User's Guide

Publication number E8171-97000
February 2001

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

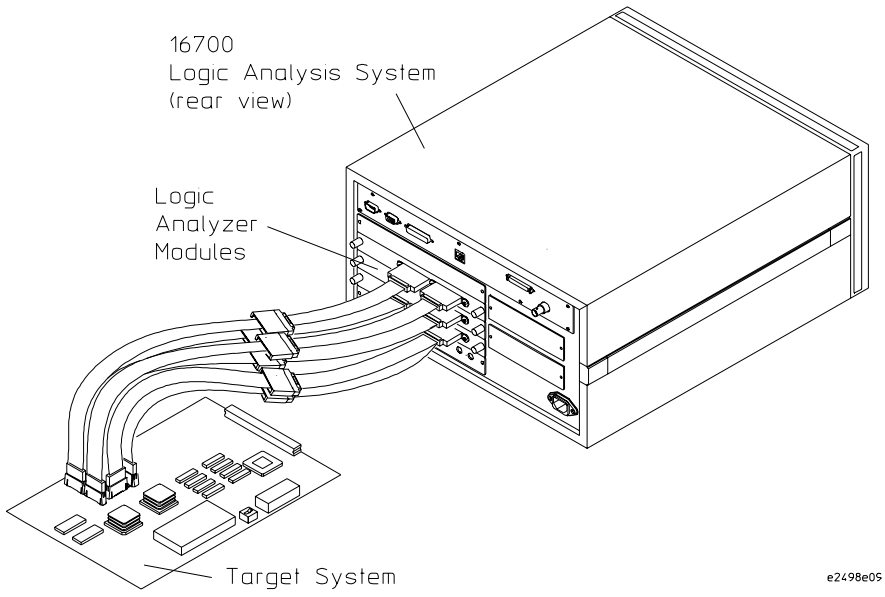
Logic Analysis Support for the IBM PowerPC 405

Logic Analysis Support for the IBM PowerPC 405GP/CR Peripherals Bus—At a Glance

This book documents the Agilent inverse assembler for the IBM PowerPC 405GP and 405CR peripherals bus.

Inverse Assembler Software

The Agilent Technologies E8171A inverse assembler, in conjunction with an Agilent Technologies logic analyzer, allows you to view assembly instructions that are executing in your target system.



The inverse assembler package model number is Agilent Technologies E9618A Option 001. The inverse assembler is identified as E8171A in the Setup Assistant.

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 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.

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 Agilent 16700-series logic analysis system.

In This Book

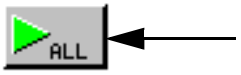
This book documents the following products:

Inverse Assembler Software		
Processors supported	Product ordered	Includes
PowerPC 405GP/CR	E9618A Option 001	E8171A inverse assembler

Use the Setup Assistant

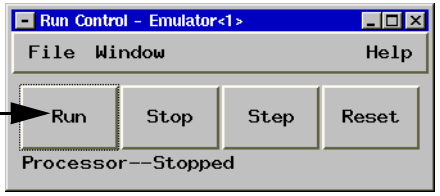
Click here to connect the logic analyzer cables and automatically load the correct configuration files. See page 17.

Use the appropriate Run button



Click here to start a measurement.

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



Contents

Contents

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.

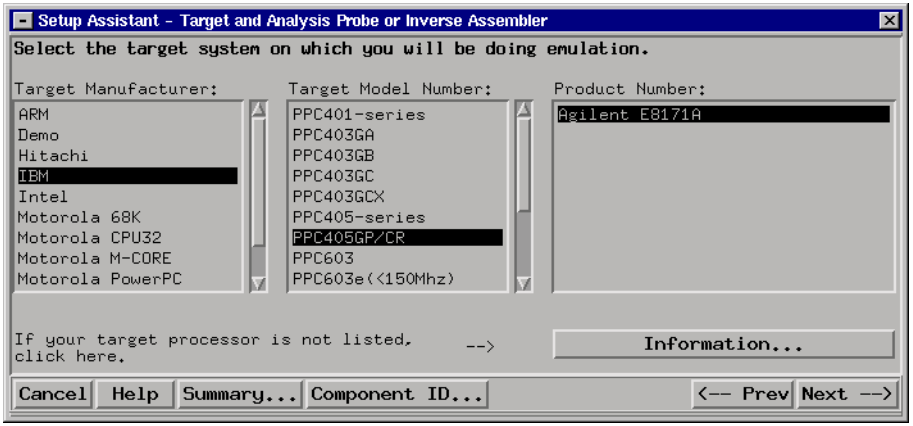
- Install the software. See page 42.
- Use the Setup Assistant to help you connect and configure your logic analysis system. See page 17.

Setup Assistant

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

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

Start the Setup Assistant by selecting  in the system window.



NOTE: 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 42.

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 E8171A inverse assembler software package consists of the following:

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

Minimum equipment required

For state and timing analysis of a PPC405 target system, you need all of the following items.

- The E8171A PPC405 Inverse Assembler.
- The appropriate connectors on the target system. Chapter 2, “Preparing the Target System,” contains information on designing the appropriate connectors into the target system.
- One of the logic analyzers listed on page 20.

Additional equipment supported

B4620B Source Correlation Tool Set

The inverse assembler software may be used with the B4620B Source Correlation Tool Set on an 16700-series logic analysis system to analyze high-level source code.

Emulation Probe

The inverse assembler may be used with an Agilent or IBM emulation probe.

The combination of an inverse assembler, and emulation module, and an Agilent Technologies 16700-series logic analysis system lets you both view 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, for the Agilent emulation probe, the logic analysis system's Emulation Control Interface to configure and control the target processor and to download program code.

Compatible Logic Analyzers

The table below lists the logic analyzers supported by the E8171A inverse assembler software.

The inverse assembler requires six logic analyzer pods. See "Connecting the Logic Analyzer to the Target System" on page 45 for details.

Logic Analyzers Supported

Logic Analyzer	Channel Count	State Speed *	Timing Speed *	Memory Depth
16752A (2 or more cards)	68/card	400 MHz	2 GHz	32 M states
16751A (2 or more cards)	68/card	400 MHz	2 GHz	16 M states
16750A (2 or more cards)	68/card	400 MHz	2 GHz	4 M states
16719A (2 or more cards)	68/card	333 MHz	2 GHz	32 M states
16718A (2 or more cards)	68/card	333 MHz	2 GHz	8 M states
16717A (2 or more cards)	68/card	333 MHz	2 GHz	2 M states
16716A (2 or more cards)	68/card	167 MHz	2 GHz	512 k states
16715A (2 or more cards)	68/card	167 MHz	667 MHz	2 M states
16712A (1 or more cards)	102/card	100 MHz	500 MHz	128 k states
16711A (1 or more cards)	102/card	100 MHz	500 MHz	32 k states
16710A (1 or more cards)	102/card	100 MHz	500 MHz	8 k states
16557D (2 or more cards)	68/card	140 MHz	500 MHz	2 M states
16556A (2 or more cards)	68/card	100 MHz	400 MHz	1 M states
16555D/56D (2 or more cards)	68/card	110 MHz	500/400 MHz	2 M states

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

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

2

Preparing the Target System

Target System Requirements

Microprocessor modes

Ensure that the processor is not using code compression or MMU mode, and that the instruction and data caches are disabled while the inverse assembler is capturing a trace. See page 100 for information on how to disable the caches.

Connectors

The target system must include three logic probes analyzer connectors.

Using general purpose (GP) probes to connect the logic analyzer to the target system is not recomended due to the large number of signals and signal integrity issues.

The following sections contain information on designing the connectors into your target system connection.

Designing and Using Built-in Connectors

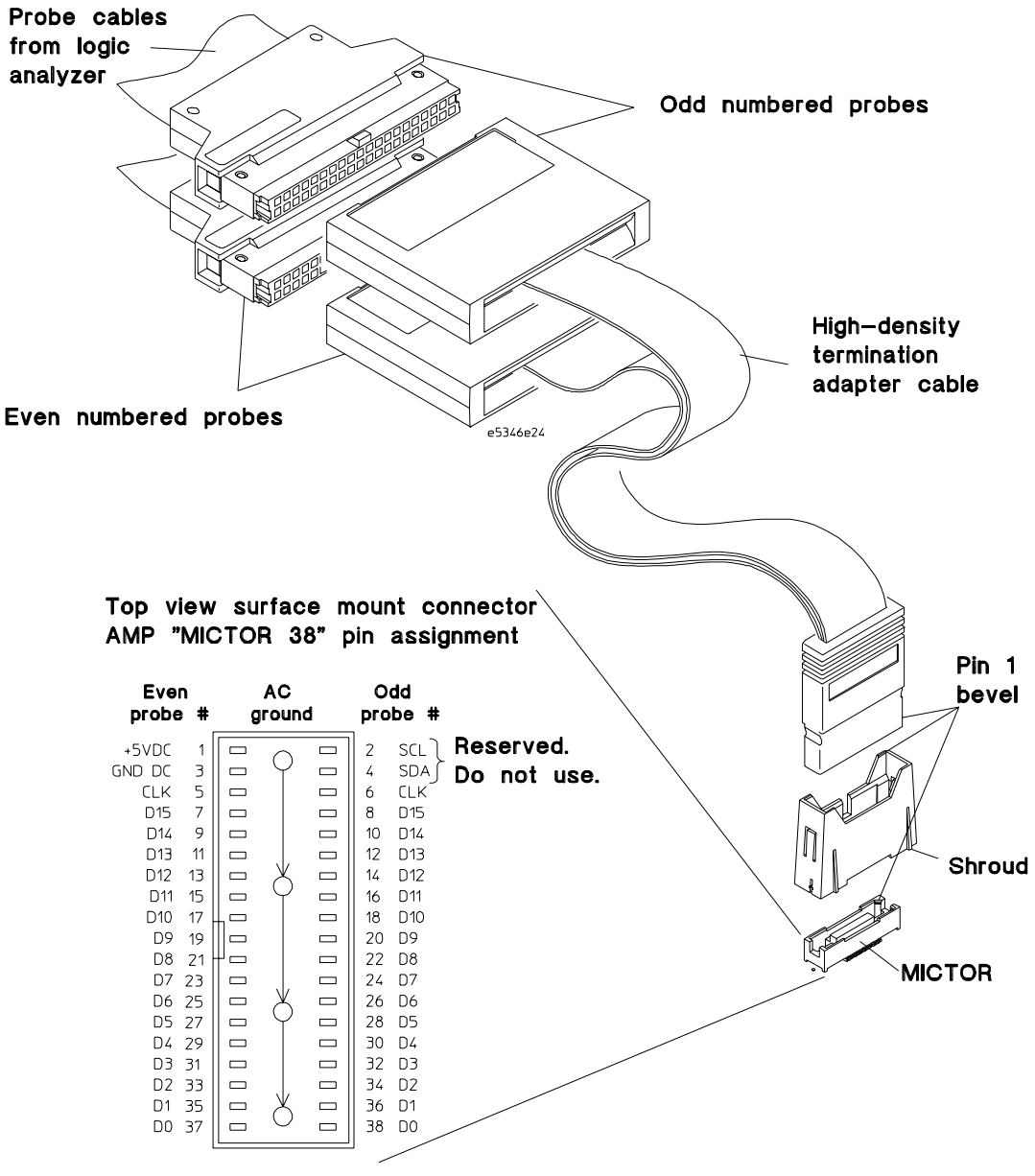
You can design analyzer-compatible connectors into the target board, and connect the logic analyzer cables to these connectors according to the tables beginning on page 30. The primary concerns when using built-in connectors are:

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

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

Part Numbers for Built-in Connectors and Cables	
Part Number	Description
Agilent 1252-7431, or AMP 2-767004-2	AMP MICTOR 2 x 19 header. Three connectors (six logic analyzer 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



AMP MICTOR 38 connectors

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

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

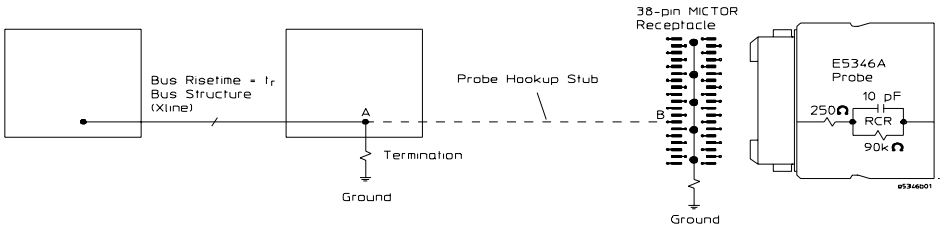
To increase the structural support for the cables, you should use cable support shrouds (part number E5346-44701) on each connector. The figures on the following page show the mechanical layouts for the shrouds and headers.

Design Considerations

The 2x19 header must be close enough to the target signal so that the stub length created is less than $^1/_5$ 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.

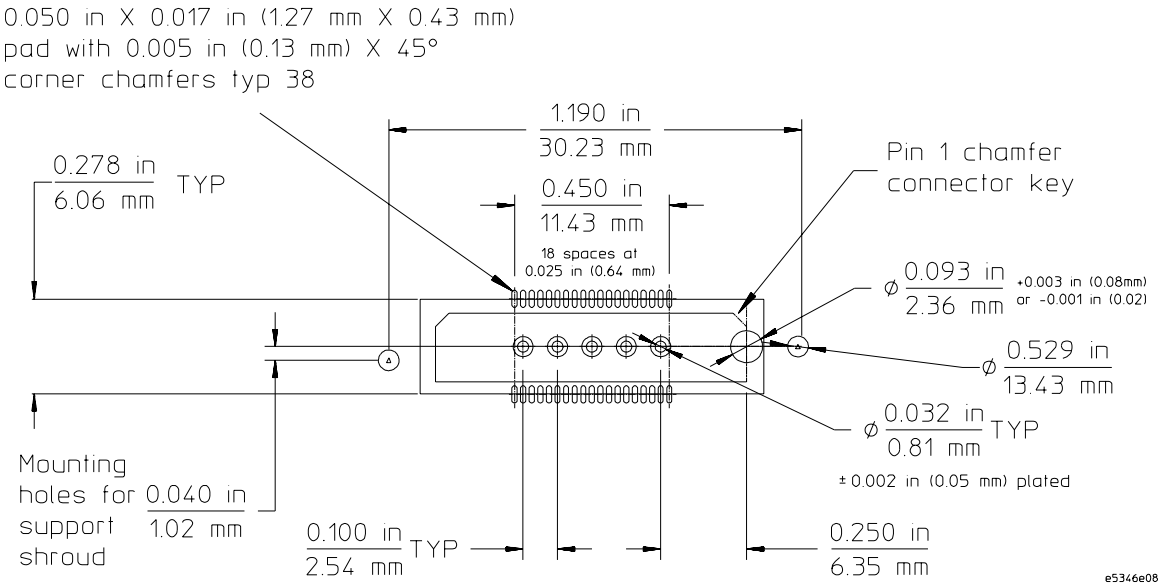
MICTOR 38 Connector Design Rules



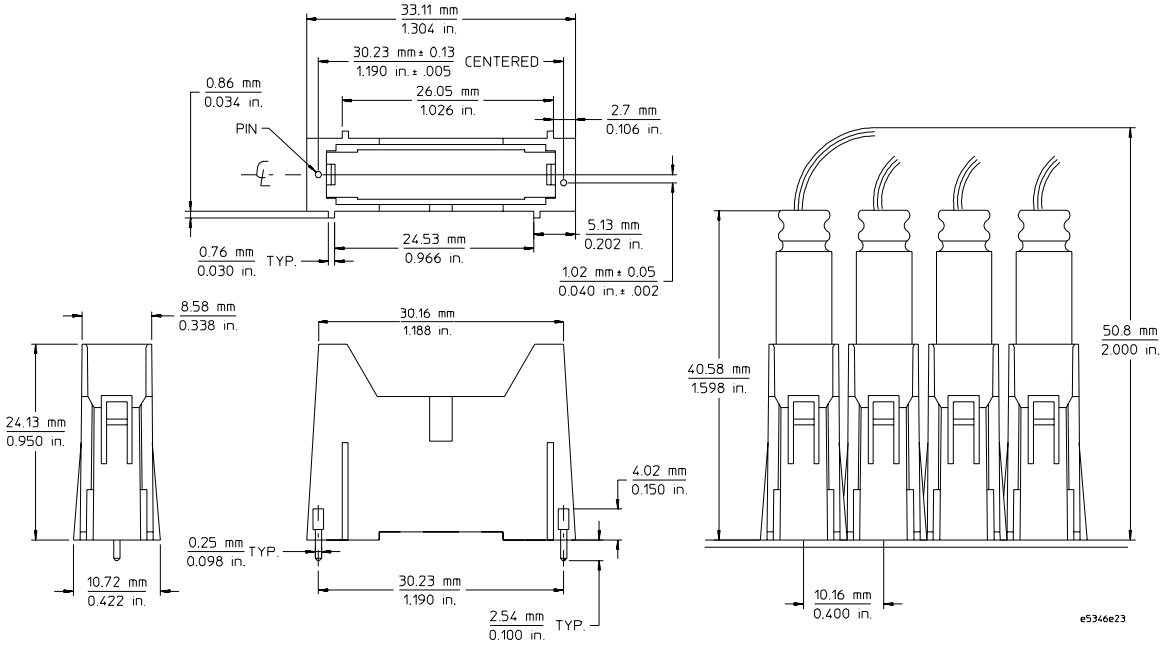
Support shroud

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

Support Shroud Mechanical Information



MICTOR 38 Connector Mechanical Information



Inverse Assembler—Signal-to-Connector Mapping

The following tables show the electrical signal-to-connector mapping required by the E8171A inverse assembler software.

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

MICTOR Connector J1 Odd			
2x19 pin	**LA bit**	**IBM Signal name**	**Analyzer label**
6	CLK	PerClk	CLK
8	15 (MSB)	PerAddr16	ADDR[15]
10	14	PerAddr17	ADDR[14]
12	13	PerAddr18	ADDR[13]
14	12	PerAddr19	ADDR[12]
16	11	PerAddr20	ADDR[11]
18	10	PerAddr21	ADDR[10]
20	9	PerAddr22	ADDR[9]
22	8	PerAddr23	ADDR[8]
24	7	PerAddr24	ADDR[7]
26	6	PerAddr25	ADDR[6]
28	5	PerAddr26	ADDR[5]
30	4	PerAddr27	ADDR[4]
32	3	PerAddr28	ADDR[3]
34	2	PerAddr29	ADDR[2]
36	1	PerAddr30	ADDR[1]
38	0 (LSB)	PerAddr31	ADDR[0]

MICTOR Connector J1 Even			
2x19 pin	**LA bit**	**IBM Signal name**	**Analyzer label**
5	CLK	--	
7	15 (MSB)	PerAddr0	ADDR[31]
9	14	PerAddr1	ADDR[30]
11	13	PerAddr2	ADDR[29]
13	12	PerAddr3	ADDR[28]
15	11	PerAddr4	ADDR[27]
17	10	PerAddr5	ADDR[26]
19	9	PerAddr6	ADDR[25]
21	8	PerAddr7	ADDR[24]
23	7	PerAddr8	ADDR[23]
25	6	PerAddr9	ADDR[22]
27	5	PerAddr10	ADDR[21]
29	4	PerAddr11	ADDR[20]
31	3	PerAddr12	ADDR[19]
33	2	PerAddr13	ADDR[18]
35	1	PerAddr14	ADDR[17]
37	0 (LSB)	PerAddr15	ADDR[16]

MICTOR Connector J2 Odd			
2x19 pin	**LA bit**	**IBM Signal name**	**Analyzer label**
6	CLK	SysErr	SYS_ERR
8	15	~PerWBE0	-WBE0
10	14	~PerWBE1	-WBE1
12	13	~PerWBE2	-WBE2
14	12	~PerWBE3	-WBE3
16	11	~PerOE	-OE
18	10	~PerWE [PCIINT]	-WE PCIINT
20	9	PerR/~W	R/-W
22	8	~PerBLast	-BLAST
24	7	~PerCS0	-CS0
26	6	~PerCS1 [GPIO10]	-CS1
28	5	~PerCS2 [GPIO11]	-CS2
30	4	~PerCS3 [GPIO12]	-CS3
32	3	~PerCS4 [GPIO13]	-CS4
34	2	~PerCS5 [GPIO14]	-CS5
36	1	~PerCS6 [GPIO15]	-CS6
38	0	~PerCS7 [GPIO16]	-CS7

MICTOR Connector J2 Even			
2x19 pin	**LA bit**	**IBM Signal name**	**Analyzer label**
5	CLK	--	
7	15	DMAReq0	DMA_REQ0
9	14	DMAReq1	DMA_REQ1
11	13	DMAReq2	DMA_REQ2
13	12	DMAReq3	DMA_REQ3
15	11	DMAAck0	DMA_ACK0
17	10	DMAAck1	DMA_ACK1
19	9	DMAAck2	DMA_ACK2
21	8	DMAAck3	DMA_ACK3
23	7	EOT0 [TC0]	DMA_EOT0
25	6	EOT1 [TC1]	DMA_EOT1
27	5	EOT2 [TC2]	DMA_EOT2
29	4	EOT3 [TC3]	DMA_EOT3
31	3	PerREADY	READY
33	2	~ExtReq	-EXT_REQ
35	1	~ExtAck	-EXT_ACK
37	0	PerErr	ERROR

MICTOR Connector J3 Odd			
2x19 pin	**LA bit**	**IBM Signal name**	**Analyzer label**
6	CLK	--	
8	15 (MSB)	PerData16	DATA[15]
10	14	PerData17	DATA[14]
12	13	PerData18	DATA[13]
14	12	PerData19	DATA[12]
16	11	PerData20	DATA[11]
18	10	PerData21	DATA[10]
20	9	PerData22	DATA[9]
22	8	PerData23	DATA[8]
24	7	PerData24	DATA[7]
26	6	PerData25	DATA[6]
28	5	PerData26	DATA[5]
30	4	PerData27	DATA[4]
32	3	PerData28	DATA[3]
34	2	PerData29	DATA[2]
36	1	PerData30	DATA[1]
38	0 (LSB)	PerData31	DATA[0]

MICTOR Connector J3 Even			
2x19 pin	**LA bit**	**IBM Signal name**	**Analyzer label**
5	CLK	--	
7	15 (MSB)	PerData0	DATA[31]
9	14	PerData1	DATA[30]
11	13	PerData2	DATA[29]
13	12	PerData3	DATA[28]
15	11	PerData4	DATA[27]
17	10	PerData5	DATA[26]
19	9	PerData6	DATA[25]
21	8	PerData7	DATA[24]
23	7	PerData8	DATA[23]
25	6	PerData9	DATA[22]
27	5	PerData10	DATA[21]
29	4	PerData11	DATA[20]
31	3	PerData12	DATA[19]
33	2	PerData13	DATA[18]
35	1	PerData14	DATA[17]
37	0 (LSB)	PerData15	DATA[16]

Designing a JTAG Connector into Your Target System

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

3

Setting Up the Logic Analysis System

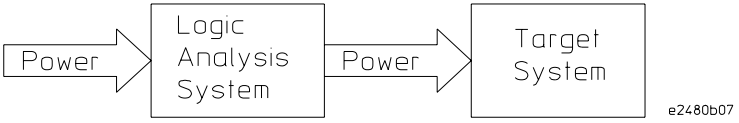
Power-on/Power-off Sequence

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.

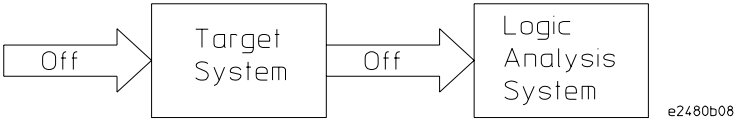
1. Turn on the logic analyzer. The Setup Assistant will guide you through the process of connecting and configuring the logic analyzer.
2. When the target system is connected to the logic analyzer, and everything is configured, turn on your target system.



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

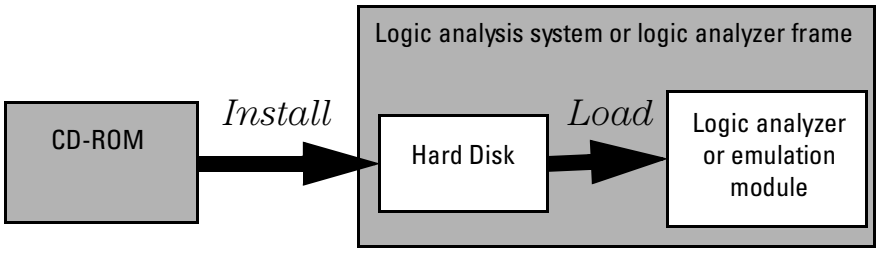
You should install logic analyzer, oscilloscope, or pattern generator modules in your logic analysis system before you install software.

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 system's *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.



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 assembler (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 on the **"POWERPC4XX"** package.

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

9. Select **Install**.

   The Continue dialog box will appear.

10. Select **Continue**.

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

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

The configuration files are stored in /logic/configs/hp/ppc4xx/ppc405. The inverse assemblers are stored in /logic/ia.

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

See Also 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).

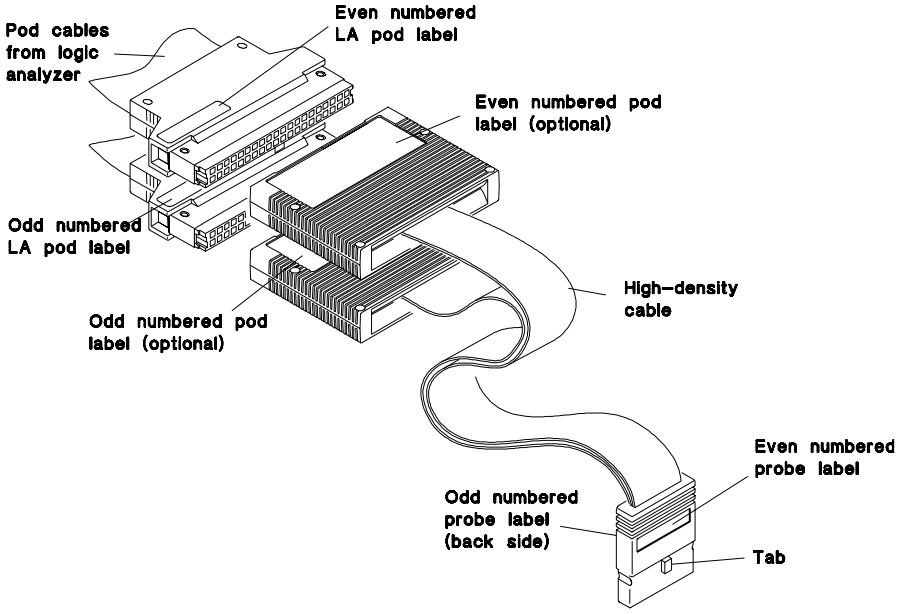
If you are not using the Setup Assistant, follow the instructions given in this chapter. This chapter covers the following tasks in the recommended order:

- Check that the target system meets the necessary requirements (see page 24)
- Read the power on/power off sequence (see page 40)
- Connect the target system to the logic analyzer (see page 47)
- Configure the logic analyzer (see page 51)

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

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

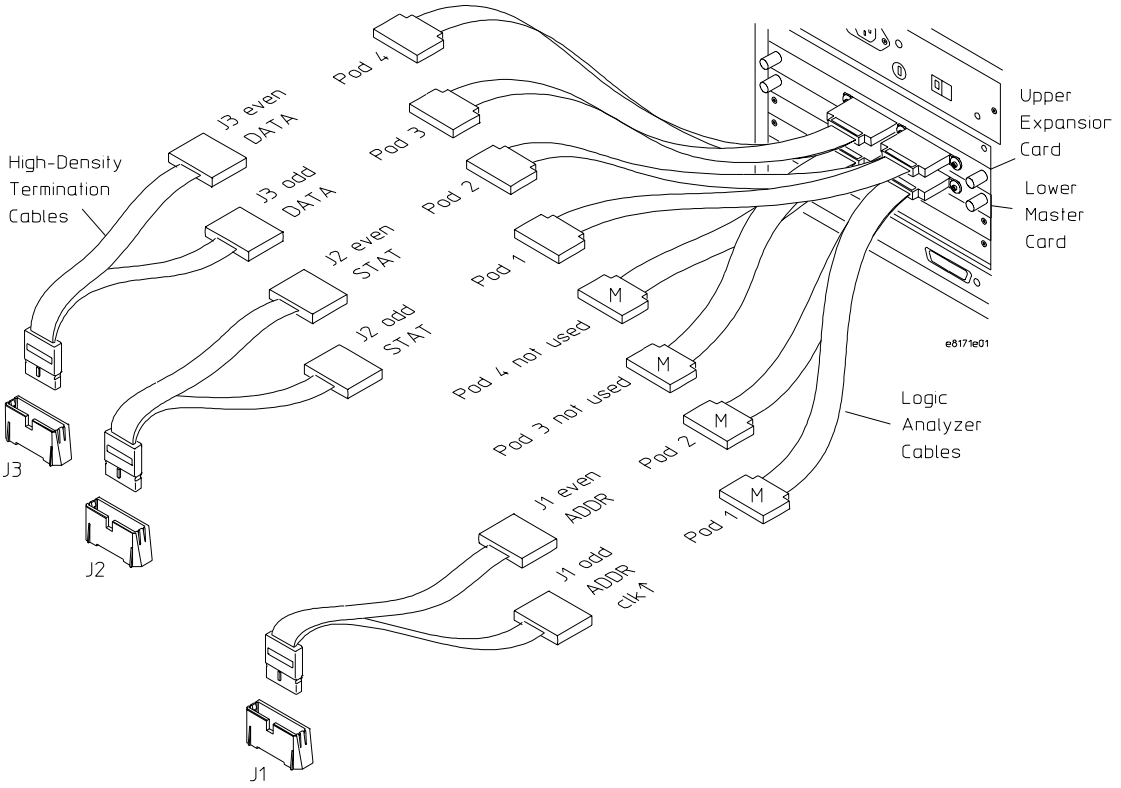
E5346A Cable Numbering



e2498e08

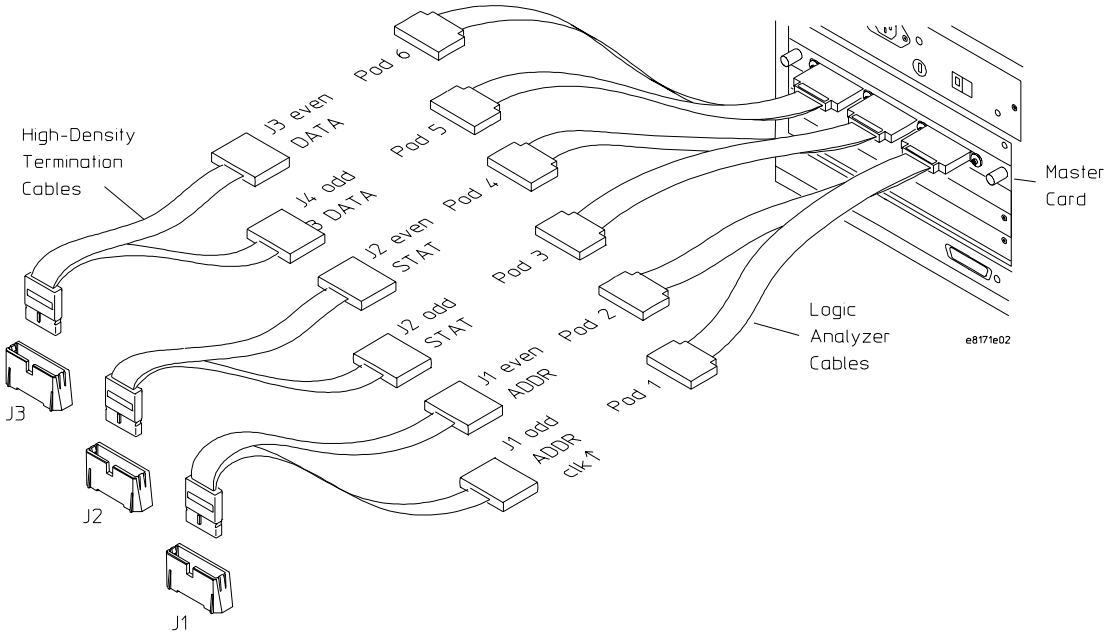
To connect to four-pod-per-card logic analyzers (two cards)

Use the figure below to connect the target system to the 16554/55/56/57 or 16715/16/17/18/19/50/51/52A logic analyzer. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



To connect to a six-pod-per-card logic analyzer (one card)

Use the figure below to connect the target system to a 16550A or 16710/11/12A 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

The sections of this chapter describe setting up and using the inverse assembler. Because your target system is designed uniquely according to your needs, it is important that you specify the available signals and memory regions to the inverse assembler.

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

- Loading the configuration file and the inverse assembler
- Configuration file names
- Using the inverse assembler
- Setting the inverse assembler preferences
- Symbols
- Changing analysis mode

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
- Inverse assembler file name
- Inverse assembler preference settings

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

It is strongly recommended that you do not change the setup related to the 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.

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/ppc4xx/ppc405 exists.

> If the above directory does not exist, you need to install the POWERPC4XX Processor Support Package. Close File Manager, then use the procedure on the CD-ROM jacket to install the POWERPC4XX Processor Support Package before you continue.

2 Using File Manager, select the configuration file that you want to load from the /logic/configs/hp/ppc4xx/ppc405 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 PPC405 analysis by loading the appropriate PPC405 configuration file. Loading the indicated file also automatically loads the correct inverse assembler.

3 Close File Manager.

To list software packages that are installed

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

Logic Analyzer Configuration Files

The following table lists the configuration files for the PPC405 for each supported logic analyzer card configuration.

Logic Analyzer Configuration Files

Analyzer Model	Configuration File
16750/51/52A	C405L
16715/16/17/18/19A	C405L
16710/11/12A	C405F
16554/5/6/7	C405M
16550A	C405F

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

Using the Inverse Assembler

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

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

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 memory controller configuration. The other dialogs assist in analyzing and displaying data. The following sections describe these dialogs.

Loading the Inverse Assembler

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

Setting Inverse Assembler Preferences

The Invasm Preferences dialog lets you give the inverse assembler information about your target system so that it can properly disassemble signal values captured by the logic analyzer.

To set the inverse assembler preferences

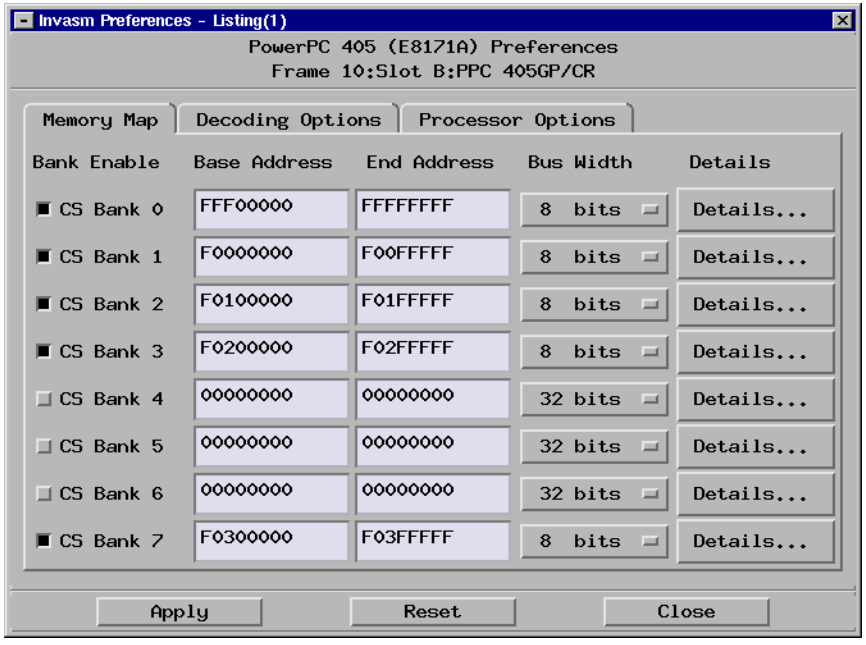
To open the Preferences dialog:

1. Load a configuration file, if needed.
2. Open the Listing window.
3. Select Preferences... from the Invasm menu at the top of the Listing window.
4. Look at each of the tabs in the Preferences dialog and adjust the settings to match your target system. At a minimum, you must set the memory map preferences.

To set the Memory Map preferences

The PowerPC 405 Peripherals Bus does not provide all of the signals required to determine the valid address and data point in a burst cycle, or whether a read cycle is an instruction fetch or a load data operation. It is therefore necessary to use information from the inverse assembler Preferences dialog to make these determinations.

It is necessary to configure the memory map in the Preferences dialog, including "Details", before using the inverse assembler. None of the chip select memory banks are enabled by default.



Bank Enable. Enable the banks for all of the chip selects, and only those chip selects, being used in the system; otherwise, the inverse assembler may not function correctly. These enables are required because with the PPC405 most chip selects can be configured to be general purpose I/O (GPIO) signals.

To find out which chip selects to enable:

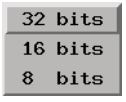
1 Look at the Device Control Register (DCR) CPC0_CR0 to determine if a chip select is being used as a GPIO; if so, disable that chip select in the Preferences dialog to keep the inverse assembler from considering the GPIO signal as a chip select.

2 Check the BU (Bank Usage) field of Device Control Register EBC0_BnCR, where n = chip select number, to determine if the chip select is "Disabled"; if so, disable that chip select in the Preferences. If BU indicates "Read-only", "Write-only" or "Read/Write" enable that chip select in the Preferences.

Base Address, End Address. Specifies the range of addresses used by the memory bank.

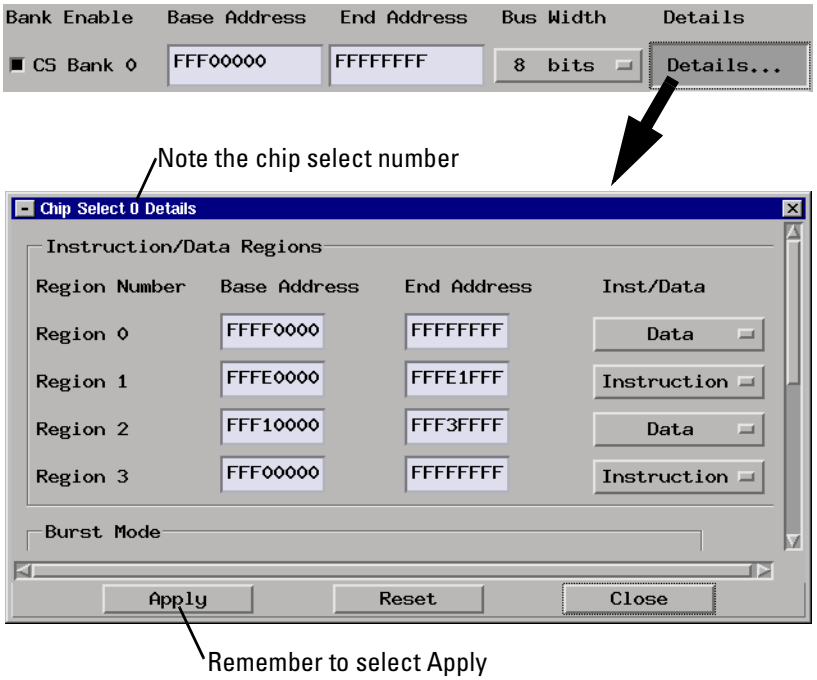
To help determine the value for Base Address, check the BAS (Base Address Select) field of DCR EBC0_BnCR. To help determine the value for the End Address, check the BS (Bank Size) field of DCR EBC0_BnCR. If a smaller range of addresses is actually used (for example, some address lines are ignored by the target) the Base Address and/or the End Address can be adjusted to reflect that.

Bus Width.



If your target system uses 8-bit or 16-bit memory, change this from the default of 32 bits. Check the BW (Bus Width) field of the DCR EBC0_BnCR.

Details (Instruction/Data Regions).



The PPC405 Peripherals Bus does not indicate whether a read cycle is an "instruction fetch" versus a "load data" operation. To supply that information to the inverse assembler:

1 Open the Invasm Preferences Dialog.

2 From the Memory Map tab, select the Details button for one of the chip selects.

3 Enter information about what kind of information is stored in each memory region within the chip select memory bank.

4 Select Apply.

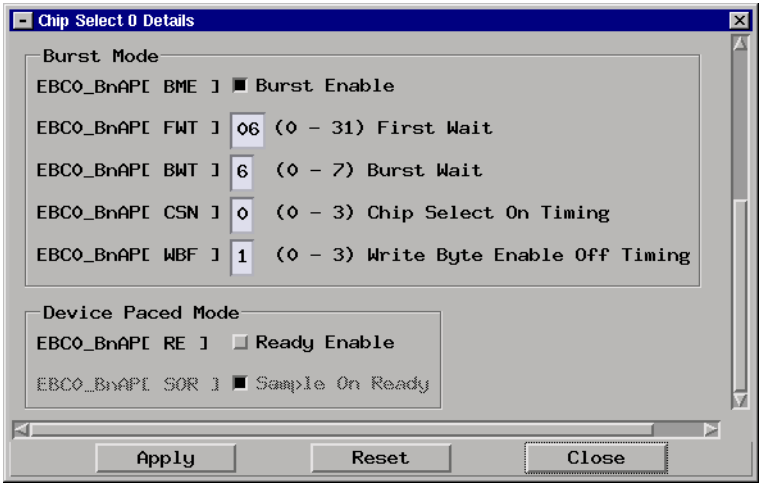
By default, all regions for a given chip select bank are off. The Base Address and End Address for a region determine the range of addresses that will be considered to be Instruction space or Data space as specified by the Inst/Data selection. The Base Address and End Address for any of the four regions should not extend past the Base Address and End Address of the chip select bank itself.

The regions are searched in order from Region 0 through Region 3. Therefore, a small region may be followed by a larger region where the larger region overlaps the small region. If the address being searched for is within the small region, the larger region will not be checked. This allows for flexibility in specifying small data areas within a large program (instruction) space to use fewer regions. Check the memory map for the target program to help in specifying the regions.

A region must be turned on by selecting either "Instruction" or "Data" under Inst/Data to allow the Base Address and End Address fields to be modified. If a region is turned off, the Base Address and End Address fields will be grayed out and the region will be skipped in the address search.

Select Apply in the Chip Select Details dialog when you have finished the details for a given chip select bank. Always select Apply before selecting Close to prevent throwing away any changes made in the Details dialog. Then select Apply in the main Preferences dialog when you are finished configuring the memory map.

Details (Burst Mode and Device Paced Mode).



For the PPC405 inverse assembler to properly handle burst cycles additional information must be provided by the user. The ~PerBlast signal on the PPC405 Peripherals Bus indicates that a burst cycle is in progress, but there is no signal indicating the point in the cycle where the address and data are valid. The number of wait states, the Chip-Select-On-Timing, and the Write-Byte-Enable-Off-Timing are needed to determine this address/data valid

point.

Peripheral devices that use the PerREADY signal to indicate when they are ready for the cycle to be completed require the user to program a PPC405 chip select to operate in Device Paced Mode. The inverse assembler also needs this information to be specified in Preferences.

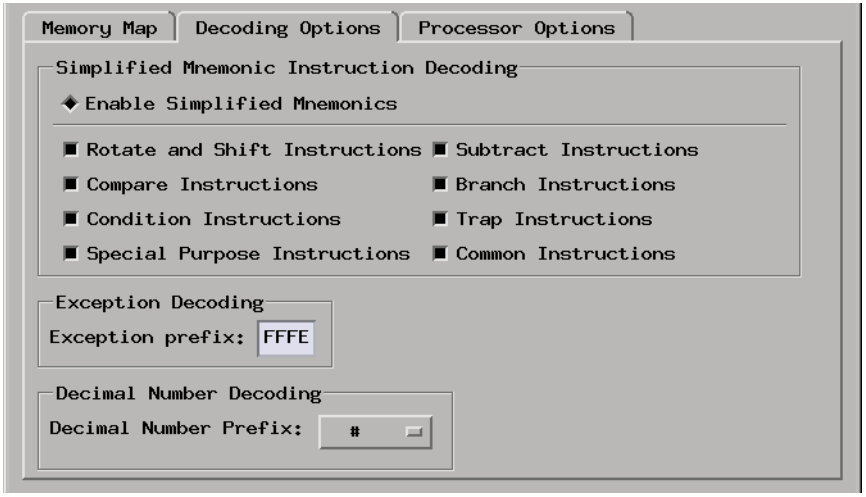
The easiest way to determine the values for Burst Mode and Device Paced Mode is to execute your code until your initialization routines have programmed the Peripheral Bank Configuration Registers (EBC0_B0AP through EBC0_B7AP) in the PPC405. Then examine these registers and enter that information into Preferences. Note that the title bar of the Chip Select Details dialog will indicate which chip select (0 through 7) that you are currently modifying. To the left of each entry is a "Register[Field]" name to help in determining the value to enter in Preferences. For example, if you have brought up the "Chip Select 1 Details" dialog the EBC0_B1AP register should be examined in the PPC405 to determine the values to enter in. In this case the dialog register name EBC0_BnAP[BME] (the n here is a 1) indicates the BME field for chip select 1. If BME is a "1" then push the toggle button for Burst Enable. The remaining entries under Burst Mode will be turned on (no longer grayed-out), and you can now enter the values associated with the FWT, BWT, CSN, and WBF fields. In similar fashion, check fields RE and SOR to determine what should be entered for Device Paced Mode.

TIP: You can leave the Chip Select Details dialog displayed and simply click on the Details... button for a different chip select bank to change to the details for that chip select bank. The chip select number appears in the dialog title to indicate which chip select bank details you are currently editing.

Remember to click Apply in the Chip Select Details dialog before changing to a different chip select bank if any changes were made; otherwise those changes will be thrown away. Note that clicking Apply for the Chip Select Details dialog will not cause the listing display to be updated. This is to allow you to configure all chip select Details without waiting for the listing to be updated each time. Remember to click Apply in the main Preferences dialog when you are finished, which will update the listing display.

Failure to properly configure the Preferences with these details can result in Preferences warning messages in the listing directing you back to this dialog. Also, listings may look correct for a few cycles and incorrect for the rest, especially if the First Wait and Burst Wait values are incorrect.

To set the Decoding Options preferences



Simplified Mnemonic Instruction Decoding. PowerPC assemblers support a number of simplified mnemonics for some popular assembly language instructions, as described in Appendix F of IBM'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 Enable Simplified Mnemonics, 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” and “cror” instructions with identical source registers are decoded as move register, “mr” and “crmove”.
- “nor” and “crnor” instructions with identical source registers are decoded as not register, “not” and “crnot”.
- “crxor” and “creqv” instructions with identical source and destination registers are decoded as “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)	`rotlwi` (rotate left immediate) `rotrwi` (rotate right immediate) `slwi` (shift left immediate) `srwi` (shift right immediate) `extlwi` (extract and left justify immediate) `extrwi` (extract and right justify immediate) `clrlwi` (clear left immediate) `clrrwi` (clear right immediate) `clrlslwi` (clear left and shift left immediate)
`rlwnm` (rotate left word then AND with mask)	`rotlw` (rotate left)

The inverse assembler supports the following categories of instructions. The table shows examples of the extended mnemonics which are supported.

Instruction Category	Example: Raw	Example: Extended
Branch	`bc 4,2,0xFFF00230`	`bne cr0,0xFFF00230`
Trap	`tw 16,r5,r6`	`twlt 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 255,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`
Rotate and Shift	`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 you to determine the location of the exception vector table. You can determine which location is set up for your target by looking at the initialization code, or by examining the Special Purpose Register (SPR) EVPR after the initialization has been run on your target. Enter the value found in the EVP field of the EVPR register into the Exception Prefix field in Preferences.

Following is a list of the exceptions that can be displayed:

Offset	Type
0x00100	Critical input interrupt
0x00200	Machine check interrupt
0x00300	Data storage interrupt
0x00400	Instruction storage interrupt
0x00500	External interrupt
0x00600	Alignment interrupt
0x00700	Program interrupt
0x00C00	System call interrupt
0x01000	PIT interrupt
0x01010	FIT interrupt
0x01020	Watchdog timer interrupt
0x01100	Data TLB miss interrupt
0x01200	Instruction TLB miss interrupt
0x02000	Debug interrupt

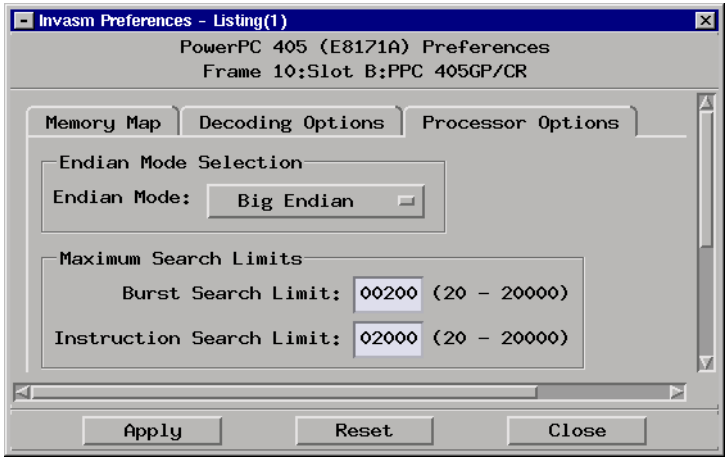
Decimal Number Decoding

The inverse assembler allows you to select the prefix symbol used when decimal numbers are displayed. You can choose '#', 'd', or none at all.

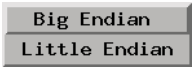
Note that binary numbers (0's and 1's) are always shown with a 'b' prefix, and hexadecimal numbers are always shown with a '0x' prefix.

To set the Processor Options preferences

The Processor Options tab of the Preferences dialog lets you tell the inverse assembler about any processor operating modes, and also allows you to change how far the inverse assembler searches forward or backward when putting together instructions and data load/store operations.



Endian Mode Selection



The inverse assembler is designed to support both the Big-Endian (most-significant byte has the smallest address) and Little-Endian modes (least-significant byte has the smallest address) of operation. When operating in the Little-Endian mode, the processor performs byte-reversal between memory and the Instruction Cache Unit (ICU) for instruction reads. For data load/store operations the processor performs the byte-reversal as needed between the Data Cache Unit (DCU) and the General Purpose Register (GPR). The inverse assembler internally does byte-reversal on instructions to properly decode and display them. However, the inverse assembler will show data load/store operations as they occur on the bus without any byte-reversal.

Check the Storage Little-Endian Register (SLER) to help in setting the Endian Mode in preferences. If you have memory blocks with different endian modes

specified you will need to choose the mode that is for the memory you are trying to disassemble.

Note that the PPC405 does not use the same "Little-Endian" mode as defined in the PowerPC Architecture and used by the PPC401 and PPC403.

If you see "Unknown Opcode" or instructions that do not make sense, especially in code that is sequential and has no prefetch that may be thrown away, check the Endian Mode selection in preferences.

Burst Search Limit

For burst cycles the inverse assembler must determine where address and data are valid by counting clock cycles relative to edges on the "chip select" (~PerCSn) and "burst last" (~PerBLast) signals. Burst cycles may be very long and could cause the inverse assembler to operate slowly. The Burst Search Limit determines the maximum number of clock cycles that will be searched backward or forward from the point the inverse assembler is trying to disassemble. If the search backward encounters the limit a search forward is generally attempted. If both searches fail a "Burst search limit exceeded" message will be displayed. The default value for Burst Search Limit is 200. Raise this limit if you see one or more "Burst search limit exceeded" messages in the disassembly listing, and you do not mind the longer search time. You can raise it to a maximum of 20,000. If disassembly seems slow you can lower this limit as appropriate to a minimum of 20.

Instruction Search Limit

When the bus width is less than 32 bits (16 bits or 8 bits) the inverse assembler must piece together the two half-words (for 16-bit bus) or four bytes (for 8-bit bus) to make up a complete instruction for disassembly. The Instruction Search Limit indicates how many clock cycles will be searched forward and/or backward to find all of the pieces for the whole instruction. If DMA occurs, or the peripherals bus is granted to an external controller in the middle of fetching a multi-part instruction, there could be many clock cycles between one part of the instruction and other parts of the instruction. Searching for all of the parts of such an instruction could cause the inverse assembler to operate slowly, so a limit is imposed. If there are many wait states specified for each cycle, such as at the beginning of boot code, the Instruction Search Limit may be exceeded. If this occurs you will see one or more "Instruction search limit exceeded" in your listing display. The default value for the Instruction Search Limit is 200. You can increase the search limit to a maximum of 20,000.

A value of 2,000 will guarantee finding all of the instruction pieces for an 8-bit bus with no DMA or external controller cycles.

If the Instruction Search Limit is exceeded you will generally see a "Partial instruction" message at the word address (last two address bits are zero) of the instruction, and one to three "Instruction search limit exceeded" messages following it on the half-word and/or byte addresses. These messages are shown in white and cannot be removed by the inverse assembler filter to make sure they are seen. To show by how far the search limit is exceeded for an 8-bit bus make sure the filter's "Show States of Type" has "Extension Byte/Half-word" enabled.

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 PPC405 symbols
- User-defined symbols
- Object-file symbols

To view predefined PPC405 symbols

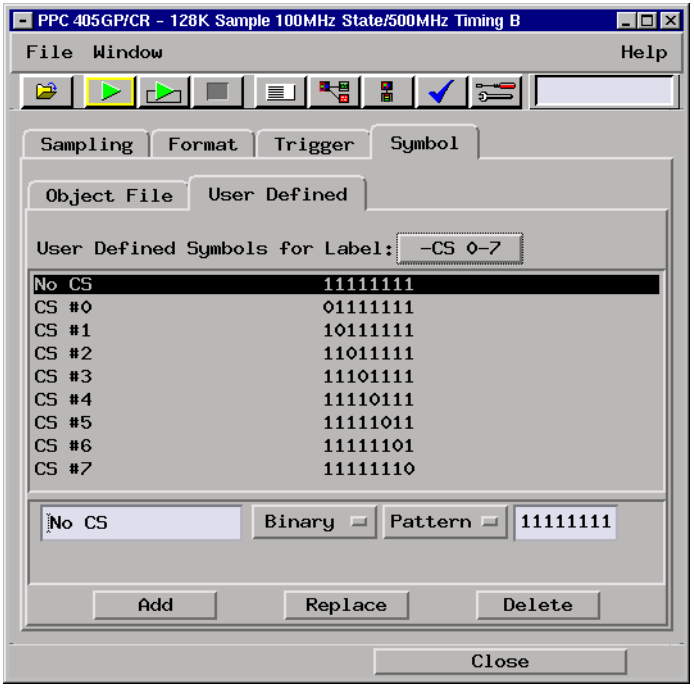
The PPC405 logic analyzer configuration files include predefined symbols. The predefined symbols for the PPC405 are listed in the following table.

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

To view the predefined symbols:

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

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



Predefined Logic Analyzer Symbol Descriptions

Label	Symbol	Encoding
R/-W	Read	1
	Write	0
-WBE 0-3	Byte, Lane #0	0111
	Byte, Lane #1	1011
	Byte, Lane #2	1101
	Byte, Lane #3	1110
	Half-word, Upper	0011
	Half-word, Lower	1100
	Word	0000
	No WBE	1111

Label	Symbol	Encoding
-CS 0-7	No CS	11111111
	CS #0	01111111
	CS #1	10111111
	CS #2	11011111
	CS #3	11101111
	CS #4	11110111
	CS #5	11111011
	CS #6	11111101
	CS #7	11111110
ACK 0-3	No DMA	0000
	DMA #0	1000
	DMA #1	0100
	DMA #2	0010
	DMA #3	0001

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.

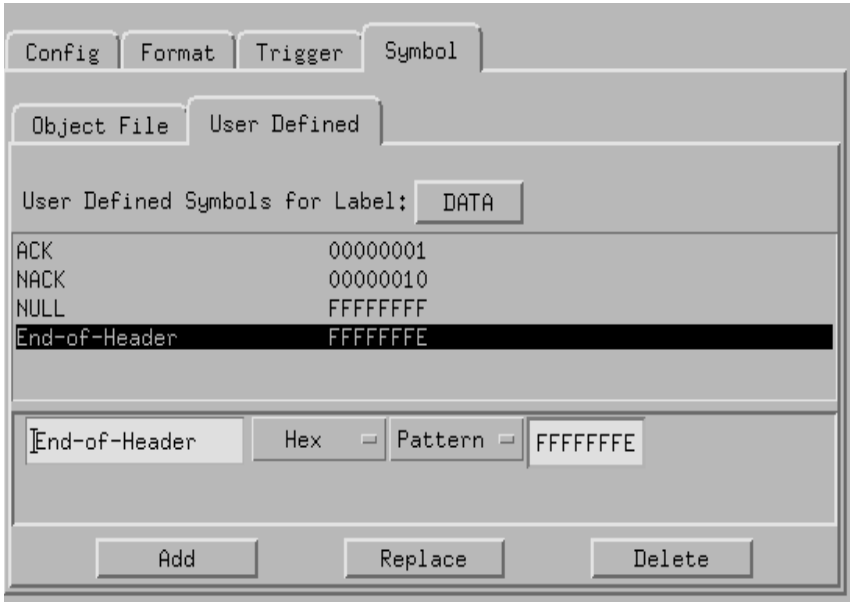
User-defined symbols are saved with logic analyzer configurations.

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.


Config
Format
Trigger
Symbol
Object File
User Defined
User Defined Symbols for Label:
DATA
ACK 00000001
NACK 00000010
NULL FFFFFFFF
End-of-Header FFFFFFFE
End-of-Header
Hex
Pattern
FFFFFFFE
Add
Replace
Delete

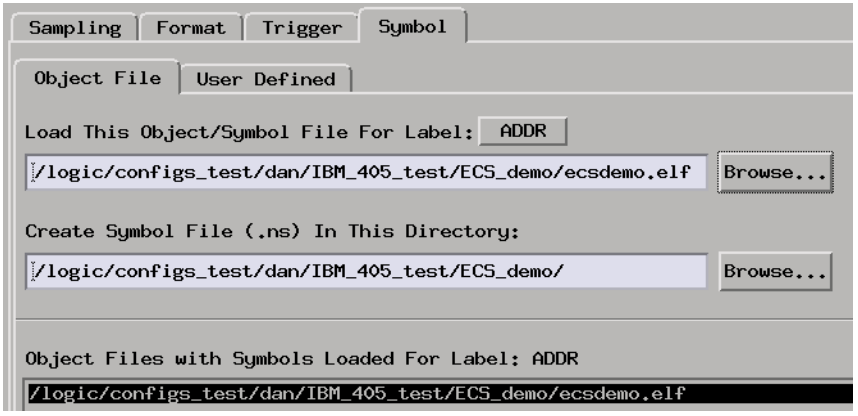
To load object file symbols

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 111).

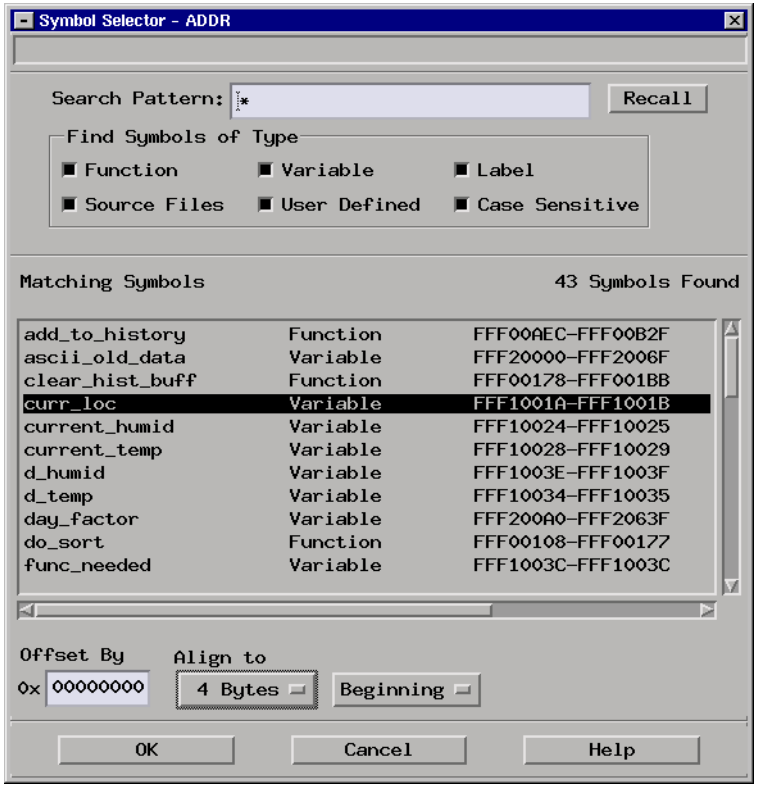
To load symbols in the 16700-series logic analysis system:

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



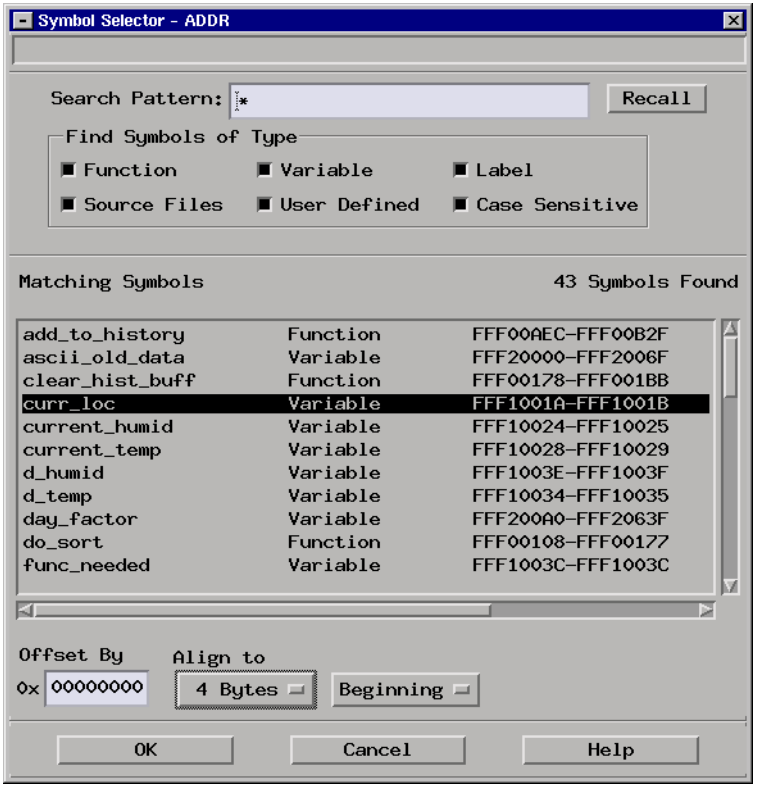
When you load object file symbols into the logic analyzer, a database of symbol/line number to address assignments is generated from the object file.

The Symbol Selector dialog allows you to view the 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.



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 Relocate Sections... dialog when you load the object file, or by using the Offset by field in the Symbol Selector dialog.



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

To adjust for prefetches, use a trigger offset of 0x10 (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 PPC405, an offset of 16 is large enough to overcome the prefetch queue.

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 E8171A inverse assembler provides PPC405 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

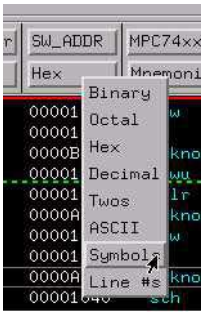
The PPC405 inverse assembler decodes captured data into program counter (PC) addresses (also known as software addresses) and assembly language mnemonics.

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 111).

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.

Setting Up Labels for Groups of Signals

Bit ordering conventions

The 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.		

Predefined Label Descriptions

The logic analyzer configuration files automatically set up labels for all signals. The following tables show some of the predefined labels for the most commonly used signals.

ADDR Label

ADDR	Logic Analyzer Name	PowerPC Name
Bit 0	ADDR[0]	PerAddr31
..	..	..
Bit 31	ADDR[31]	PerAddr0

DATA Label

DATA	Logic Analyzer Name	PowerPC Name
Bit 0	DATA[0]	PerData31
..	..	..
Bit 31	DATA[31]	PerData0

STAT Label

STAT	Logic Analyzer Name	PowerPC Name
Bit 0	-CS7	~PerCS7 [GPIO10]
Bit 1	-CS6	~PerCS6 [GPIO11]
Bit 2	-CS5	~PerCS5 [GPIO12]
Bit 3	-CS4	~PerCS4 [GPIO13]
Bit 4	-CS3	~PerCS3 [GPIO14]
Bit 5	-CS2	~PerCS2 [GPIO15]
Bit 6	-CS1	~PerCS1 [GPIO16]
Bit 7	-CS0	~PerCS0
Bit 8	-BLAST	~PerBLast
Bit 9	R/-W	PerR/~W
Bit 10	-WE PCIINT	~PerWE [PCIINT]
Bit 11	-OE	~PerOE
Bit 12	-WBE3	~PerWBE3
Bit 13	-WBE2	~PerWBE2
Bit 14	-WBE1	~PerWBE1
Bit 15	-WBE0	~PerWBE0
Bit 16	ERROR	PerErr
Bit 17	-EXT_ACK	~ExtAck
Bit 18	-EXT_REQ	~ExtReq
Bit 19	READY	PerREADY
Bit 20	DMA_EOT3	EOT3 [TC3]
Bit 21	DMA_EOT2	EOT2 [TC2]
Bit 22	DMA_EOT1	EOT1 [TC1]
Bit 23	DMA_EOT0	EOT0 [TC0]
Bit 24	DMA_ACK3	DMAAck3
Bit 25	DMA_ACK2	DMAAck2
Bit 26	DMA_ACK1	DMAAck1
Bit 27	DMA_ACK0	DMAAck0
Bit 28	DMA_REQ3	DMAReq3
Bit 29	DMA_REQ2	DMAReq2
Bit 30	DMA_REQ1	DMAReq1
Bit 31	DMA_REQ0	DMAReq0

Clock & Extra Status

CLK	Logic Analyzer Name	PowerPC Name
J	CLK	PerClk
K	(unused)	
L	SYS_ERR	SysErr
M	(unused)	
N *	(unused)	
P *	(unused)	

* Note: Only seen with 16550A, 16710A/11A/12A logic analyzers

To define additional labels

1. Open the Setup window.
2. Click the Format tab.
3. Click a label and select Insert before... or Insert after....
4. Click the signals under the appropriate pod, then select which bits to include in the label.

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

Changing the Analysis Mode

The logic analyzer can be set up to operate in the following analysis modes:

- State.
- Timing.

Inverse assembly is available in the state analysis mode.

To change to state analysis

In state mode, the logic analyzer uses the PerClk signal from the target system to capture data synchronously. This mode allows inverse assembly and is the default mode set up by the configuration files.

To configure the logic analyzer for state mode:

- Load the appropriate logic analyzer configuration file (see "To load configuration files (and the inverse assembler) from the system hard disk" on page 54).

  You can change the master clock setting by opening the logic analyzer's Setup window, selecting the **Format** tab, and clicking the **Master Clock** button to open the master clock dialog.

To change to timing analysis

In timing mode, the logic analyzer samples the processor pins asynchronously, according to an internal, adjustable sample rate clock. The minimum sample period for a 250 MHz timing analyzer is 4 ns.

Inverse assembly is not available in the timing analysis mode.

To configure the logic analyzer for timing analysis:

1. Load the appropriate logic analyzer configuration file (see "To load configuration files (and the inverse assembler) from the system hard disk" on page 54).
2. Open the logic analyzer's Setup window.
3. Select the Sampling tab.
4. Change the type option from State Mode to Timing Mode.

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 51.

- 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 51.

- 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 74.

- 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 PPC405 Execution,” beginning on page 93 for information on displaying captured data.

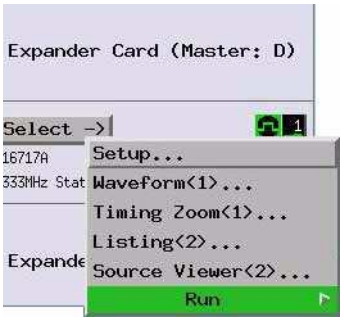
To Set Up Logic Analyzer Triggers

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.

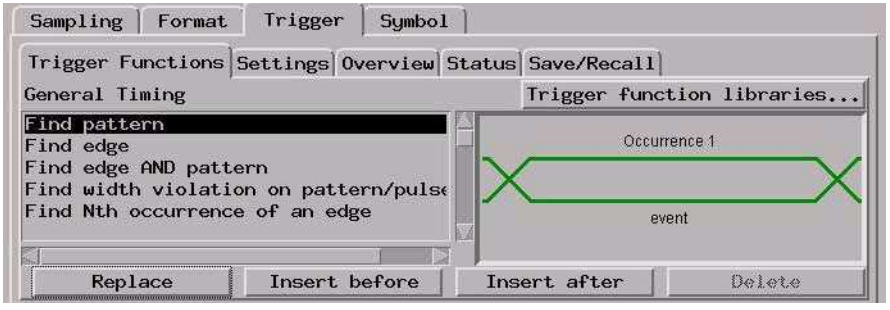
CAUTION: If you modify the trigger sequence to store only selected bus cycles, incorrect or incomplete disassembly may be displayed.

1 Open the logic analyzer's Setup window.

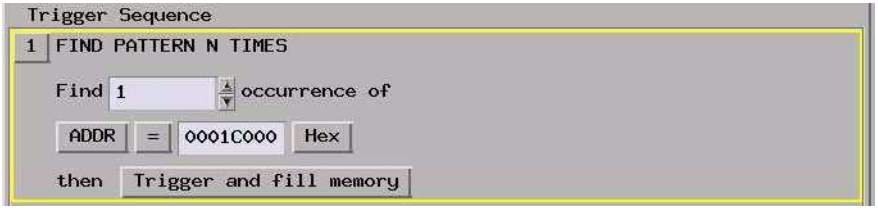


2 Select the Trigger tab.

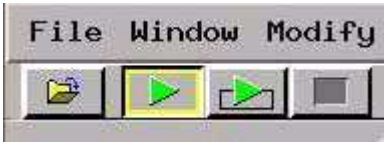
3 Select the trigger function that will be used in the logic analysis measurement and press the Replace button.



4 Set up the trigger sequence.



5 Run the measurement.



See Also

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

To Setup Trigger Alignment and Offset for Symbols and Source Code

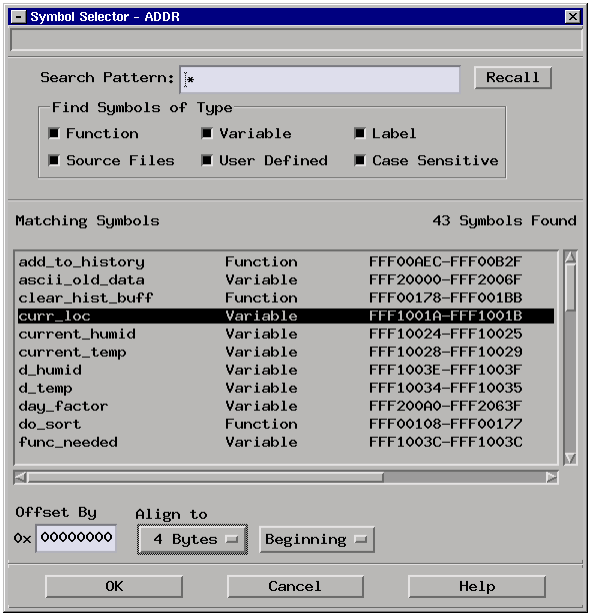
When setting up trigger specifications to capture PPC405 execution:

- Use the logic analyzer trigger alignment for proper triggering.
- Use the logic analyzer address offset to compensate for relocated code.

Using trigger alignment

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 and data load/store operations to the width of the program memory in bytes using the Align To menu in the Symbol Selector dialog.

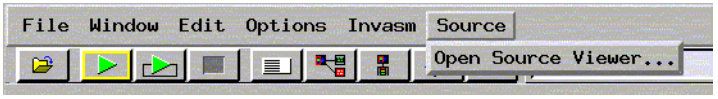


To compensate for relocated code, enter the nuber of bytes by which the code has been relocated in the Offset By field.

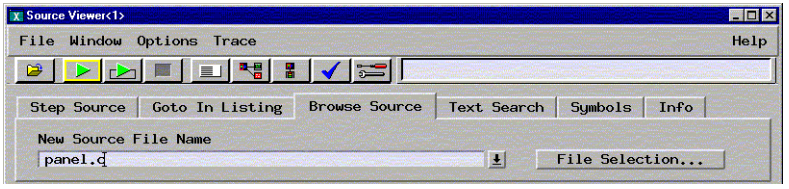
To Trigger on Source Code

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

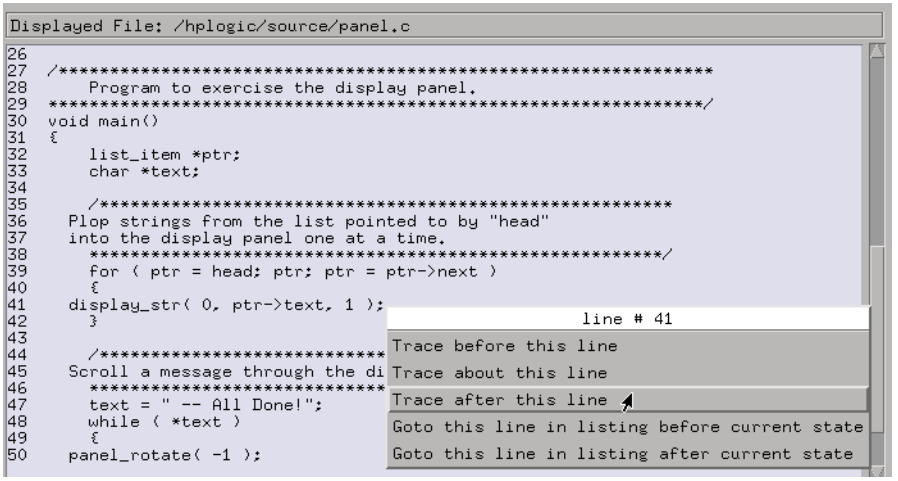
1 Open the Source Viewer window.



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



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



4 Run the measurement.

To avoid capturing library code execution

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

Logic analyzer storage qualification can be used to avoid capturing library code routines. The best way to do this is to specify a range of memory addresses for which states should not be stored.

NOTE: Do not configure the storage qualification to exclude wait states or idle states. The inverse assembler uses these states; for example, it uses them to determine sample points for burst cycles.

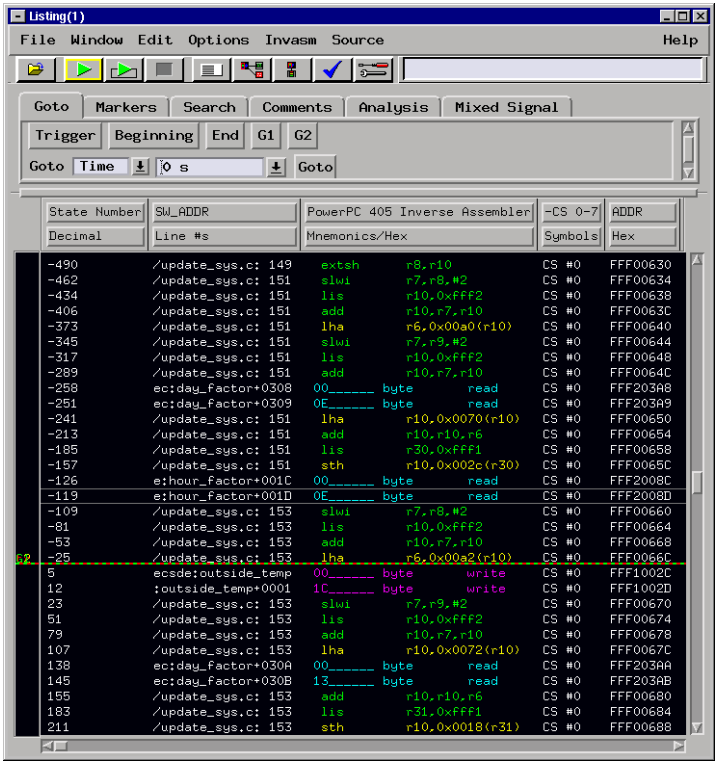
7

Displaying Captured PPC405 Execution

To Display Captured State Data

1 Open the Listing display window.

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



The inverse assembler is loaded when state configuration files are loaded, but it can also be loaded into a Listing display using the Invasm menu. The name of the inverse assembler file is I405E.

See Also

"To Use the Inverse Assembler" on page 96.

"To Use the Inverse Assembler Filters" on page 97 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 five choices: Load, Unload, 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 Filter and Options dialogs assist in analyzing and displaying data. The following sections describe these dialogs.

To load the inverse assembler

If you used the Setup Assistant, the inverse assembler is already loaded.

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.

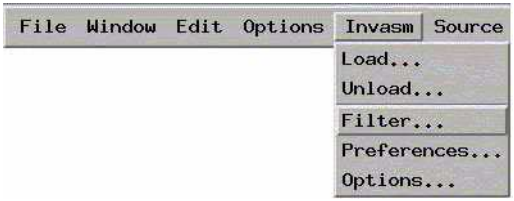
See Also

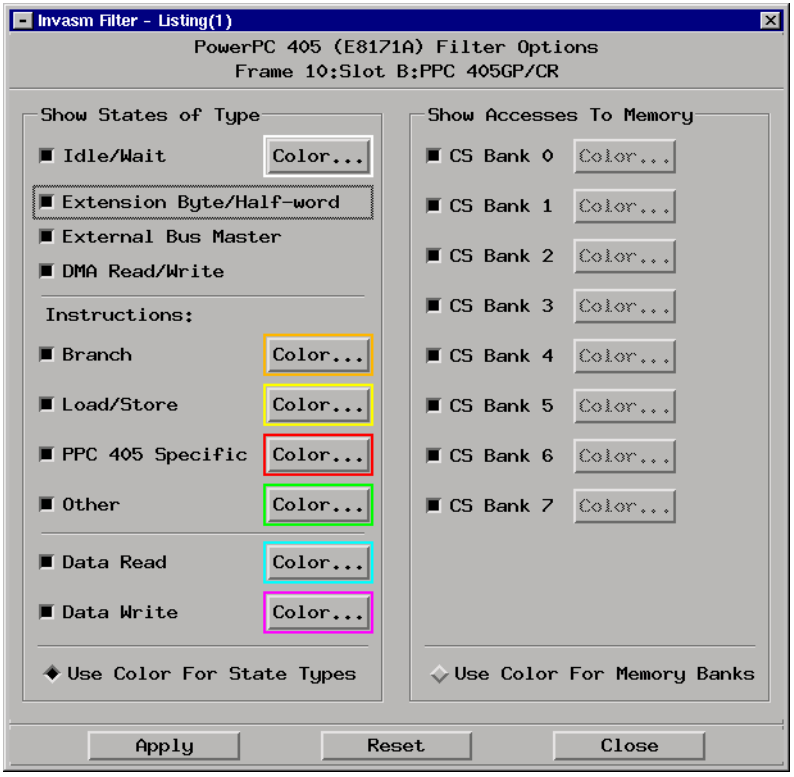
“Setup Assistant” on page 17.

“To load configuration files (and the inverse assembler) from the system hard disk” on page 54.

To Use the Inverse Assembler Filters

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





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 several 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.
- Accesses to a particular memory area can be suppressed, based on chip select memory bank.

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

To Interpret the Inverse Assembler Output

Data formats

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

Most numerical data is displayed in hexadecimal, for example, “stwu r1,0xfff8(r1)”.

Bit numbers and shift counts are displayed in decimal. You can set a prefix for decimal numbers in the Preferences dialog (page 66). For example, if you set the decimal prefix to “#”, the following simplified mnemonic would be displayed: “inslwi r1,r2,#5,#3”. (If “Enable Simplified Mnemonics” is not selected, the instruction would be displayed as “rlwimi r1,r2,#29,#3,#7”.)

A few instructions display their operands in binary with a “b” prefix, for example, “mtcrf b10100111,r2”.

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. Overfetch marking is not implemented in the PowerPC 405 inverse assembler.

Error messages

If you see error messages or other problems in the inverse assembly, refer to “Inverse Assembler Problems” on page 127.

To enable/disable the instruction cache on the PPC405

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.

Examples

The following examples assume that the processor is running in Real Mode (MMU is not enabled.)

To disable the instruction and data caches with the emulation module:

Use your debugger or the Emulation Control Interface to configure the ICCR and DCCR registers.

```
ICCR = 0x00000000   No memory regions cacheable (Instruction cache off)
ICCR = 0xFFFFFFFF   All memory regions cacheable (Instruction cache on)
ICCR = other values allow selected memory regions cacheable for instructions

DCCR = 0x00000000   No memory regions cacheable (Data cache off)
DCCR = 0xFFFFFFFF   All memory regions cacheable (Data cache on)
DCCR = other values allow selected memory regions cacheable for data
```

To disable the instruction cache with code:

```
        li      r1,0x0000
        mticcr  r1      # Disable instruction cache (all of memory).
        iccci   r0,r1   # Invalidate instruction cache.
        isync
```

To disable the data cache with code:

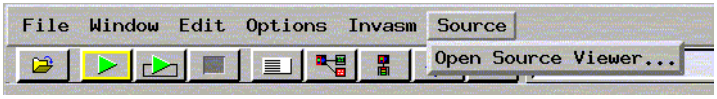
```
        li      r2,0x1000
        dcbf    r0,r2   # Flush the data cache block at 0x00001000 .
                        # Flush other data cache blocks as necessary.
        li      r1,0x0000
        mtdccr  r1      # Disable data cache (all of memory).
        li      r2,0x0080
        mtctr   r2
Loop:
        dccci   r0,r1   # Invalidate data cache.
        la      r1,0x0020(r1)
        bdnz    Loop
        sync
```

To View the Source Code Associated With Captured Data

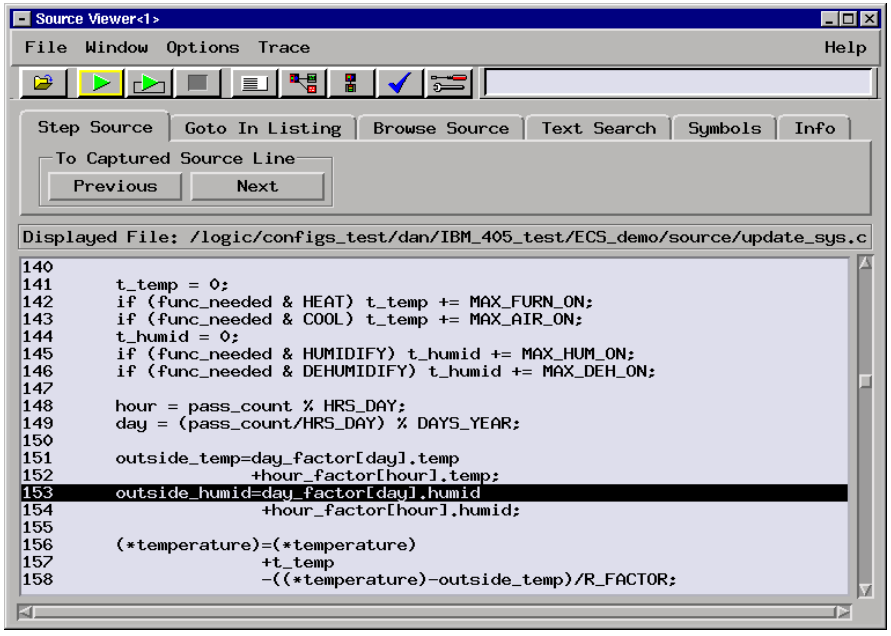
The B4620B Source Correlation Tool Set lets you view the high-level source code associated with captured data and set up triggers based on source code.

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

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



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



Inverse assembler generated SW_ADDR (software address) label

In the 16700-series logic analysis system, the PPC405 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.

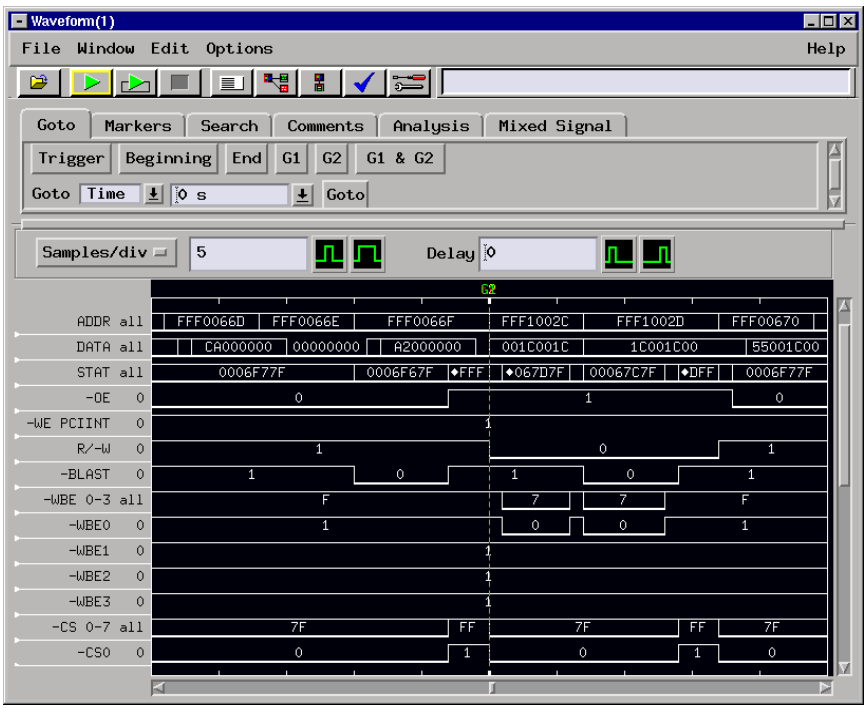
Source File Version Control. If the source code files are under a source code or version control utility, check the file names and paths carefully. These utilities can change source code file paths and file names. If these names are changed from the information contained in the symbol file, the source 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 and an emulation probe together to gain insight into your target system.

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

The Emulation Probe TRIG OUT Signal

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

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

- The most common method to generate the signal is to use the emulation probe to run then break processor execution (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 use the emulation probe to reset then break the processor. Going from the reset state of the processor to the “In Monitor” state will generate the signal.
- 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.

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.

Important reminders

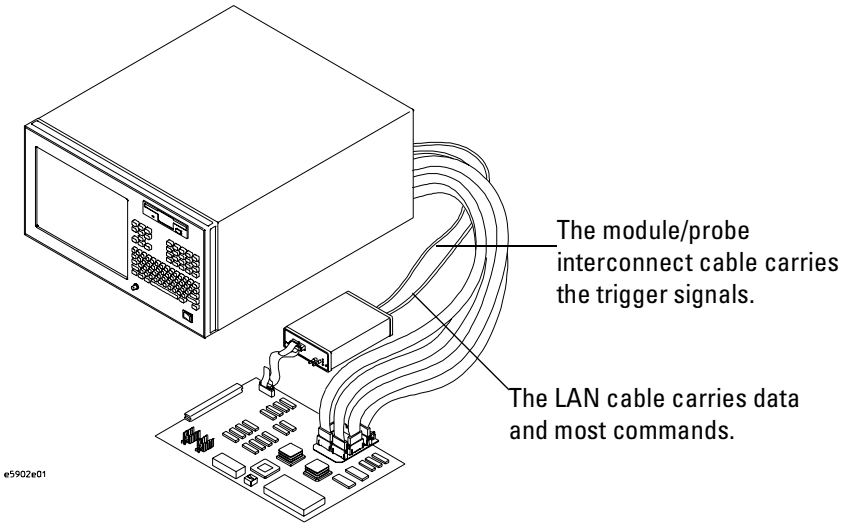
- Running a logic analyzer measurement does not run the processor. Usually you will need to start the logic analyzer measurement by selecting Run in the logic analyzer, then start the processor running by selecting Run in the Emulation Control Interface or your debugger.
- If the processor stops running during a logic analyzer measurement, you may need to manually stop the logic analyzer measurement before you see a listing.

See Also

The emulation probe *User's Guide* for the electrical characteristics of the TRIG OUT and BREAK IN signals.

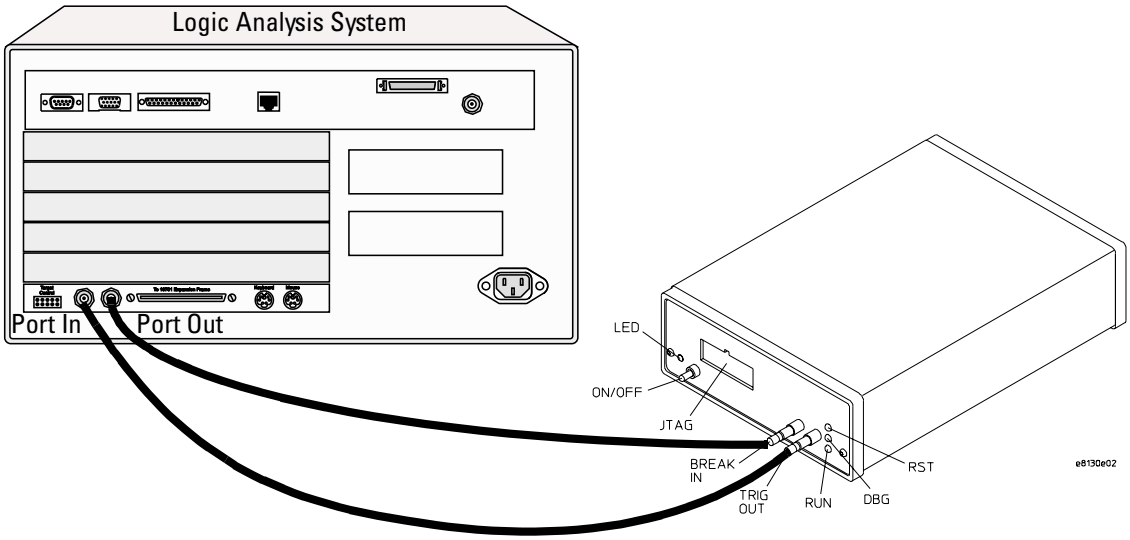
To connect the trigger signals to a logic analyzer with an emulation module

The module/probe interconnect cable connects the BREAK IN and TRIG OUT signals to the logic analysis system. You do not need to connect any cables to the SMB connectors on the front of the emulation probe.



To connect the trigger signals to a logic analyzer without an emulation module

1. Connect the emulation probe's BREAK IN connector to the logic analyzer's Port Out connector.
2. Connect the emulation probe's TRIG OUT connector to the logic analyzer's Port In connector.



To cross-trigger the emulation probe and a 16700-series logic analysis system

Cross-triggering features are available in the following places:

- The Intermodule window.

  Select the Intermodule icon in the main system window.

- The Source Viewer window.

  This window is available if you are using the B4620B Source Correlation Tool Set. From the Listing window, select Source Viewer from the Source menu.

See Also

For more detail on triggering features, see the online help in the Intermodule or Source Viewer window.

For some practical examples, select the Help icon in the main system window, then select "Measurement Examples."

9

General-Purpose ASCII (GPA) Symbol File Format

This chapter is a guide to the General-purpose ASCII format, which allows you to create symbol files without the help of a compiler or assembler.

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

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

[FUNCTIONS]
func_name start..end

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

[SOURCE LINES]
File: file_name
line# address

[START ADDRESS]
address

#Comments
```

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

Example

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

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

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

[VARIABLES]
total     40002000  4
value     40008000  4
```

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

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

SECTIONS

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

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

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

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

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

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

- NORMAL (default)—The section is a normal, relocatable section, such as code or data.
- NONRELOC—The section contains variables or code that cannot be relocated; this is an absolute segment.

Define sections first

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

Example

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

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

FUNCTIONS

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

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

`func_name` A symbol representing the function name.

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

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

Example

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

VARIABLES

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

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

`var_name` A symbol representing the variable name.

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

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

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

Example

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

SOURCE LINES

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

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

`file_name` The name of a file.

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

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

Example

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

START ADDRESS

```
[START ADDRESS]
address
```

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

Example

```
[START ADDRESS]
00001000
```

Comments

```
#comment text
```

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

Example

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

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 126 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 each logic analyzer pod is connected to the correct connector.

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

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

- ❏ Verify that the STAT, DATA, and ADDR format labels have not been modified from their default values.

  These labels must remain as they are configured by the configuration file. Do not change the names of these labels or the bit assignments within the labels. See“Configuring the Logic Analyzer” on page 51 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 storage qualification has not excluded storage of all the needed opcodes and operands.

- ❏ Verify that storage qualification has not excluded wait states or idle states.

  The PowerPC 405 inverse assembler uses idle states and wait states as reference points and to count cycles during burst cycles.

- ❏ Verify that the endian selection is correct in the preferences menu (see page 67).

Inverse assembler will not load or run

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

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

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

  See Chapter 3, “Setting Up the Logic Analysis System,” beginning on page 39 for details.

Error messages in the listing

Error messages in the inverse assembly listing are usually caused by incorrect settings in the Preferences dialog (page 57). Most messages will tell you which setting caused the problem; if necessary, widen the “Inverse Assembler” column to see the entire message.

Here is a list of the messages you may encounter:

- No chip select banks/regions enabled - See Preferences...
- Address not found in any bank - See Preferences...
- Address not found in any region - See Preferences->Details...
- Address found only in disabled bank - See Preferences...
- Bank used does not match hardware chip select - See Preferences...
- Port size used does not match hardware WBE0-3 - See preferences...
- Inst. address does not conform to bus width - See preferences...
- Burst cycle seen with burst disabled - See Preferences->Details...
- Invalid burst value WBF > FWT or BWT - See Preferences->Details...
- Invalid burst value CSN > FWT - See Preferences->Details...
- Burst search limit exceeded - See Preferences...
- Instruction search limit exceeded - See Preferences...

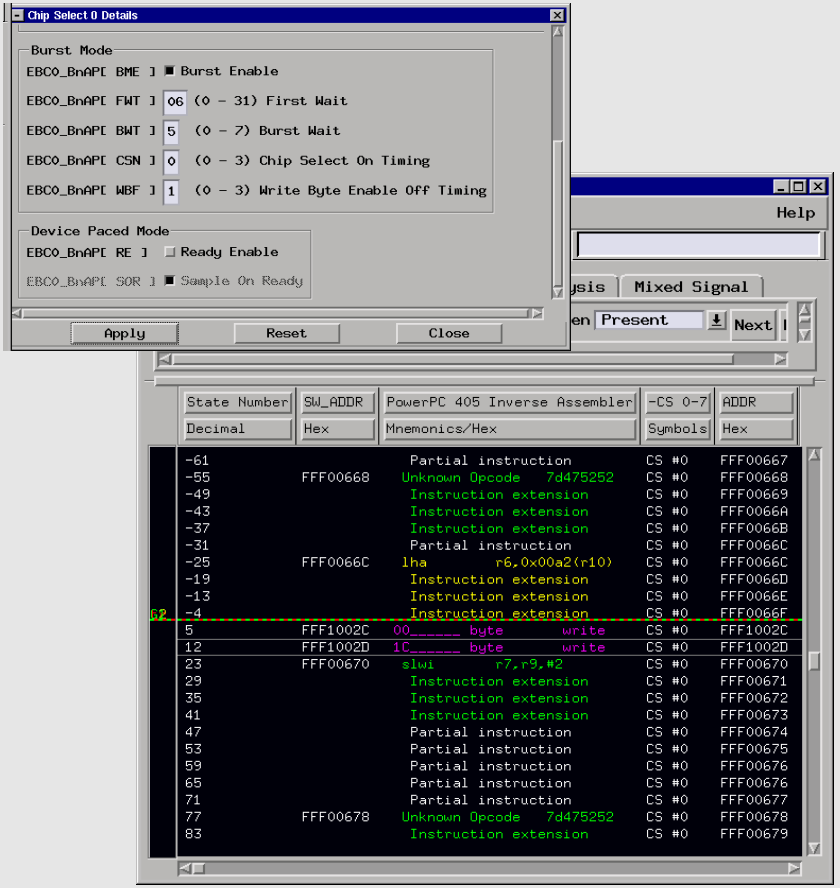
Partial instructions in the listing

"Partial instruction" messages are displayed when the inverse assembler is unable to identify all of the states corresponding to an instruction.

One cause of "Partial instruction" messages is if the Burst Wait (BWT) is not configured correctly in the Preferences dialog.

Example

In the following example, the BWT should be 6, but was set to 5. Note that the inverse assembly is correct near the trigger, but gets worse as timing problems accumulate away from the trigger.



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.

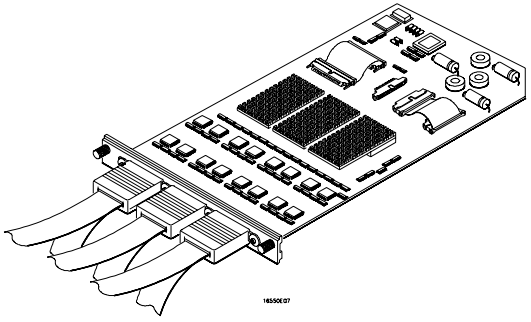
Ensure that the inverse assembler file is not renamed or deleted, and that it is located in /logic/ia.

See Chapter 3, "Setting Up the Logic Analysis System," beginning on page 39 for details.

"Measurement Initialization Error"

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

Cable Connections for One-Card 16550A Installations



See Also

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

Service Guide.

"No Configuration File Loaded"

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

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

See Also

See "To load configuration files (and the inverse assembler) from the system hard disk" on page 54 for details on loading configuration files.

"Selected File is Incompatible"

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

"Slow or Missing Clock"

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

probe interface. See Chapter 3 to determine the proper connections.

"Time from Arm Greater Than 41.93 ms"

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

"Waiting for Trigger"

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

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

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 the E8171A software.

Operating characteristics

The following operating characteristics are not specifications, but are typical operating characteristics for the E8171A Inverse Assemblers.

Operating Characteristics		
Microprocessor Compatibility	IBM PowerPC 405GP/CR	
Maximum Supported Microprocessor Bus Speed	100 MHz (70 MHz for the 16554A logic analyzer)	
Microprocessor Features	Peripherals bus PCI bus (PPC405GP) SDRAM bus Code compression Instruction & Data Caches Real Mode MMU Mode	Supported Not Supported Not Supported Not Supported Disable caches for best results Supported Not Supported
Logic Analyzers Supported	16550A (one or two cards) 16554/55/56/57 (two or more cards) 16710/11/12A (one or two cards) 16715/16/17/18/19A (two or more cards) 16750/51/52A (two or more cards)	

Operating Characteristics	
Probes Required	The inverse assembler requires six logic analyzer pods (102 channels).
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. 16550A: 3.5 ns 16554A, 16555A, 16555D: 3.5 ns 16556A/D: 4.0 ns 16557D: 3.0 ns 16710/11/12A: 4.0 ns 16715/16/17/18/19A: 2.5 ns (1.25 ns using eye finder) 16750/51/52A: 2.5 ns (1.25 ns using eye finder).

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 A set of tools for debugging your target system. A solution includes 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

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software Clause in DFARS 252.227-7013. Agilent Technologies Inc., 395 Page Mill Road, Palo Alto, CA 94303-0870 U.S.A. Rights for non-DOD U.S. Government Departments and Agencies are set forth in FAR 52.227-19 (c) (1,2).

Document Warranty

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

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

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

Safety

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

Warning

• Before turning on the instrument, you must connect the protective earth terminal of the instrument to the protective conductor of the (mains) power cord. The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. You must not negate the protective action by using an extension cord (power cable) without a protective conductor (grounding). Grounding one conductor of a two-conductor outlet is not sufficient protection.

• Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuseholders. To do so could cause a shock of fire hazard.

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

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

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

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

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

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

Safety Symbols

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

Hazardous voltage symbol.

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

WARNING

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

CAUTION

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

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

Product Warranty

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

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

For products returned to Agilent Technologies for warranty service, the Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to the Buyer. However, the Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

Agilent Technologies warrants that its software and firmware designated by Agilent Technologies for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent Technologies does not warrant that the operation of the instrument software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer- supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

No other warranty is expressed or implied. Agilent Technologies specifically disclaims the implied warranties of merchantability or fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are the buyer's sole and exclusive remedies. Agilent Technologies shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Assistance

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

Certification

Agilent Technologies certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

About this edition

This is the *Agilent E8171A Logic Analysis Support for the IBM PowerPC 405 User's Guide*.

Publication number
E8171-97000, February 2001
Printed in USA.

Print history is as follows:
First edition, February 2000

New editions are complete revisions of the manual. Many product updates do not require manual changes, and manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.