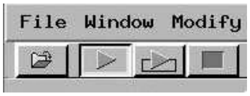


Displayed File: /hplogic/source/panel.c

```

/*******************************************************************
    Program to exercise the display panel.
*******************************************************************/
void main()
{
    list_item *ptr;
    char *text;

    /**********************************************************
  Plop strings from the list pointed to by "head"
  into the display panel one at a time.
    *********************************************************/
    for ( ptr = head; ptr; ptr = ptr->next )
    {
  display_str( 0, ptr->text, 1 );
    }

    /*****************************
  Scroll a message through the di
    *****************************
    text = " -- All Done!";
    while ( *text )
    {
  panel_rotate( -1 );
```

line # 41
Trace before this line
Trace about this line
Trace after this line
Goto this line in listing before current state
Goto this line in listing after current state

4 Run the measurement.



To avoid capturing library code execution

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

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

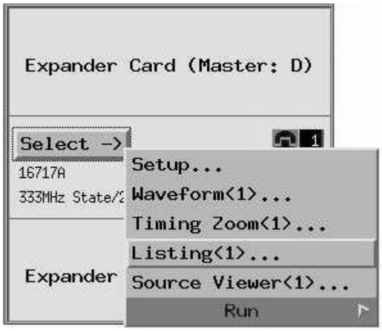
NOTE: Do not exclude states with important bus information (Bus Grants, Address Acknowledge, Transfer Acknowledge, Address Retry, HITs, etc.) as this may cause incorrect inverse assembly.

7

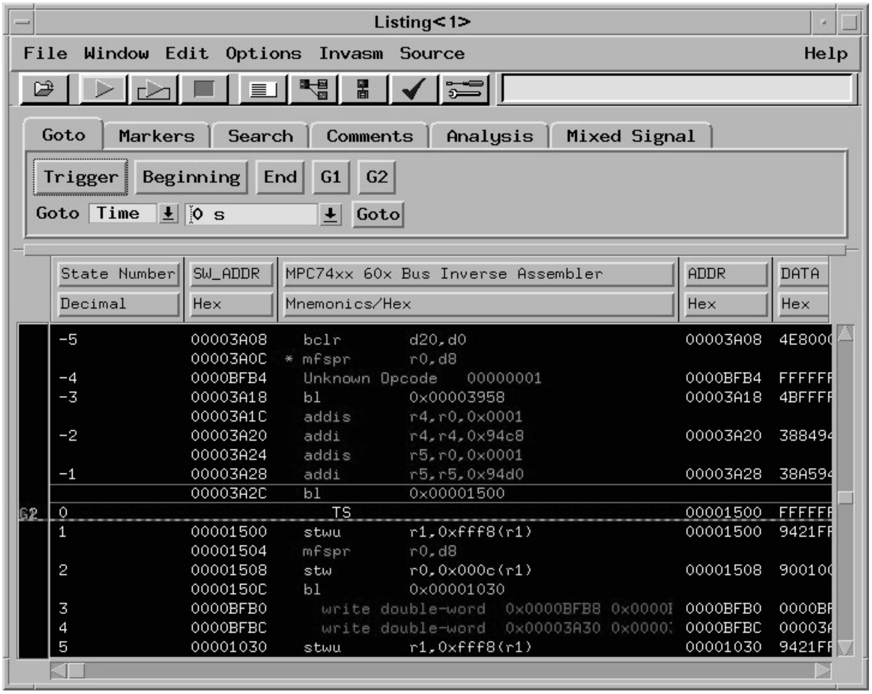
Displaying Captured MPC7410/4X/5X Execution

To Display Captured State Data

1 Open the Listing display window.



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


Listing<1>
File Window Edit Options Invasm Source Help
Goto Markers Search Comments Analysis Mixed Signal
Trigger Beginning End G1 G2
Goto Time 0 s Goto

```
State Number  SW_ADDR   MPC74xx 60x Bus Inverse Assembler          ADDR      DATA
Decimal       Hex       Mnemonics/Hex                              Hex       Hex
-5            00003A08    bclr      d20,d0                         00003A08  4E800C
              00003A0C  * mfspr     r0,d8
-4            0000BFB4    Unknown Opcode   00000001                0000BFB4  FFFFFF
-3            00003A18    bl        0x00003958                     00003A18  4BFFFF
              00003A1C    addis     r4,r0,0x0001
-2            00003A20    addi      r4,r4,0x94c8                   00003A20  388494
              00003A24    addis     r5,r0,0x0001
-1            00003A28    addi      r5,r5,0x94d0                   00003A28  38A594
              00003A2C    bl        0x00001500
G2 0                         TS                                    00001500  FFFFFF
1             00001500    stwu      r1,0xfff8(r1)                  00001500  9421FF
              00001504    mfspr     r0,d8
2             00001508    stw       r0,0x000c(r1)                  00001508  900100
              0000150C    bl        0x00001030
3             0000BFB0      write double-word  0x0000BFB8 0x0000I  0000BFB0  0000BF
4             0000BFBC      write double-word  0x00003A30 0x0000:  0000BFBC  00003A
5             00001030    stwu      r1,0xfff8(r1)                  00001030  9421FF
```

The inverse assembler is loaded when state configuration files are loaded, but it can also be loaded into a Listing display using the Invasm menu. The name of the 60x bus inverse assembler file is I74xx60XE, and the name of the MPXbus inverse assembler file is I74xxMPXE. The inverse assemblers are located in the /logic/ia directory.

CAUTION: The MPXbus inverse assembler (I74xxMPXE) requires the MPXbus Tool. You cannot load the MPXbus inverse assembler without first connecting the MPXbus Tool icon to the left of the Listing icon. See “Using the MPXbus Tool” on page 98 for more information.

See Also “To Use the Inverse Assembler” on page 122.

See Also “To Use the Inverse Assembler Filters” on page 135 for information on displaying or hiding certain types of microprocessor bus cycles.

To Use the Inverse Assembler

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

To use the Invasm menu

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

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

To load the inverse assembler

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

CAUTION: You should NOT switch between MPXbus and 60x bus inverse assemblers simply by loading the inverse assembler from the Load... menu.

If the MPXbus inverse assembler is loaded and you Load... the 60x bus inverse assembler, the MPXbus tool isn't needed. The inverse assembler will work, but it will not work as fast as it could.

If the 60x bus inverse assembler is loaded and you Load... the MPXbus inverse assembler, the MPXbus tool is needed but missing. The MPXbus inverse assembler will not load or work.

To switch between MPXbus and 60x bus, load a different configuration file, use the Setup Assistant, or use a different Listing window off of the same data set. See "Using the MPXbus Tool" on page 98 for more information.

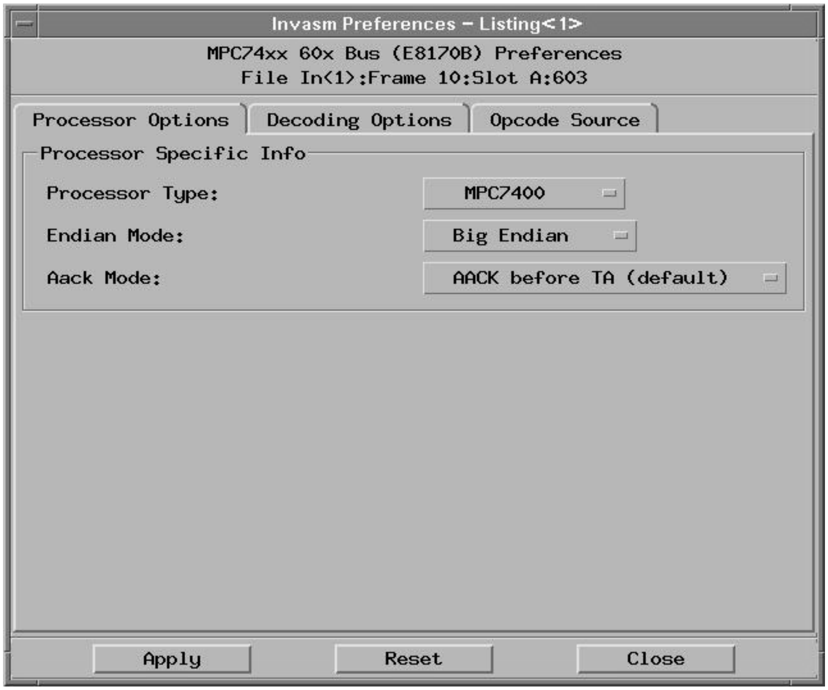
To Set the Inverse Assembler Preferences

The 60x bus inverse assembler (E8170B) Preferences window contains three tabs: Processor Options, Decoding Options, and Opcode Source.

The MPXbus inverse assembler (E8135A) Preferences window also contains three tabs: Target System Options, Decoding Options, and Opcode Source.

Processor options—60x bus inverse assembler

Inverse Assembler Processor Options



Invasm Preferences – Listing<1>

MPC74xx 60x Bus (E8170B) Preferences
File In<1>:Frame 10:Slot A:603

Processor Specific Info

Processor Type: MPC7400

Endian Mode: Big Endian

Aack Mode: AACK before TA (default)

Processor Type

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

Endian Mode

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

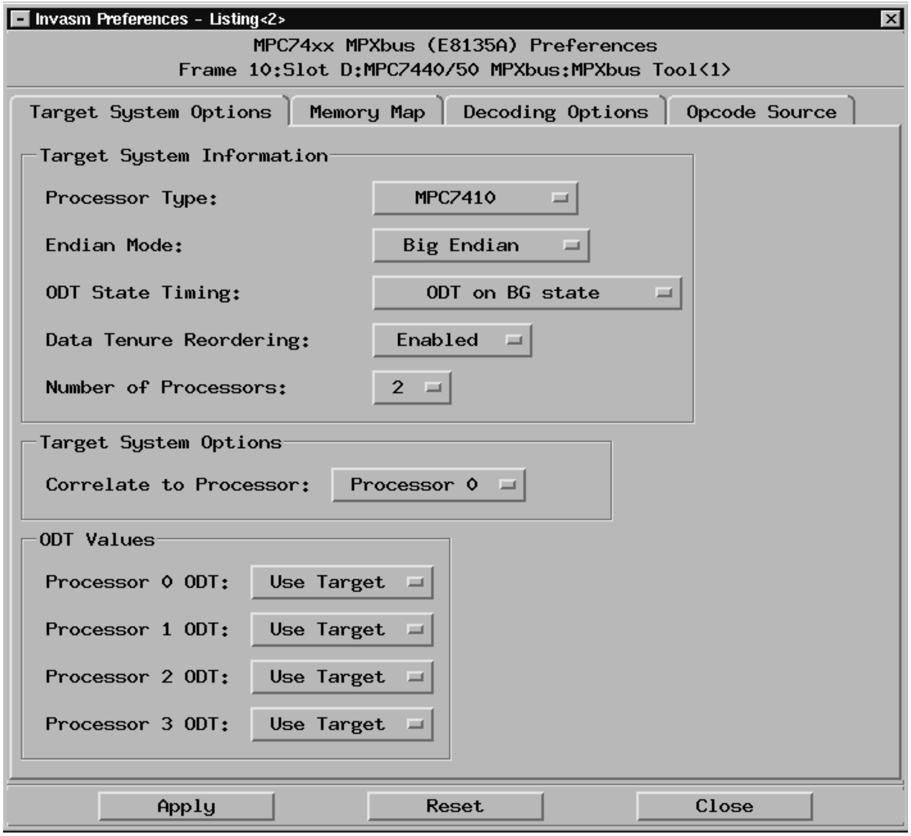
Little endian mode causes the instruction word from DL0...31 (DATA_B label; external address xxx4) to be dispatched before the instruction word from DH0...31 (DATA label; external address xxx0). It also causes byte and half-word reads and writes to appear on the opposite side of the bus and swaps the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

AACK Mode

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

Target system options—MPXbus inverse assembler

Inverse Assembler Target System Options



Invasm Preferences - Listing<2>

MPC74xx MPXbus (E8135A) Preferences
Frame 10:Slot D:MPC7440/50 MPXbus:MPXbus Tool<1>

Target System Information

Processor Type: MPC7410
Endian Mode: Big Endian
ODT State Timing: ODT on BG state
Data Tenure Reordering: Enabled
Number of Processors: 2

Target System Options

Correlate to Processor: Processor 0

ODT Values

Processor 0 ODT: Use Target
Processor 1 ODT: Use Target
Processor 2 ODT: Use Target
Processor 3 ODT: Use Target

Processor Type

The Processor Type option is used to setup the inverse assembler for the MPC74xx processor you are using.

Endian Mode

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

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.

ODT State Timing

The ODT State Timing selection is used to select the state in which the ODT signal is valid on your system. The default value is for the ODT signal to be driven on the same state as Bus Grant (BG). In certain systems, the ODT signal may need to be delayed to the state after the Bus Grant. Use this selection to set up the inverse assembler to sample the ODT signal on the appropriate state for your system.

See "Using the MPXbus Tool" on page 98 for more information about the ODT signal.

Data Tenure Reordering

The Data Tenure Reordering selection is used to tell the inverse assembler if your target allows for data tenure reordering. Most systems will allow data tenure reordering, and the default selection of "Enabled" should be used. Some target systems, however, leave the bus in a parked state with DBG constantly enabled. When DBG is parked, the inverse assembler is constantly assuming a new DBG and trying to match it with an address in the Data Tenure Queue - causing incorrect decode of data.

In order to support a DBG parked target, there are two requirements:

1. Data tenures must always be in order, since the DTI signal (which tells which address in the queue is the match for the ensuing data) is validated based upon the DBG timing. Without a change in the DBG state, DTI information cannot be determined.

2. You must use only one processor, since the DBGx signal determines which processor the ensuing data is intened for.

If your target has DBG parked and meets the two criteria, select "Disable" for the Data Tenure Reordering and select the Number of Processors as being "1". Then, the inverse assembler will ignore the DBG signal and decode based on this special case.

Number of Processors

The Number of Processors selection is used to setup the inverse assembler for the number of MPC7410/4X/5X processors that you are probing. For more than one processor, the STAT_B label is required.

Correlated to Processor

The Correlate to Processor selection is used to set which of the probed processors will be used for any Source Correlation and/or Cache-On trace reconstruction operations.

NOTE: To correlate to multiple processors simultaneously, you can bring up multiple Listing windows and set each Listing window's preferences to correlate to a different processor. See "To use Multiple Listing Windows for Multiple Processors" on page 139 for more information.

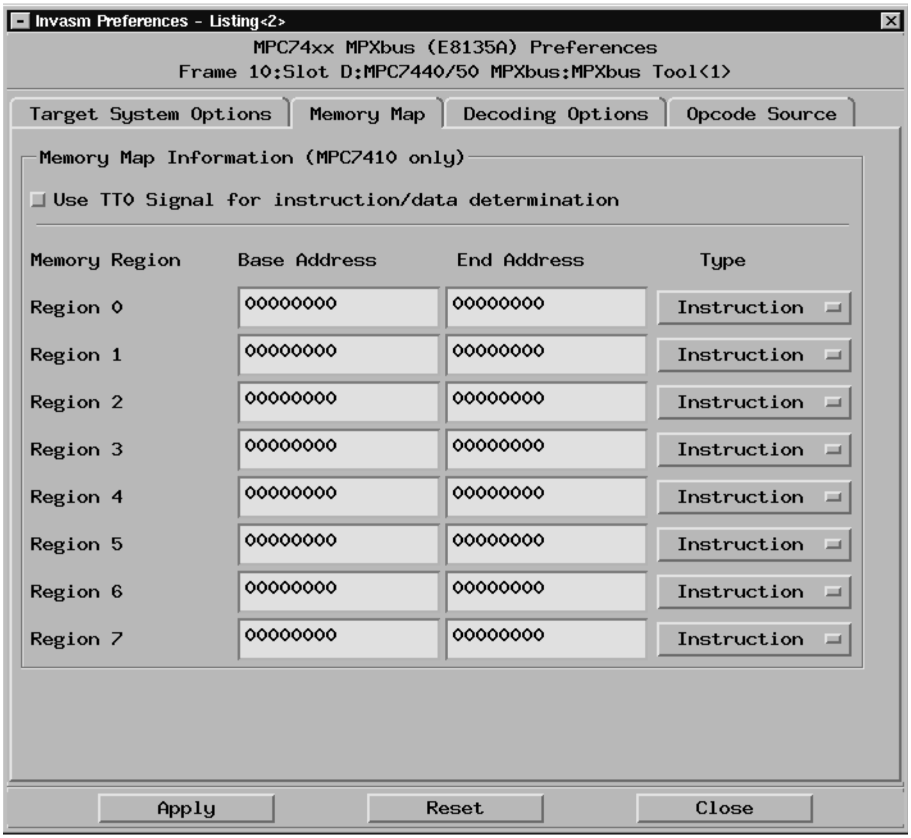
ODT Values

These selectors allow you to manually select the ODT value for each processor if your target system does not have the ODT signal. Since the ODT signal is unique to the processor and to each trace, and since there is no way to derive what the ODT value is for any given trace, using these selectors requires a guess-and-check iteration which must be re-done for every new run.

See Also See "Using the MPXbus Tool" on page 98 for more information about the ODT signal.

Memory Map—MPXbus inverse assembler for MPC7410

MPC7410 Memory Map Dialog



MPC74xx MPXbus (E8135A) Preferences
Frame 10:Slot D:MPC7440/50 MPXbus:MPXbus Tool<1>

Memory Map Information (MPC7410 only)

□ Use TTO Signal for instruction/data determination

Memory Region	Base Address	End Address	Type
Region 0	00000000	00000000	Instruction
Region 1	00000000	00000000	Instruction
Region 2	00000000	00000000	Instruction
Region 3	00000000	00000000	Instruction
Region 4	00000000	00000000	Instruction
Region 5	00000000	00000000	Instruction
Region 6	00000000	00000000	Instruction
Region 7	00000000	00000000	Instruction

In the MPC7410, the TTO signal is usually used to differentiate between data and instruction cycles when HIDO[IFFT]=1. However, if HIDO[IFFT]=0, then the TTO signal is used to differentiate between Read Atomic and Read operations. When the TTO signal is used in the latter mode, the inverse assembler cannot determine whether the cycle was data or instruction. Therefore, a Memory Map is needed to tell the inverse assembler which memory regions are data and which are instruction so that the inverse assemler can decode the cycles properly.

Use TTO signal for instruction/data determination

If your target uses TTO as instruction/data determinator (HIDO[IFFT]=1) then enable this option. When TTO is used in this manner, it can be used to differentiate between instruction cycles and data cycles and there is no need to input data into the Memory Map. If, however, your target uses TTO as a Read/Read Atomic determinator (HIDO[IFFT]=0) then disable this option and fill out the Memory Map. When TTO is used in this manner, it cannot be used to differentiate between instruction cycles and data cycles and therefore, the user must input the target memory configuration into the Memory Map.

Base Address, End Address

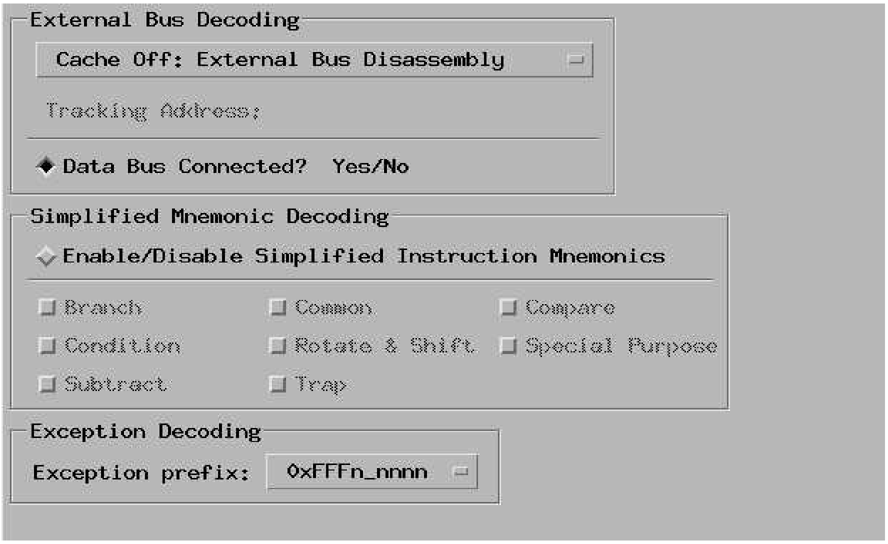
Specifies the starting and ending address of the memory region.

Type selector

Specifies whether the data in this memory region is of type instruction or data.

Decoding options

Inverse Assembler Decoding Options Dialog



External Bus Decoding

Cache Off: External Bus Disassembly

Tracking Address:

◆ Data Bus Connected? Yes/No

Simplified Mnemonic Decoding

◇ Enable/Disable Simplified Instruction Mnemonics

□ Branch □ Common □ Compare
□ Condition □ Rotate & Shift □ Special Purpose
□ Subtract □ Trap

Exception Decoding

Exception prefix: 0xFFFn_nnnn

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.

NOTE: Cache-on trace reconstruction can only be done for one processor. When using the MPXbus inverse assembler for multiple MPC7410/4X/5X processors, only the correlated processor will have opcodes displayed.

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 145.

Simplified Mnemonic Decoding

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

- Conditional traps and branches decode the condition mnemonically when possible. For some conditions which have no conventional mnemonics (for example, “signed less than or unsigned greater than”), the condition field is displayed in binary.
- The L bit is omitted as a compare operand. Instead, compares are decoded as “cmpw” (or “?cmpd”).
- “Add immediate” instructions with a negative immediate operand are decoded as subtract immediate (“subi”).
- “Subtract from” instructions subf and subfc are decoded as subtract instructions sub and subc with the operands exchanged so that “sub r3 r4 r5” is mnemonically interpreted as “r3 = r4 - r5.”
- ori r0 r0 0000 is decoded as “nop”.
- Add immediate and add immediate shifted instructions, addi and addis, with a null source register are decoded as load immediate and load immediate shifted, li and lis.
- or instructions with identical source registers are decoded as move

register, mr.

- nor instructions with identical source registers are decoded as not register, not.
- 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 mtcrf instruction field mask specifies the entire cr, it is decoded as mtcr.

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

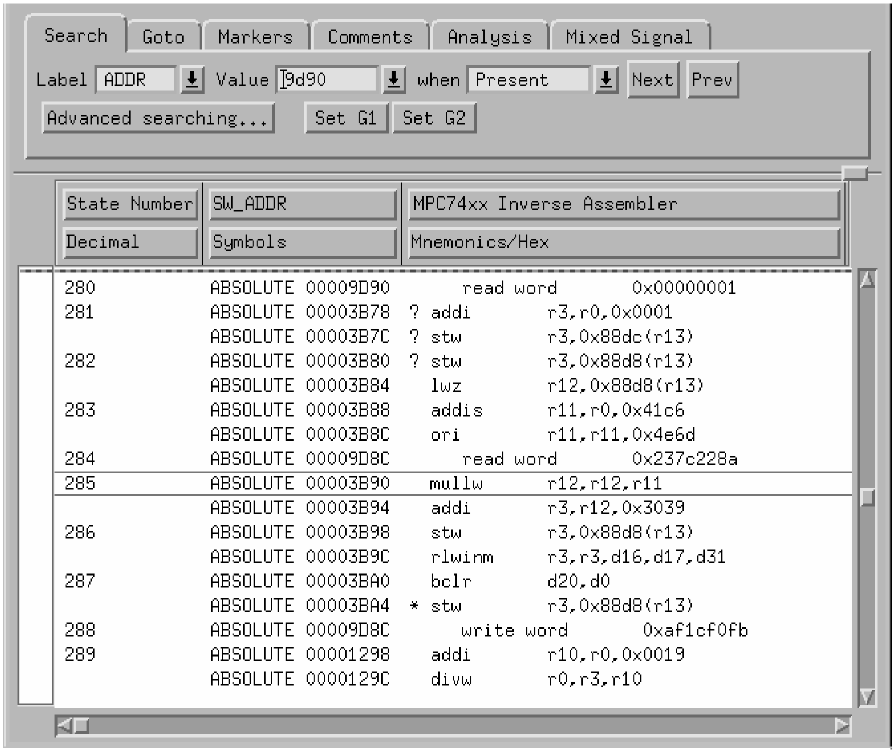
The inverse assembler supports the following extensions of dialect-sensitive instructions.

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	mtcrf %11111111,r5	mtcr 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

Exception Decoding

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

Listing Window Showing Trace with Data Bus Connected: Cache Off



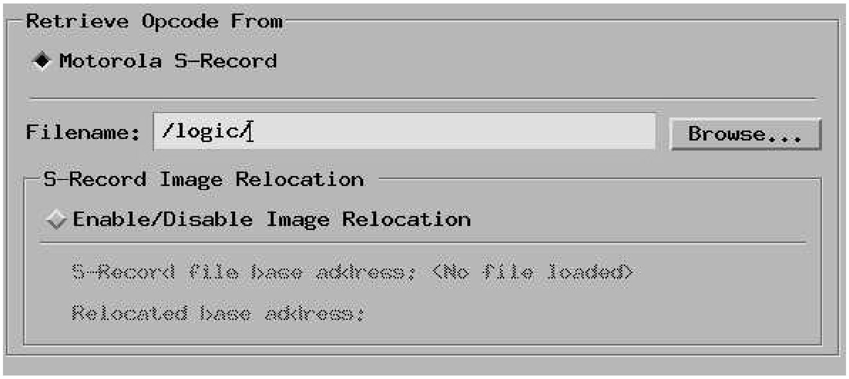
Label ADDR Value 9d90 when Present

```
State Number   SW_ADDR              MPC74xx Inverse Assembler
Decimal        Symbols              Mnemonics/Hex

280            ABSOLUTE 00009D90       read word       0x00000001
281            ABSOLUTE 00003B78  ? addi       r3,r0,0x0001
               ABSOLUTE 00003B7C  ? stw        r3,0x88dc(r13)
282            ABSOLUTE 00003B80  ? stw        r3,0x88d8(r13)
               ABSOLUTE 00003B84    lwz        r12,0x88d8(r13)
283            ABSOLUTE 00003B88    addis      r11,r0,0x41c6
               ABSOLUTE 00003B8C    ori        r11,r11,0x4e6d
284            ABSOLUTE 00009D8C       read word       0x237c228a
285            ABSOLUTE 00003B90    mullw      r12,r12,r11
               ABSOLUTE 00003B94    addi       r3,r12,0x3039
286            ABSOLUTE 00003B98    stw        r3,0x88d8(r13)
               ABSOLUTE 00003B9C    rlwinm     r3,r3,d16,d17,d31
287            ABSOLUTE 00003BA0    bclr       d20,d0
               ABSOLUTE 00003BA4  * stw        r3,0x88d8(r13)
288            ABSOLUTE 00009D8C       write word       0xaf1cf0fb
289            ABSOLUTE 00001298    addi       r10,r0,0x0019
               ABSOLUTE 0000129C    divw       r0,r3,r10
```

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

Opcode Source

Inverse Assembler Preferences Opcode Source Dialog



Retrieve Opcode From

◆ Motorola S-Record

Filename: /logic/

Browse...

S-Record Image Relocation

◇ Enable/Disable Image Relocation

S-Record file base address: <No file loaded>

Relocated base address:

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.

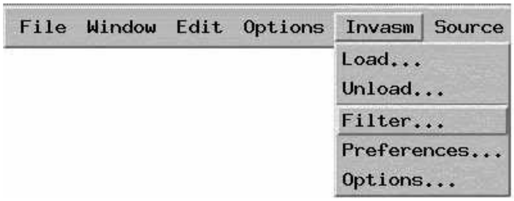
NOTE: Cache-on trace reconstruction can only be done for one processor. When using the MPXbus inverse assembler for multiple MPC7410/4X/5X processors, the S-Record loaded should be for the correlated processor.

S-Record image relocation

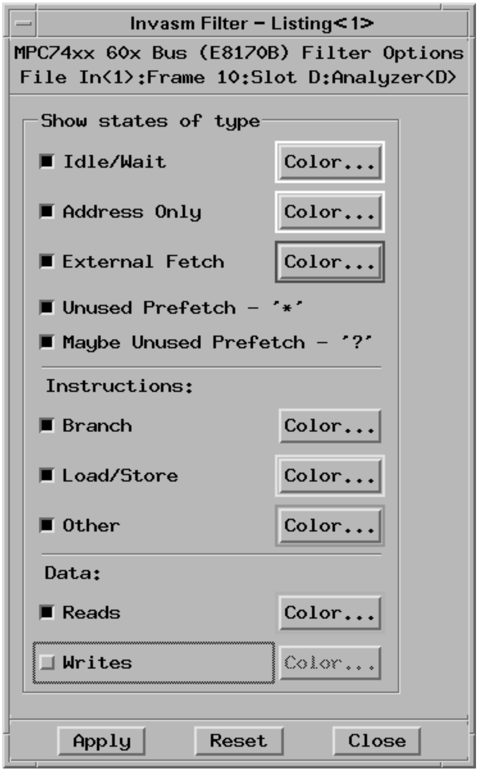
The Image Relocation portion of the dialog box allows you to relocate the SREC file to some other location in memory. This is useful when the loaded file is moved to some other location in memory. For example, the starting address in the SREC file is 1000. However, memory starting at 1000 is relocated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

To Use the Inverse Assembler Filters

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



60x Bus Filter Window



MPC74xx 60x Bus (E8170B) Filter Options
File In<1>:Frame 10:Slot D:Analyzer<D>

Show states of type

☒ Idle/Wait [Color...]
☒ Address Only [Color...]
☒ External Fetch [Color...]
☒ Unused Prefetch – '*'
☒ Maybe Unused Prefetch – '?'

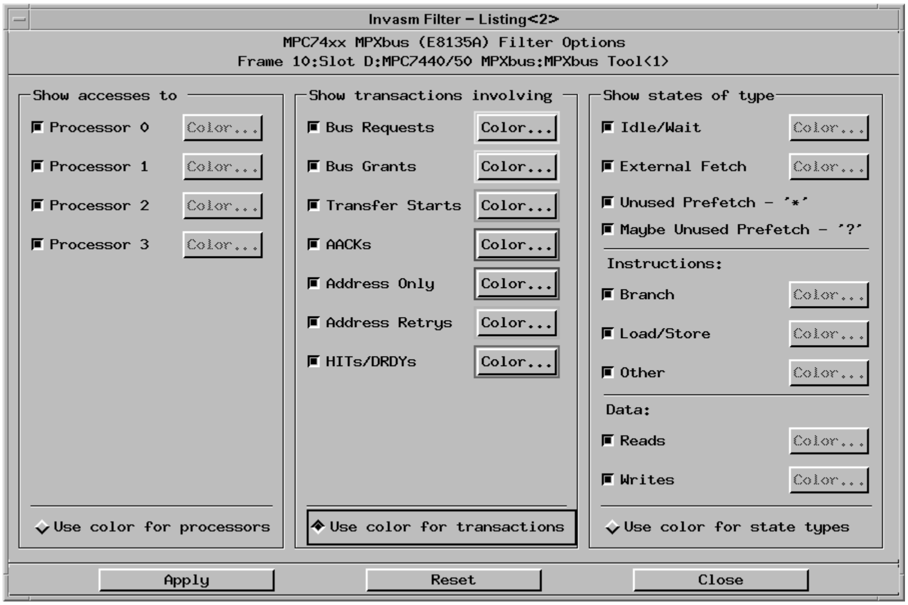
Instructions:

☒ Branch [Color...]
☒ Load/Store [Color...]
☒ Other [Color...]

Data:

☒ Reads [Color...]
□ Writes [Color...]

MPXbus Filter Window



Invasm Filter – Listing<2>

MPC74xx MPXbus (E8135A) Filter Options
Frame 10:Slot D:MPC7440/50 MPXbus:MPXbus Tool<1>

Show accesses to

☑ Processor 0
☑ Processor 1
☑ Processor 2
☑ Processor 3

◇ Use color for processors

Show transactions involving

☑ Bus Requests
☑ Bus Grants
☑ Transfer Starts
☑ AACKs
☑ Address Only
☑ Address Retrys
☑ HITs/DRDYs

◆ Use color for transactions

Show states of type

☑ Idle/Wait
☑ External Fetch
☑ Unused Prefetch – '*'
☑ Maybe Unused Prefetch – '?'

Instructions:

☑ Branch
☑ Load/Store
☑ Other

Data:

☑ Reads
☑ Writes

◇ Use color for state types

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

Filtering allows faster analysis in two ways:

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

You can also use color to distinguish between cycle types (when they are displayed). In the MPXbus inverse assembler (E8135A), color can be used for distinguishing between processors, transaction types, or cycle types, but only one at a time.

To Interpret the Inverse Assembler Output

Data formats

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

Most numerical data is displayed in hexadecimal, for example, “stwu r1,0xfff8(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 AltiVec instructions. When unimplemented opcodes are encountered, the listing displays “illegal opcode.”

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

Branch instructions

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

Overfetch marking

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

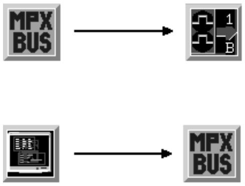
- When the processor executes a branch instruction, the instructions between the branch and the branch target are not executed. These instructions are indicated with an asterisk “*”, or if the bus trace is ambiguous, with a question mark “?”. If the instruction cache is enabled, the branch target may 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.

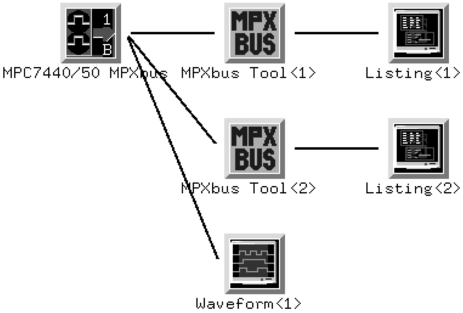
To use Multiple Listing Windows for Multiple Processors

You may display the same data in multiple Listing windows to aid in comparison and analysis. Each Listing window functions independently of the other. This means that each Listing window has its own filtering and source correlation capabilities. This can allow you to display the same data in different ways simultaneously for easy comparison. For example, you may have two Listing windows, each correlated to a different processor or with different filtering options set. Because the Listing windows will both be using the same data, they will be time correlated.

To add another Listing window to the same set of data, drag an MPXbus Tool icon onto the machine, then drag a Listing Icon onto the new MPXbus Tool Icon.



The workspace will now look like this:



The new MPXbus tool (MPXbus Tool <2>) and the new Listing window (Listing<2>) will be created using default preferences. You will need to manually set the new preferences to reflect your target system. See “To Set the Inverse Assembler Preferences” on page 123 and “Setting the MPXbus tool preferences” on page 100.

To Use Cache-On Trace Reconstruction

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

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

The branch exception is located at 0x00000D00 for an exception prefix MSR.IP=0 or 0xFFF00D00 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:

1 In the Decoding Options tab External Bus Decoding dialog:

a Set the cache-on mode

b Set data bus connected

c Provide the tracking address

2 In the Opcode Source tab:

a Load an S-Record executable file

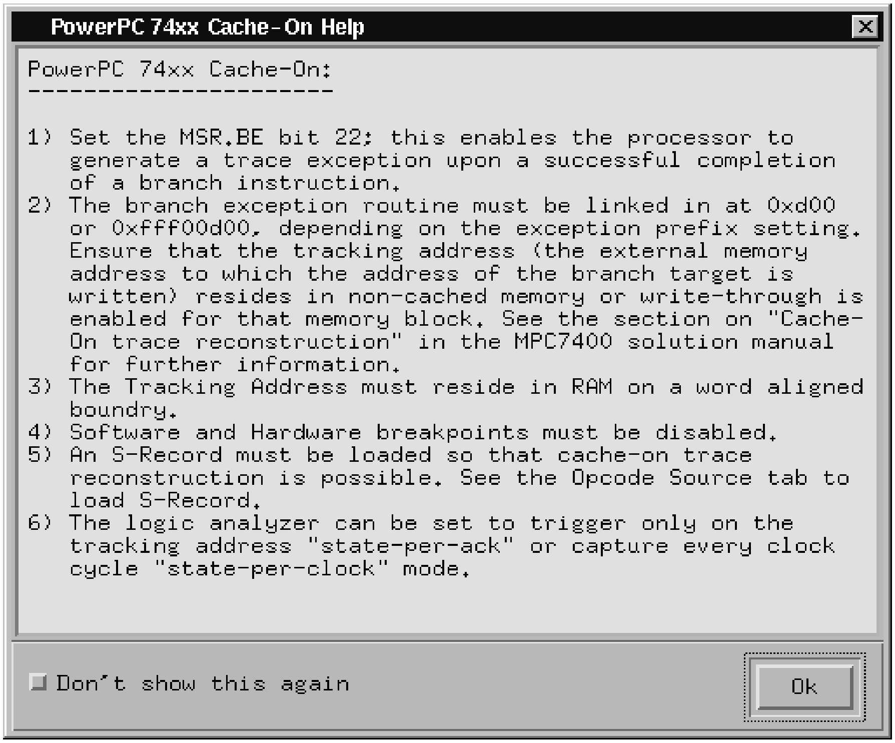
NOTE: Cache-on trace reconstruction can only be done for one processor. When using the MPXbus inverse assembler for multiple MPC7410/4X/5X processors, only the correlated processor will have opcodes displayed.

Minimizing effects of cache-on trace on system performance

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

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

Cache-on help dialog



```
PowerPC 74xx Cache-On:
----------------------

1) Set the MSR.BE bit 22; this enables the processor to
   generate a trace exception upon a successful completion
   of a branch instruction.
2) The branch exception routine must be linked in at 0xd00
   or 0xfff00d00, depending on the exception prefix setting.
   Ensure that the tracking address (the external memory
   address to which the address of the branch target is
   written) resides in non-cached memory or write-through is
   enabled for that memory block. See the section on "Cache-
   On trace reconstruction" in the MPC7400 solution manual
   for further information.
3) The Tracking Address must reside in RAM on a word aligned
   boundry.
4) Software and Hardware breakpoints must be disabled.
5) An S-Record must be loaded so that cache-on trace
   reconstruction is possible. See the Opcode Source tab to
   load S-Record.
6) The logic analyzer can be set to trigger only on the
   tracking address "state-per-ack" or capture every clock
   cycle "state-per-clock" mode.
```

□ Don't show this again

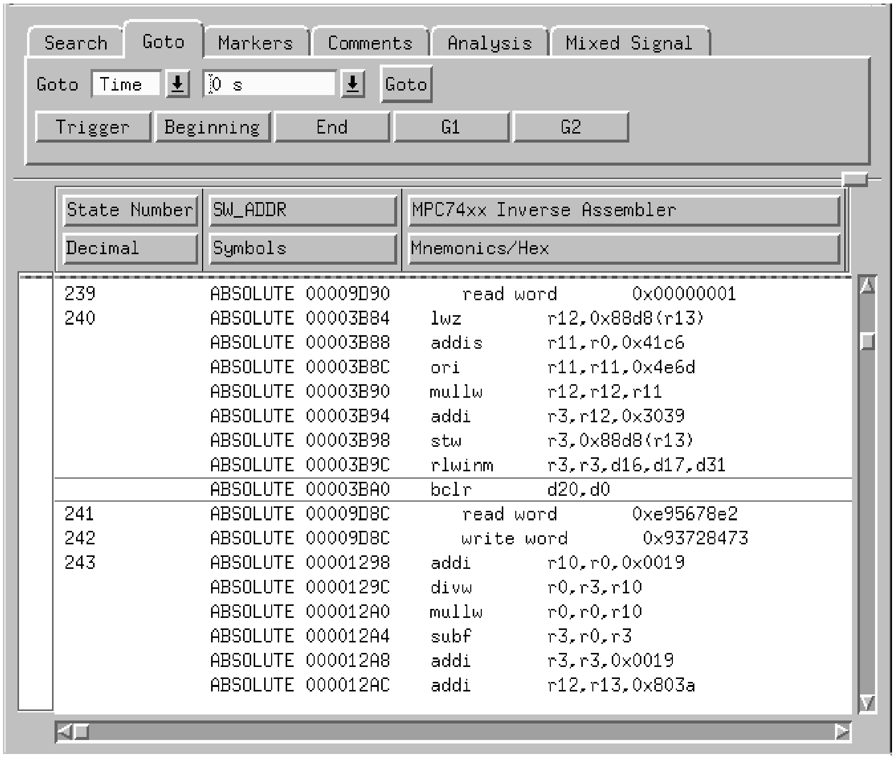
The Don't show this again button can be selected to prevent this dialog from appearing until the inverse assembler is loaded again.

To enable branch exception disassembly

The following trace shows cache-on execution using branch trace exception disassembly. See page 140 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



```
State Number    SW_ADDR                MPC74xx Inverse Assembler
Decimal         Symbols                Mnemonics/Hex

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.

NOTE: Cache-on trace reconstruction can only be done for one processor. When using the MPXbus inverse assembler for multiple MPC7410/4X/5X processors, only the correlated processor will have opcodes displayed.

To enable/disable the instruction cache on the MPC7410/4X/5X

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 code:

- Disable the instruction cache with the following code:

```
mfspr    r3, hid0
rlwinm   r3, r3, 0, 17, 15  # clear bit 16 (ICE)
mtspr    hid0, r3
isync
```

- To also disable the data cache use:

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

- To invalidate and disable both caches use:

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

- Enable the instruction cache with the following code:

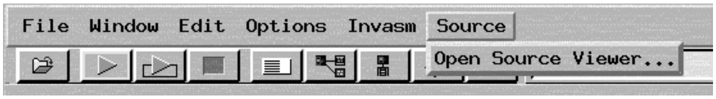
```
mfspr   r3, hid0
rlwinm  r3, r3, 1, 17, 15   # set ICE
mtspr   hid0, r3
isync
```

To View the Source Code Associated With Captured Data

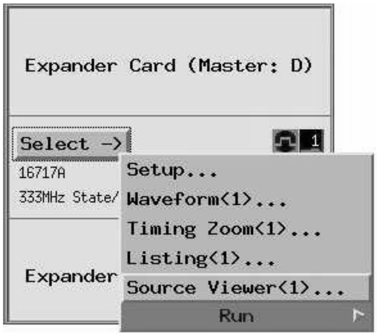
The B4620B Source Correlation Tool Set lets you view the high-level source code associated with captured data and set up triggers based on source code.

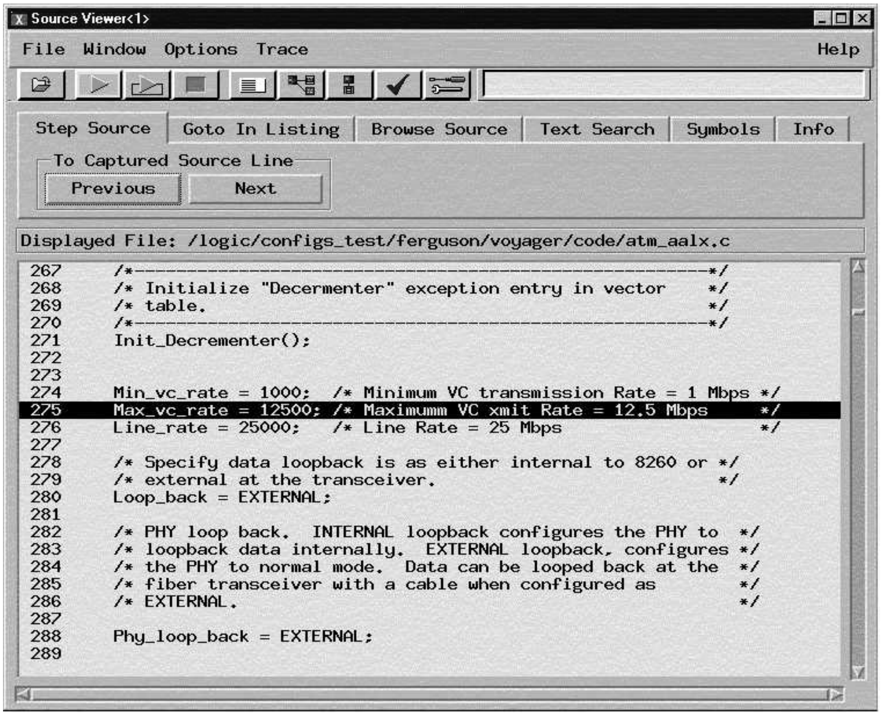
The source correlation tool set correlates the logic analyzer's address label with a line of high-level source code whose address, symbol name, file name, and line numbers are described in a symbol file downloaded to the logic analyzer (see “To load object file symbols” on page 103).

- 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.





Displayed File: /logic/configs_test/ferguson/voyager/code/atm_aalx.c

```
/*-----------------------------------------------------------*/
/* Initialize "Decermenter" exception entry in vector    */
/* table.                                                */
/*-----------------------------------------------------------*/
Init_Decrementer();


Min_vc_rate = 1000;  /* Minimum VC transmission Rate = 1 Mbps */
Max_vc_rate = 12500; /* Maximumm VC xmit Rate = 12.5 Mbps     */
Line_rate = 25000;   /* Line Rate = 25 Mbps                   */

/* Specify data loopback is as either internal to 8260 or */
/* external at the transceiver.                           */
Loop_back = EXTERNAL;

/* PHY loop back.  INTERNAL loopback configures the PHY to  */
/* loopback data internally.  EXTERNAL loopback, configures */
/* the PHY to normal mode.  Data can be looped back at the  */
/* fiber transceiver with a cable when configured as        */
/* EXTERNAL.                                                */

Phy_loop_back = EXTERNAL;

```

Inverse assembler generated SW_ADDR (software address) label

In the 16700-series logic analysis system, the MPC7410/4X/5X inverse assembler generates a “SW_ADDR” label. The SW_ADDR label is displayed as another column in the Listing tool. This label is also known as the software address generated by the inverse assembler.

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

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

For example, if the compiler unrolls a loop in the source code, the trace could begin after the first assembly instruction of the loop has been executed. A Goto In Listing command would not find the source line.

Access to Source Code Files

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

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

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

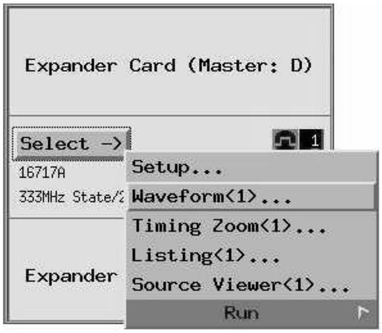
Source File Version Control. If the source code files are under a source code or version control utility, check the file names and paths carefully. These utilities can change source code file paths and file names. If these names are changed from the information contained in the symbol file, the source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

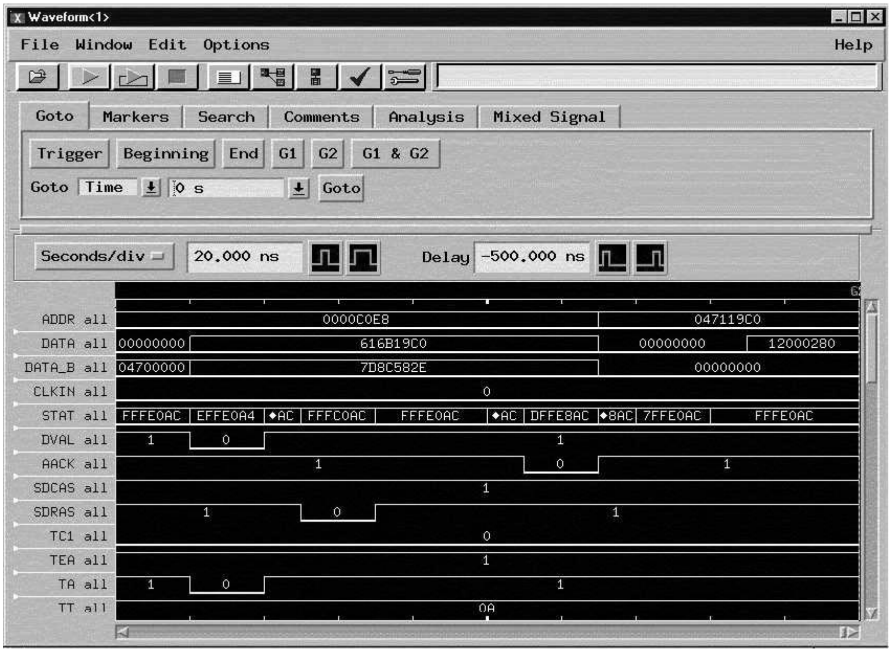
See Also

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

To Display Captured Timing Analysis Mode Data

- Open the Waveform display for your logic analyzer.





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

8

Coordinating Logic Analysis with Processor Execution

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

What are some of the tools I can use?

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

- Your logic analyzer, to acquire data from the processor bus while it is running full-speed.
- Your emulation module, to control the execution of your target processor and to examine the state of the processor and of the target system.
- The Emulation Control Interface, to control and configure the emulation module, and to display or change target registers and memory.
- Display tools including the Listing tool, Chart tool, and System Performance Analyzer tool, to provide different views of the data collected using the logic analyzer.
- Your debugger, to control your target system using the emulation module.

  Do not use the debugger at the same time as the Emulation Control Interface.
- The B4620B source correlation tool set, to relate the analysis trace to your high-level source code.

Which assembly-level listing should I use?

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

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

Which source-level listing should I use?

Different tools display source code for different uses:

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

Where can I find practical examples of measurements?

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

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

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

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

Stopping Processor Execution on a Logic Analyzer Trigger

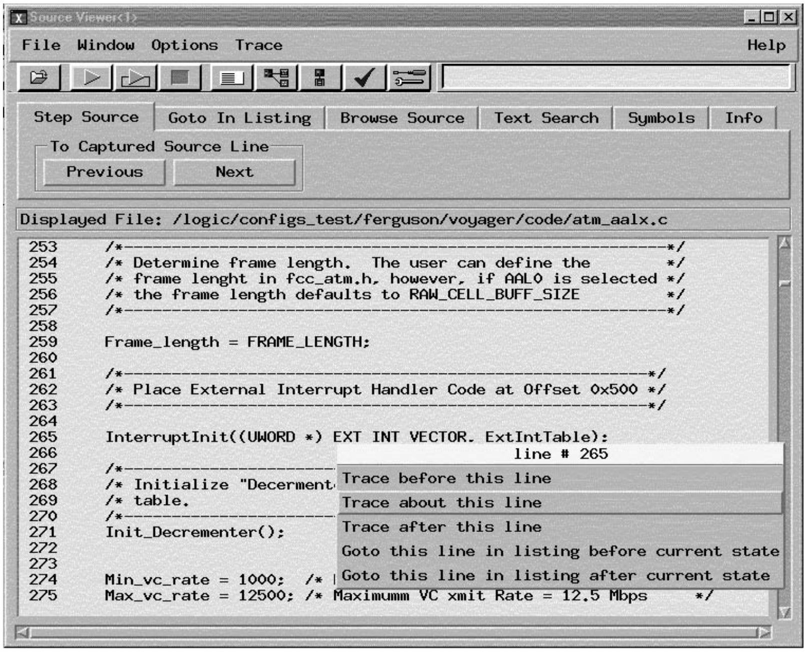
You can trigger the emulation module from the logic analyzer using either the Source Viewer window or the Intermodule window. If you are using the B4620B source correlation tool set, using the Source Viewer window is the easiest method.

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

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

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

The logic analyzer trigger is now set.



Displayed File: /logic/configs_test/ferguson/voyager/code/atm_aalx.c

```
/*--------------------------------------------------------------*/
/* Determine frame length.  The user can define the             */
/* frame lenght in fcc_atm.h, however, if AAL0 is selected */
/* the frame length defaults to RAW_CELL_BUFF_SIZE              */
/*--------------------------------------------------------------*/

Frame_length = FRAME_LENGTH;

/*------------------------------------------------------------*/
/* Place External Interrupt Handler Code at Offset 0x500 */
/*------------------------------------------------------------*/

InterruptInit((UWORD *) EXT INT VECTOR, ExtIntTable);

/*-----------------------
/* Initialize "Decerment
/* table.
/*-----------------------
Init_Decrementer();


Min_vc_rate = 1000;  /* |
Max_vc_rate = 12500; /* Maximumm VC xmit Rate = 12.5 Mbps     */
```

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

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

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

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

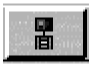
4 If your target system is not already running, select Group Run in the emulation Run Control window to start your target.

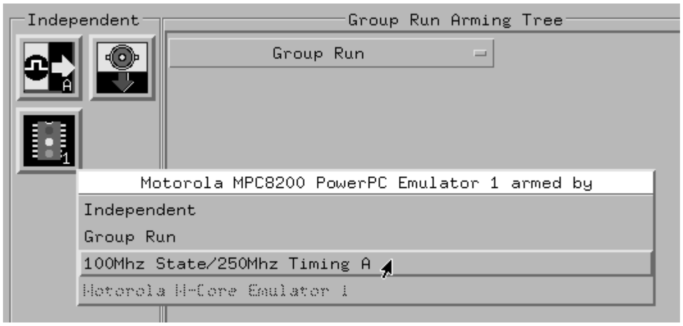
To stop on any trigger (Intermodule window)

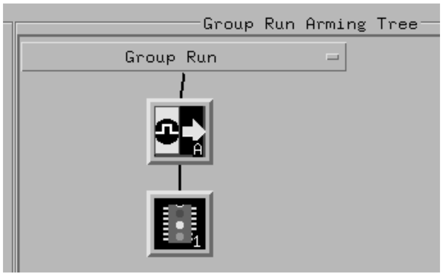
Use the Intermodule window if you do not have the B4620B source correlation tool set or if you need to use a more sophisticated trigger than is possible in the Source Viewer window.

1 Create a logic analyzer trigger.

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




Independent
Group Run Arming Tree
Group Run
A
1
Motorola MPC8200 PowerPC Emulator 1 armed by
Independent
Group Run
100Mhz State/250Mhz Timing A
Motorola M-Core Emulator 1



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

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

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

See Also

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

To minimize the “skid” effect

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

To minimize the skid effect:

1 In the Emulation Control Interface, open the Configuration window.

2 Set processor clock speed to the maximum value which your target can support.

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

To stop the analyzer and view a measurement

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

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

Example

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

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

   If the trigger position is not “End”, the analyzer may continue waiting for more states.
6. The user selects Stop in a logic analyzer window, which tells the logic analyzer to stop waiting, and to display the trace.

Tracing Until the Processor Halts

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

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

Some possible uses for this measurement are:

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

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

To capture a trace before the processor halts

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

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

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

Triggering the Logic Analyzer when Processor Execution Stops

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

If you are only using a state analyzer to capture the processor bus, it will be much simpler to trace until a processor halts (see “Tracing Until the Processor Halts” on page 159).

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

The Emulation Module Trigger Signal

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

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

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

Group Run

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

The following setups can operate independently of Group Run:

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

Here are some examples:

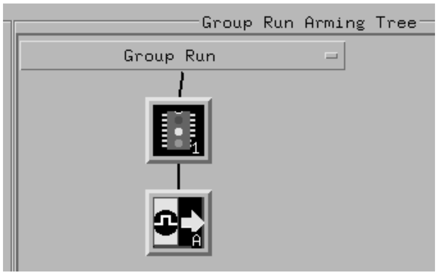
- If “Group Run” is armed from “Port In” and an emulation module is connected to Group Run, any “Port In” signal will cause the emulation module to go into monitor. The Group Run button does not have to be selected for this to operate.
- If two emulation modules are connected together so that one triggers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

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

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

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

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



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

Debuggers can cause triggers

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

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

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

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

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

To trigger the analyzer when the processor halts

Remember: if you are only using a state analyzer to capture the processor bus then it will be much simpler to trace until a processor halts (see "Tracing Until the Processor Halts" on page 159).

1 Set the logic analyzer to trigger on anystate.

2 Set the trigger point to center or end.

3 In the Intermodule window, select the logic analyzer you want to trigger and select the emulation module.

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

4 Select Group Run to start the analyzer(s).

5 Select Run in the Emulation Control Interface or use your debugger to start the target processor running.

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

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

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

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

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

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

To trigger the analyzer when the processor reaches a breakpoint

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

Remember: if you are only using a state analyzer to capture the processor bus, it will be much simpler to trace until a processor halts (see "Tracing Until the Processor Halts" on page 159).

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

   The logic analyzer is now set to trigger on a processor halt.
4. Set the breakpoint.

   If you are going to run the emulation module from Reset you must do a Reset followed by Break to properly set the breakpoints. The Reset will clear all on-chip hardware breakpoint registers. The Break command will then reinitialize the breakpoint registers. If you are using software breakpoints which insert an illegal instruction into your program at the breakpoint location you will not need to do the Reset, Break sequence. Instead, you must take care to properly insert your software breakpoint in your RAM program location.
5. Select Group Run to start the analyzer(s).
6. Select Run in the Emulation Control Interface or use your debugger to start the target processor running.

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

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

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

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

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

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

9

General-Purpose ASCII (GPA) Symbol File Format

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

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

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

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

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

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

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

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

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

Example

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

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

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

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

GPA Record Format Summary

Format

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

 [FUNCTIONS]
 func_name  start..end

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

 [SOURCE LINES]
 File: file_name
 line#  address

 [START ADDRESS]
 address

 #Comments
```

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

Example

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

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

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

 [VARIABLES]
 total    40002000  4
 value    40008000  4

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

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

SECTIONS

Format

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

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

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

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

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

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

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

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

Enable Section Relocation

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

Example

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

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

FUNCTIONS

Format

```
[FUNCTIONS]
 func_name  start..end
```

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

`func_name` A symbol representing the function name.

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

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

Example

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

VARIABLES

Format

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

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

`var_name` A symbol representing the variable name.

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

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

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

Example

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

SOURCE LINES

Format

```
[SOURCE LINES]
 File: file_name
 line#  address
```

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

`file_name` The name of a file.

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

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

Example

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

START ADDRESS

Format

```
[START ADDRESS]
 address
```

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

Example

```
[START ADDRESS]
 00001000
```

Comments

Format

```
#comment text
```

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

Example

```
 #This is a comment.
```

10

Troubleshooting the Inverse Assembler

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

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

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

Logic Analyzer Problems

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

Intermittent data errors

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

❑ Remove and re-seat all cables and probes, ensuring that there are no bent pins or poor probe connections.

❑ Adjust the threshold level of the data pod to match the logic levels in the system under test.

❑ Use an oscilloscope to check the signal integrity of the data lines.

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

See Also

See "Capacitive loading" on page 182 for information on other sources of intermittent data errors.

Unwanted triggers

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

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

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

No activity on activity indicators

❑ Check for loose cables or board connections.

❑ Check for bent or damaged pins on the connectors.

No trace list display

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

❑ Check your trigger sequencer specification to ensure that it will capture the events of interest.

❑ Try stopping the analyzer; if the trace list is partially filled, this should display the contents of trace memory.

Analyzer won't power up

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

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

Target System Problems

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

Target system will not boot up

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

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

a Power up the analyzer.

b Power up the target system.

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

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

Erratic trace measurements

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

Some designs require a full reset to ensure correct configuration.

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

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

❏ Ensure that you have sufficient cooling for the microprocessor.

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

Capacitive loading

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

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

❏ Remove as many pin protectors, extenders, and adapters as possible.

Inverse Assembler Problems

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

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

No inverse assembly or incorrect inverse assembly

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

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

Refer to page 123 to select the correct processor type.

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

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

❏ Check the activity indicators for status lines locked in a high or low state.

❏ Verify that the STAT, STAT_B, DATA, DATA_B, ADDR, and ADDR_B 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. See“Configuring the Logic Analyzer” on page 85 for more information.

❑ Verify that memory managers have been disabled.

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

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

For instructions on how to disable the cache, see page 146.

If the cache is enabled, ensure that an S-Record is loaded and that the preferences are set correctly (see page 123.)

❑ Verify that storage qualification has not excluded storage of all the needed opcodes and operands.

❑ Verify that the endian selection is correct in the preferences menu (see page 123).

Inverse assembler will not load or run

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

In addition, the E8135A (MPXbus inverse assembler) will not load without the MPXbus Tool. If you get an error regarding the STAT_R and ADDR_R labels, it is because you have not loaded the MPXbus Tool. See “Using the MPXbus Tool” on page 98 for more information.

❑ 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.

❑ If using the E8135A (MPXbus) inverse assembler, ensure that you have the MPXbus Tool on the workspace between the data source icon and the Listing icon. The configuration files should load the MPXbus Tool automatically. See “Using the MPXbus Tool” on page 98 for more information.

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

See Chapter 3, “Setting Up the Logic Analysis System,” beginning on page 69 for details.

Intermodule Measurement Problems

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

An event wasn't captured by one of the modules

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

❑ Adjust the skew in the Intermodule menu.

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

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

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

Logic Analyzer Messages

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

“. . . Inverse Assembler Not Found”

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

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

See Chapter 3, “Setting Up the Logic Analysis System,” beginning on page 69 for details.

“No Configuration File Loaded”

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

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

See Also

See “To load configuration files (and the inverse assembler) from the system hard disk” on page 88 for details on loading configuration files.

“Selected File is Incompatible”

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

“Slow or Missing Clock”

❑ This error message might occur if the logic analyzer cards are not firmly seated in the logic analysis system frame. Ensure that the cards are firmly seated.

❑ This error might occur if the target system is not running properly. Ensure that the target system is on and operating properly.

❑ If the error message persists, check that the logic analyzer pods are connected to the proper connectors on the target system or analysis probe interface. See Chapter 3 to determine the proper connections.

“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.

11

Hardware Reference

This chapter contains additional reference information including the specifications and characteristics for the target system when using the inverse assembler software and signal mapping for E8170B and E8135A software.

Operating characteristics - MPC7410/4X/5X

The following operating characteristics are not specifications, but are typical operating characteristics for the E8170B and E8135A Inverse Assemblers.

Operating Characteristics - MPC7410/4X/5X	
Microprocessor Compatibility	MPC7410/40/41/45/50/51/55
Maximum Supported Bus Speed	200 MHz for the 16740/41/42A Logic Analyzers. 400 MHz for the 16750/51/52A Logic Analyzers. 600 MHz for the 16753/54/55/56A Logic Analyzers. 167 MHz for the 16715/16/17/18/19A Logic Analyzers. 140 MHz for the 16557D Logic Analyzers.
Logic Analyzers Supported	16715/16/17/18/19/40/41/42/50/51/52/53/54/55/56A, 16557D (One, two, three, four, or five cards.)
Probes Required	The E8170B requires eight logic analyzer pods (136 channels) for traditional inverse assembly (with 64-bit data). The E8135A requires eight logic analyzer pods (136 channels) for traditional inverse assembly of a single MPC7410/40/41/45/50/51/55 processor (with 64-bit data). The E8135A requires 10 logic analyzer pods (180 channels) for inverse assembly of multiple MPC7410/40/41/45/50/51/55 processors.
Signal Line Loading	Typically 100 kΩ plus 10 pF.
Setup/Hold Requirement	For all signals, the logic analyzers require a minimum combined setup/hold window. For 16715/16/17/18/19/40/41/42/50/51/52A logic analyzers the combined window must be at least 2.5 ns (1.25 ns using eye finder). For 16557 logic analyzers, the combined setup/hold time must be at least 3.0 ns.

Glossary

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

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

Debug Mode See *Background Debug Monitor*.

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

Elastomeric Probe Adapter A connector that is fastened on top of a target microprocessor using a retainer and knurled nut. The conductive elastomer on the bottom of the probe adapter makes contact with pins of the target microprocessor and delivers their signals to connection points on top of the probe adapter.

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

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

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

Emulator An emulation module or an emulation probe.

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

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 Historical term for a set of tools for debugging your target system. A solution included probing, inverse assembly, the B4620B Source

Correlation Tool 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.

Index

Safety Notices

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."

Warnings

- 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.
- If you energize this instrument by an auto transformer (for voltage reduction or mains isolation), the common terminal must be 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.
- 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.
- 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.
- 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 use the instrument in a manner not specified by the manufacturer.

To clean the instrument

If the instrument requires cleaning: (1) Remove power from the instrument. (2) Clean the external surfaces of the instrument with a soft cloth dampened with a mixture of mild detergent and water. (3) Make sure that the instrument is completely dry before reconnecting it to a power source.

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.

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

Notices

Manual Part Number

E8135-97005, October 2002

Print History

E8135-97004, September 2001
E8135-97003, July 2001
E8135-97002, June 2001
E8135-97001, February 2001
E8135-97000, October 2000
E8170-97000, December 1999

Agilent Technologies, Inc.
1900 Garden of the Gods Road
Colorado Springs, CO 80907 USA

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as “Commercial computer software” as defined in DFAR 252.227-7014 (June 1995), or as a “commercial item” as defined in FAR 2.101(a) or as “Restricted computer software” as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies’ standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Document Warranty

The material contained in this document is provided “as is,” and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

CAUTION

A CAUTION notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a CAUTION notice until the indicated conditions are fully understood and met.