User's Guide

Publication number E2498-97004
December 1999

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

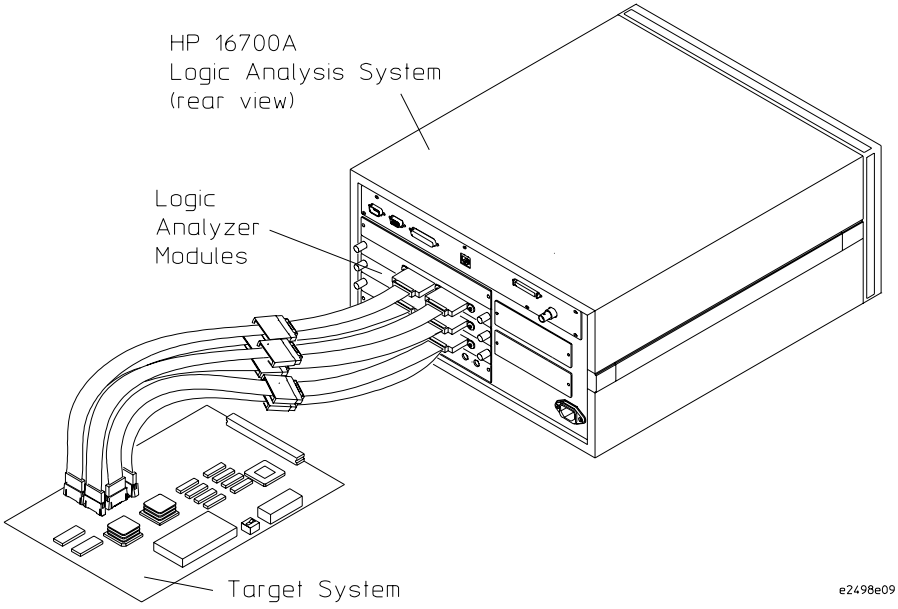
Solutions for the PowerPC 7XX

This manual describes how to connect an HP logic analysis system to your target system using custom connectors. It also describes how to connect an emulation module (for an emulation solution).

Inverse Assembler Software

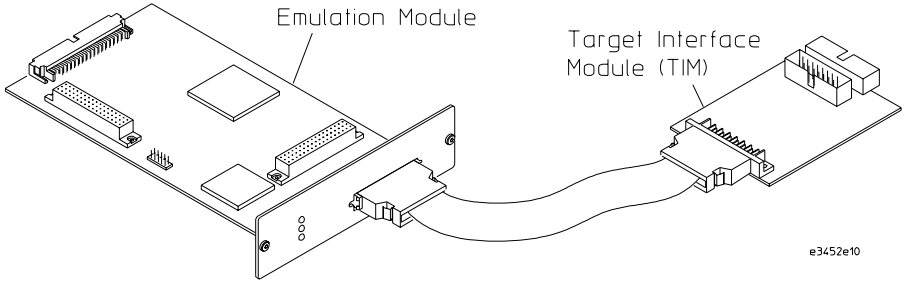
The custom connectors connect your logic analyzer to your target system for state and timing analysis using the inverse assembler software. The inverse assembler can be used with an HP 16600A/700A-series logic analysis system or with other HP logic analyzers.

The inverse assembler software can be purchased alone, or as part of an emulation solution.



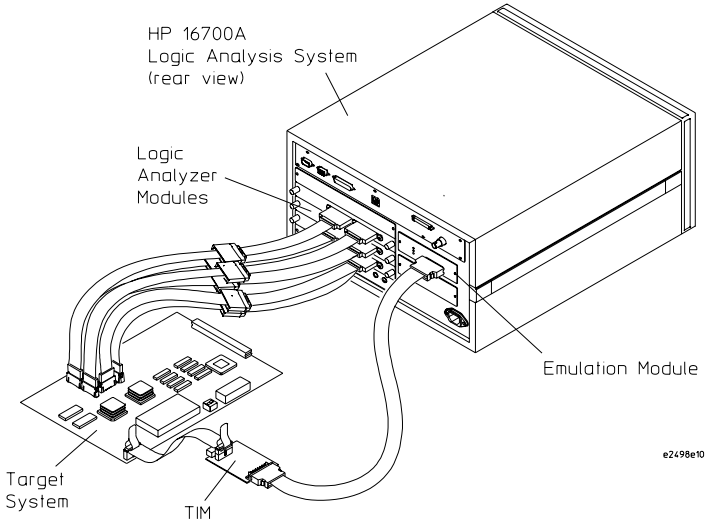
Emulation Module and Target Interface Module

The emulation module plugs into your HP 16600A/700A-series logic analysis system frame. The emulation module lets you use the target processor's built-in background debugging features, including run control and access to registers and memory. A high-level source debugger can use the emulation module to debug code running on the target system. You can use the target interface module (TIM) to connect the emulation module to a debug port on the target system.



Emulation Solution

The emulation solution includes the inverse assembler software, an emulation module, cables and adapters, and the HP B4620B Source Correlation Tool Set (for analyzing high-level source code). This solution is designed to be used with an HP 16600A/700A-series logic analysis system.



In This Book

This book documents the following products:

Inverse Assembler Software

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

Emulation Solution

Processors supported	Product ordered	Includes
PPC 740 and PPC 750	HP E9486A Option #001	HP E2498A inverse assembler , HP 16610A emulation module, target interface module (TIM), HP B4620B Source Correlation Tool Set

Tips To Save You Time

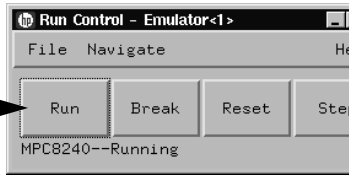
Use the Setup Assistant

Click here to connect the logic analyzer cables, and automatically load the correct configuration files. See page 23.

Use the appropriate Run button

Click here to start a measurement.

Click here to run the target microprocessor.



Additional Information Sources

Additional or updated information can be found in the following places:

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

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

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

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

If you have an HP 16600A or HP 16700A logic analysis system, the **online help** for the Emulation Control Interface has additional information on using the emulation module.

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

Contents

Contents

Contents

4 Analyzing the PPC7XX with an HP 16600A/16700A-series Logic Analyzer

12 Troubleshooting the Inverse Assembler

13 Troubleshooting the Emulation Module

1

Overview

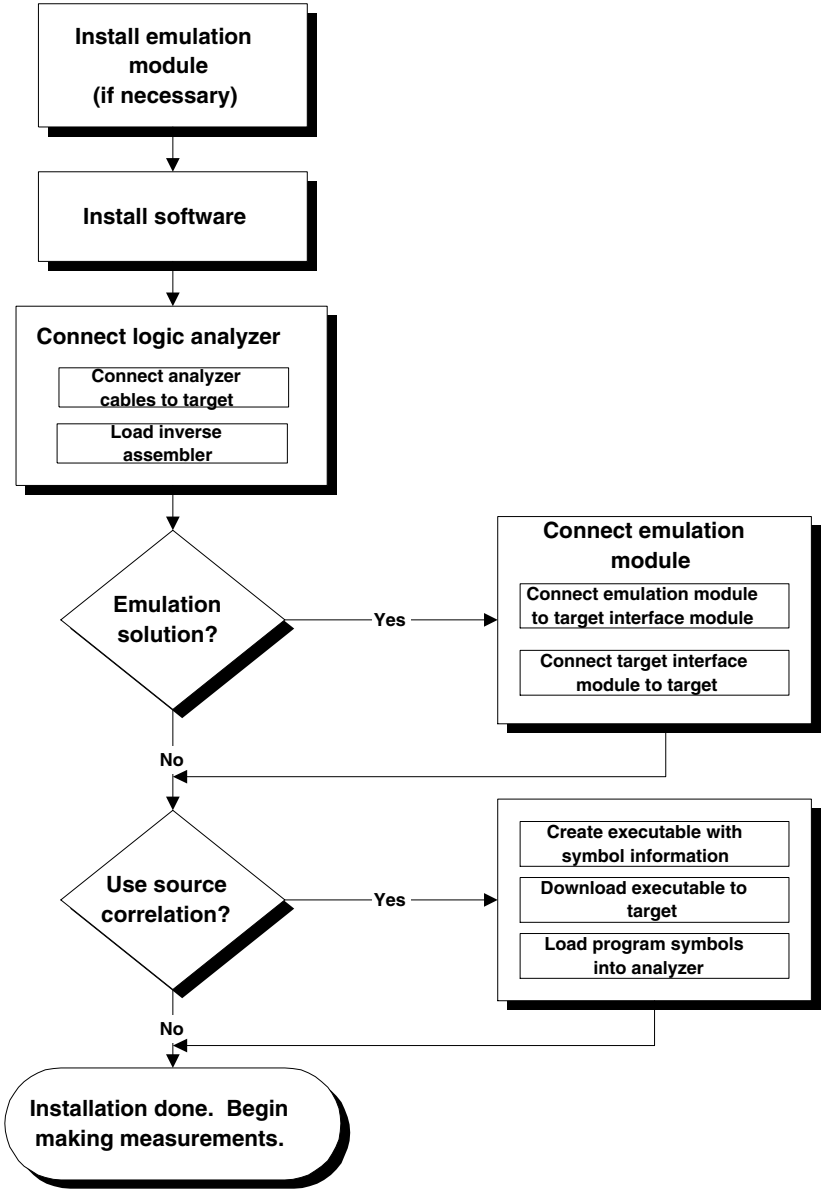
This chapter describes:

- Setup Checklist
- Setup Assistant
- Equipment used with the inverse assembler software (including a list of logic analyzers supported)
- Equipment used with the emulation module
- Additional information sources

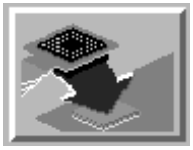
Setup Checklist

Follow these steps to connect your equipment:

- Check that you received all of the necessary equipment. See page 290 and page 28.
- If you need to install an emulation module in an HP 16600A/700A series logic analysis system, see page 150.
- Install the software. See page 33.
- If you have an HP 16600A/700A-series logic analysis system, use the Setup Assistant to help you connect and configure the logic analysis system and emulation module. See page 23.
- If you do not have an HP 16600A/700A-series logic analysis system, connect the logic analyzer cables. See page 39.


Install emulation module (if necessary)
Install software
Connect logic analyzer
Connect analyzer cables to target
Load inverse assembler
Emulation solution?
Yes
Connect emulation module
Connect emulation module to target interface module
Connect target interface module to target
No
Use source correlation?
Yes
Create executable with symbol information
Download executable to target
Load program symbols into analyzer
No
Installation done. Begin making measurements.
E2498F01.VSD

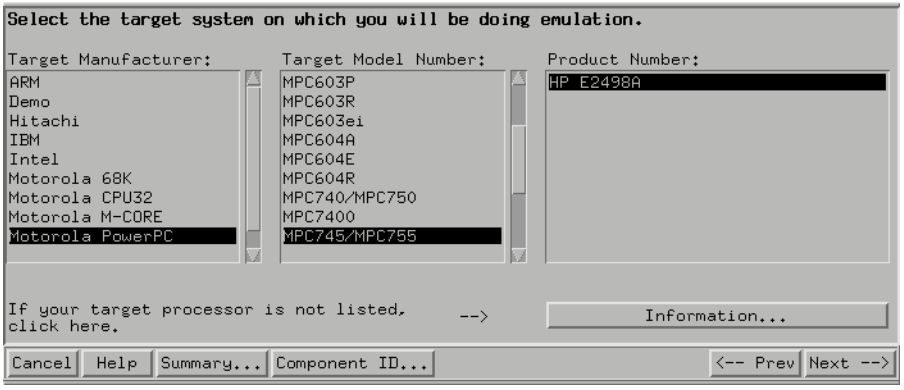
Setup Assistant



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 HP 16600A and HP 16700A-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 clicking its icon in the system window.



If you ordered this inverse assembler software or emulation solution with your HP 16600A/700A-series logic analysis system, the logic analysis system has the latest software installed, including support for this product. If you received this product after you received your logic analysis system, see the "Installing Software" chapter (page 33).

Inverse assembler software

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

Equipment supplied

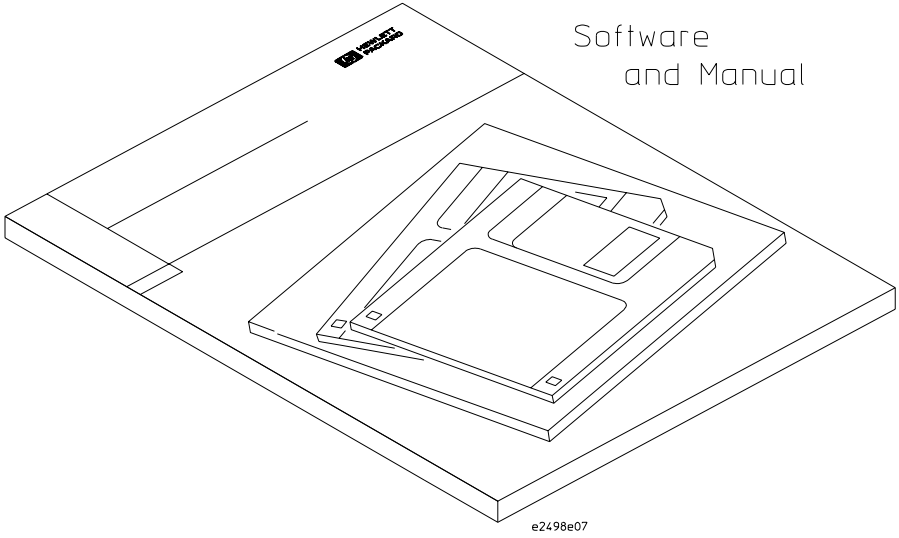
The HP E2498A consists of the following:

- Logic analyzer configuration files and the inverse assembler on a CD-ROM (for HP 16600A/700A series logic analysis systems).
- Logic analyzer configuration files and the inverse assembler on a 3.5-inch disk (for other HP logic analyzers).
- The configuration software for the HP 16505A Prototype Analyzer on a 3.5-inch disk.
- This User's Guide.

Minimum equipment required

For state and timing analysis of a PowerPC 7XX target system, you need all of the following items.

- The HP E2498A PowerPC 7XX Inverse Assembler.
- The appropriate connectors on the target system. Chapter 3 contains information on designing the appropriate connectors into the target system. You could use General Purpose probes instead of connectors.
- One of the logic analyzers listed on page 26. The logic analyzer software version requirements are listed on page 27.



Equipment Supplied with the HP E2498A Inverse Assembler Software

Additional equipment supported

HP B4620B Source Correlation Tool Set

The inverse assembler software may be used with the HP B4620B Source Correlation Tool Set on an HP 16600A/700A-series logic analysis system.

Logic analyzers supported

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

The HP E2498A requires eight logic analyzer pods (136 channels) for inverse assembly. The inverse assembler software contains two additional pods that you can monitor.

Logic Analyzers Supported

Logic Analyzer	Channel Count	State Speed	Timing Speed	Memory Depth
16600A	204	100 MHz	125 MHz	64 k states
16601A	136	100 MHz	125 MHz	64 k states
16550A (two cards)	102/card	100 MHz	250 MHz	4 k states
16554A (two or three cards)	68/card	70 MHz	125 MHz	512 k states
16555A (two or three cards)	68/card	110 MHz	250 MHz	1 M states
16555D (two or three cards)	68/card	110 MHz	250 MHz	2 M states
16556A (two or three cards)	68/card	100 MHz	200 MHz	1 M states
16556D (two or three cards)	68/card	100 MHz	200 MHz	2 M states
16557D (two or three cards)	68/card	135 MHz*	250 MHz	2 M states
1660A/AS/C/CS/CP	136	100 MHz	250 MHz	4 k states
1670A	136	70 MHz	125 MHz	64 k or .5 M states
1670D	136	100 MHz	125 MHz	64 k or 1 M states

* For the HP 16557D, the state speed decreases for four- or five-card configurations.

Logic analyzer software version requirements

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

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

If your software version is older than those listed, load new system software with the higher version numbers before loading the HP E2498A software.

Logic Analyzer Software Version Requirements

Logic Analyzer	Minimum Logic Analyzer Software Version for use with HP E2498A
HP 16600 Series	The latest HP 16600 logic analyzer software version is on the CD-ROM shipped with this product.
HP 1660A/AS Series	3.01
HP 1660C/CS/CP Series	A.02.01
HP 1670A/D Series	A.02.02
Mainframes*	
HP 16700 Series	The latest HP 16700 logic analyzer software version is on the CD-ROM shipped with this product.
HP 16500C Mainframe	1.07
HP 16500B Mainframe	3.14

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

The HP 16505A provides a windowed user interface for the HP 16500B/C Logic Analysis System. Refer to the *HP 16505A Prototype Analyzer Installation Guide* for information on connecting the HP 16505A to the HP 16500B/C Logic Analysis System.

Emulation Module

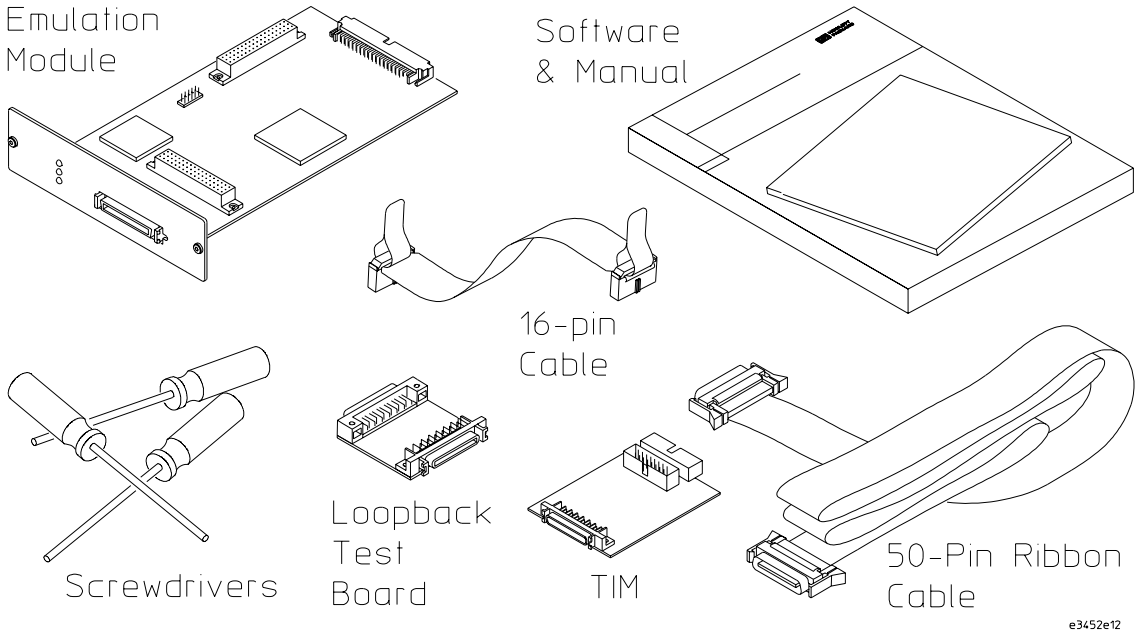
This section lists equipment supplied with the emulation module and lists the minimum equipment required to use the emulation module.

Equipment supplied

The equipment supplied with your emulation module includes:

- An HP 16610A emulation module. If you ordered an emulation module as part of your HP 16600A or HP 16700A-series logic analysis system, it is already installed in the frame.
- A target interface module (TIM) circuit board.
- An emulation module loopback test board (HP part number E3496-66502).
- Firmware for the emulation module and/or updated software for the Emulation Control Interface on a CD-ROM.
- A 50-pin ribbon cable for connecting the emulation module to the target interface module.
- A 16-pin ribbon cable for connecting the target interface module to the target system.
- One Torx T-8, one Torx T-10, and one Torx T-15 screwdriver.
- This User's Guide.

Equipment Supplied with the Emulation Module



Minimum equipment required

The following equipment is required to use the emulation module:

- A method for connecting to the target system. You can design a debug port connector on the target system. See the section titled “Designing a Target System for the Emulation Module” on page 154 for information on designing a debug port on the target system.
- An HP 16600A or HP 16700A logic analysis system.
- A user interface, such as a high-level source debugger or the logic analysis system's Emulation Control Interface.

Power Up Configuration Requirement

When it powers up, the emulation module tries to read the PVR register in the target processor. If it could read the PVR register, it would configure itself correctly. It can't read the PVR register because your target system is powered up last. In order to get the emulation module to read the PVR register in the target processor and configure itself correctly, simply select the emulation module icon in the interface and click Update Firmware. Then in the Update Firmware window, click Display Current Version.

Failure to make the emulation module read the PVR register in the target microprocessor may cause the emulation module to be configured incorrectly.

Emulation Solution

An emulation solution uses the equipment and software already described in this chapter.

The combination of an inverse assembler, an emulation module, and an HP 16600A or HP 16700A logic analysis system lets you both trace and control microprocessor activity on the target system.

The inverse assembler software sets up the logic analyzer to assign and properly interpret the signals from the target microprocessor to the logic analyzer.

You can use a debugger or the logic analysis system's Emulation Control Interface to configure and control the target processor and to download program code.

Additional Information Sources

Additional or updated information can be found in the following places:

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

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

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

If you have an HP 16600A or HP 16700A logic analysis system, the **online help** for the Emulation Control Interface has additional information on using the emulation module.

The **measurement examples** include valuable tips for making emulation and analysis measurements. You can find the measurement examples in the online help in your HP 16600A/700A logic analysis system.

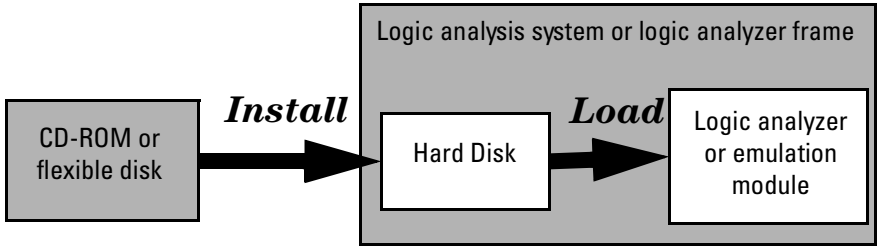
2

Installing Software

This chapter explains how to install the software you will need for your analysis probe or emulation solution.

Installing and loading

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



What needs to be installed

HP 16600A/700A-series logic analysis systems

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

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

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

The HP 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.

The following files can be installed from the floppy disk that contains your inverse assembler.

- Logic analysis system configuration file
- Inverse assembler (automatically loaded with the configuration file)

Other HP logic analyzers

The following files can be installed from a floppy disk:

- Logic analyzer configuration files, which automatically load the inverse assembler

To install the software from CD-ROM (HP 16600A/700A)

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 HP 16600A/700A 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.
2. Insert the CD-ROM in the drive.
3. Click the System Admin icon.
4. Click Install... .

   Change the media type to "CD-ROM" if necessary.
5. Click Apply.
6. From the list of types of packages, select "PROC-SUPPORT."

   A list of the processor support packages on the CD-ROM will be displayed.
7. Click on the "POWERPC7XX" package.

   If you are unsure if this is the correct package, click Details for information on what the package contains.
8. Click Install... .

   The dialog box will display "Progress: completed successfully" when the installation is complete.
9. Click Close.

   The configuration files are stored in /hplogic/configs/hp/ppc7xx/.

   The inverse assemblers are stored in /hplogic/ia.

See Also

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

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

To list software packages which are installed (HP 16600A/700A)

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

To install software on other logic analyzers

Consult the documentation for your logic analyzer for details.

3

Connecting the Target System to a Logic Analyzer

This chapter shows you how to connect the logic analyzer to the target system.

If you are connecting to an HP 16600A-series or HP 16700A series logic analyzer, and have designed connectors into the target system as described in this chapter, use the Setup Assistant to connect and configure your system (see page 23). Use this manual for additional information, if desired.

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

- Check that the target system meets the necessary requirements. See page 41.
- Read the power on/power off sequence. See page 47.
- Connect the target system to the logic analyzer. See page 48.
- Configure the logic analyzer. See page 67.

Target System Requirements

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

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

The following sections contain information on designing or using these approaches. The signal-to-connector mapping, showing which signals must go to which logic analyzer connector, is shown in "Inverse assembler—signal-to-connector mapping" on page 227.

Direct Probing with General Purpose (GP) Probes

If you are using General Purpose probes, connect the individual probes to the signals according to the tables in the section titled "Inverse assembler—signal-to-connector mapping" on page 227.

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

Designing and using built-in connectors

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

- The board real estate 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. See the "Hardware Reference" chapter for pinouts.

The connection scheme shown in this section uses 38-pin 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.

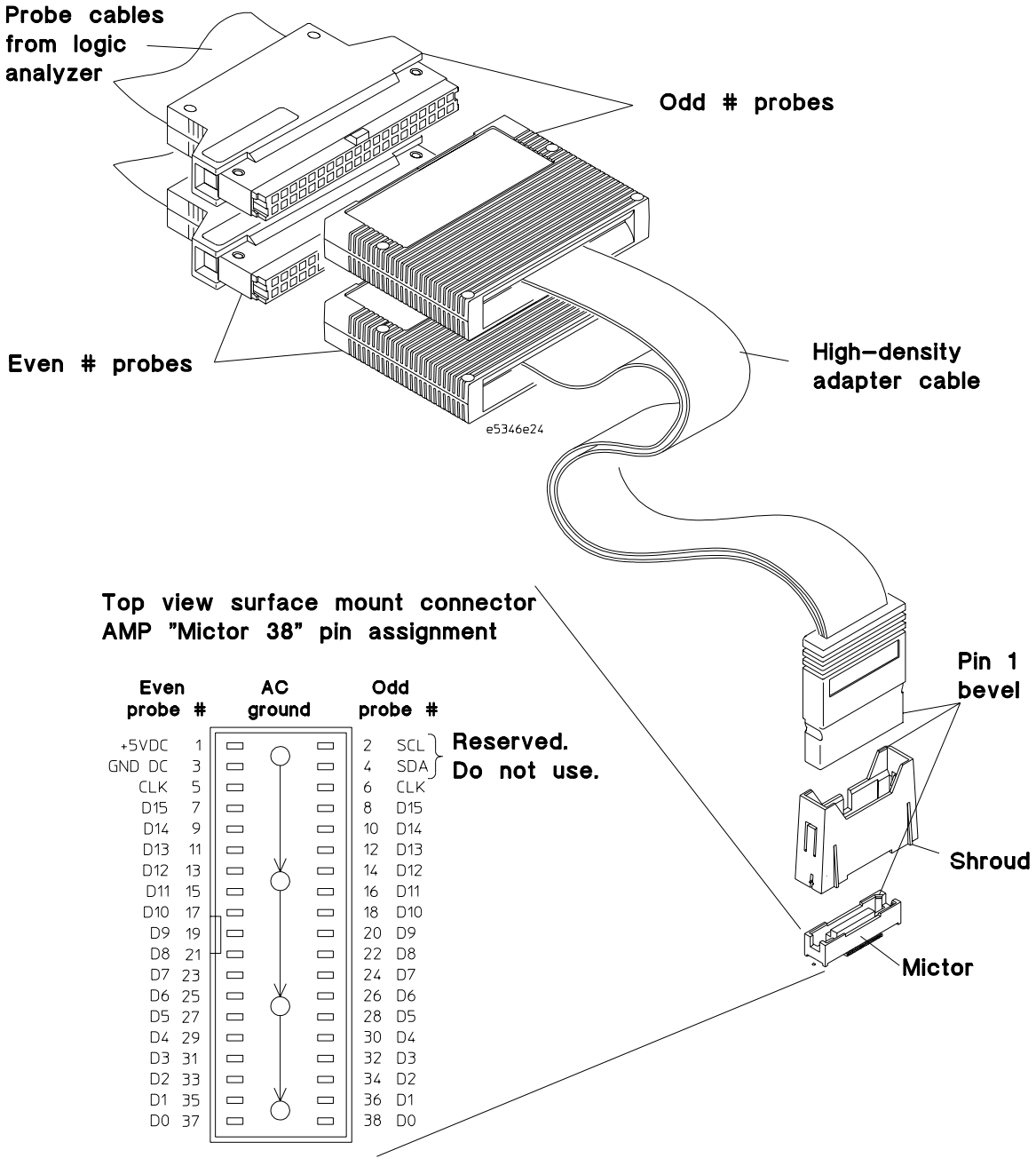
Part Numbers for Built-in Connectors and Cables

Part Number	Description
HP 1252-7431, or AMP 2-767004-2	2 x 19 header. A minimum of four connectors (eight logic analyzer pods) is required; a fifth connector may be used.
HP E5346-44701	Optional connector-support shroud
HP E5346A	High-density termination cable. One required for each 2x19 connector.

See Also

The *Passively Probing a Motorola/IBM PowerPC 740/750 Target System with HP E5346A High-Density Termination Adapters* application note for information on signal routing.

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 (HP 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 "Hardware Reference" chapter for the tables showing the microprocessor signals for 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.

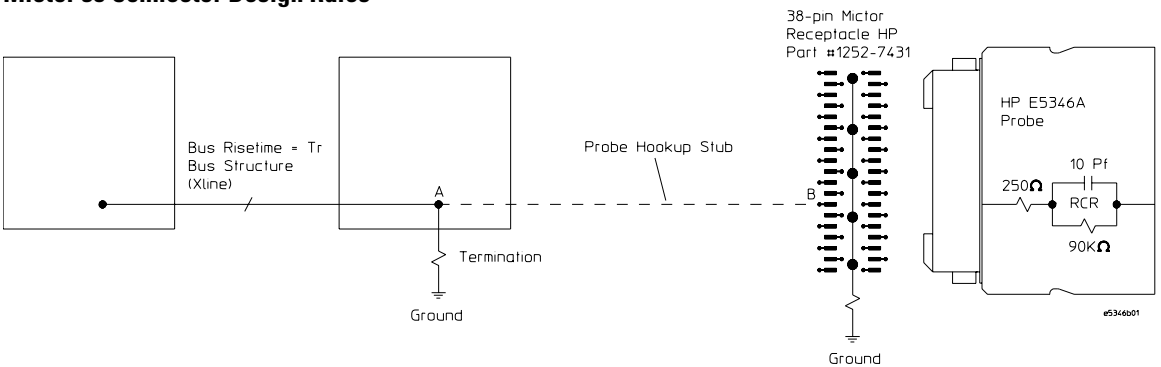
To increase the structural support for the cables, you can also use cable support shrouds (HP part number E5346-44701) on each connector. The figures on the following page show the mechanical layouts for the shrouds and headers.

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.

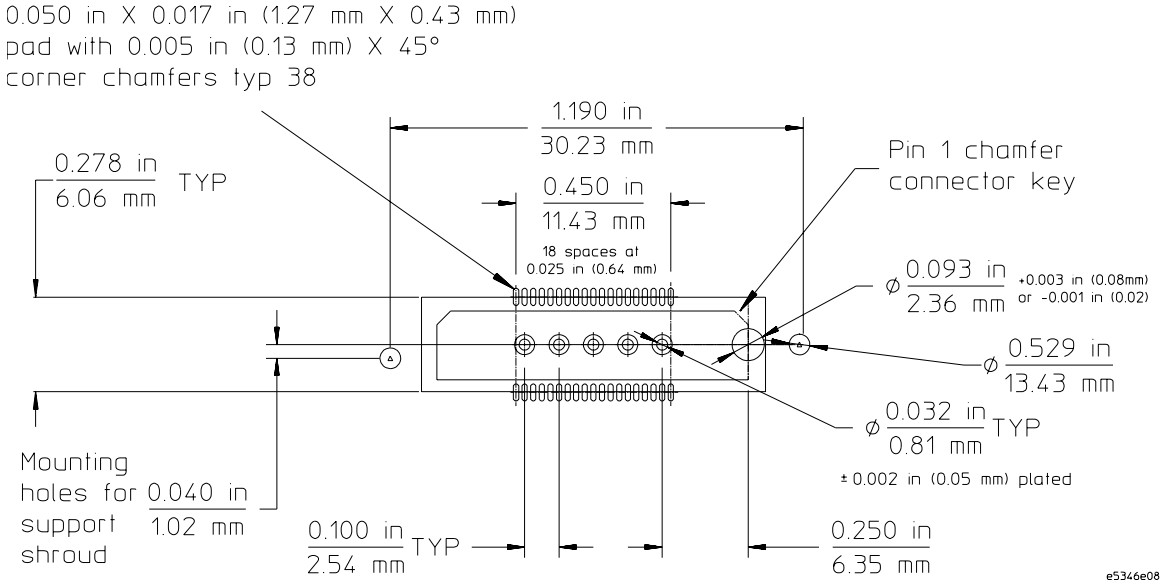
Mictor 38 Connector Design Rules



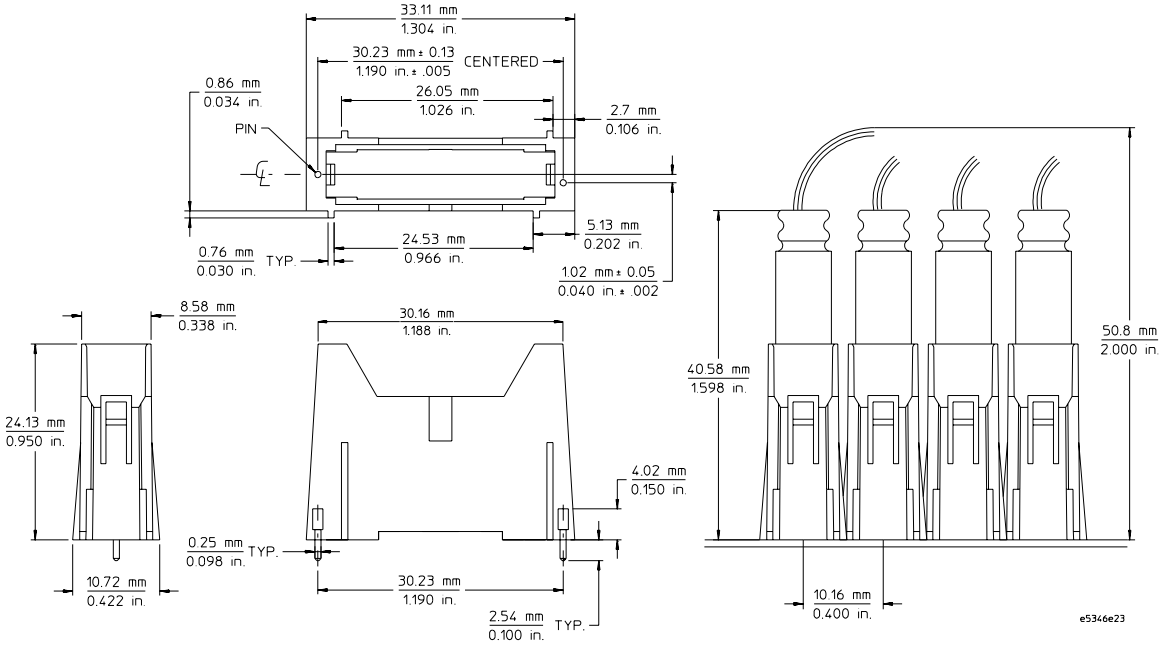
Support shroud

The support shroud (HP 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



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 HP 16600A and HP 16700A-series logic analysis systems

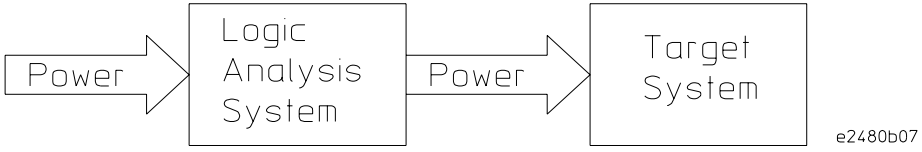
Ensure the target system is powered off.

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 on all other logic analyzers

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

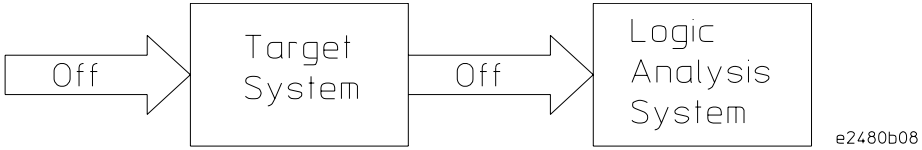
1. Logic analysis system.
2. Your target system.



To power off

Turn off power to your system in the following order:

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



Connecting the Analysis Probe to the Logic Analyzer

This section shows the connections between the logic analyzer pod cables and the cables on the analysis probe. The illustrations on the following pages show the analysis probe pod locations. There are three types of analysis available:

64-bit data

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

- HP 16715/16/17/18/19A logic analyzers - two cards (see page 52)
- HP 16715/16/17/18/19A logic analyzers - three cards (see page 53)
- HP 16710/11/12A logic analyzers - two cards (see page 55)
- HP 16600A logic analyzers (see page 56)
- HP 16601A logic analyzers (see page 57)
- HP 16550A logic analyzers- two cards (see page 61)
- HP 16554/55/56/57 logic analyzers - two cards (see page 63)
- HP 16554/55/56/57 logic analyzers - three cards (see page 64)

32-bit data

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

- HP 16550A logic analyzers - one card (see page 60)

No data

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

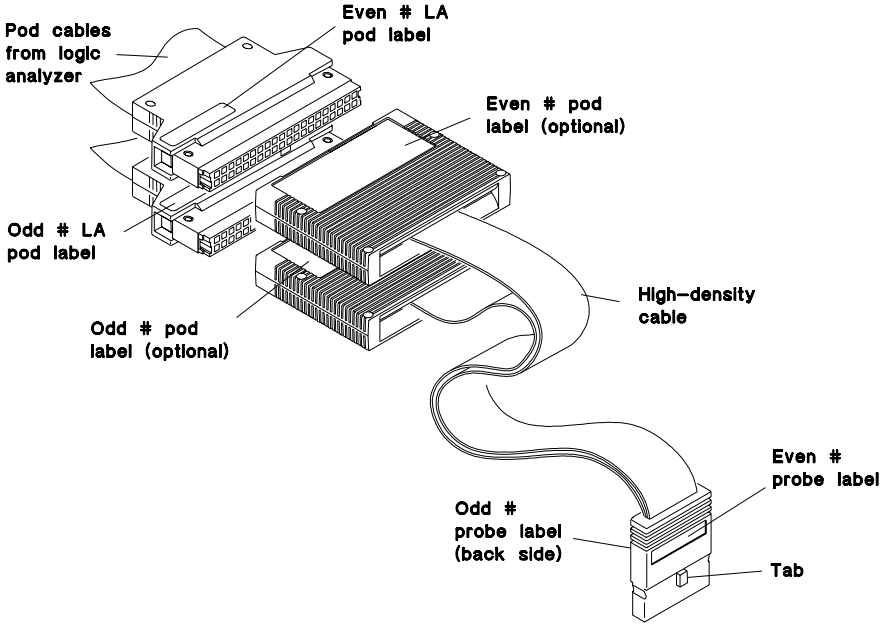
- HP 16715/16/17/18/19A logic analyzers - one card (see page 51)
- HP 16710/11/12A logic analyzers - one card (see page 54)
- HP 16554/55/56/57 logic analyzers - one card (see page 62)
- HP 16550A logic analyzers - one card (see page 60)

NOTE: To connect logic analyzers not listed here use the Setup Assistant, as described on page 23.

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

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

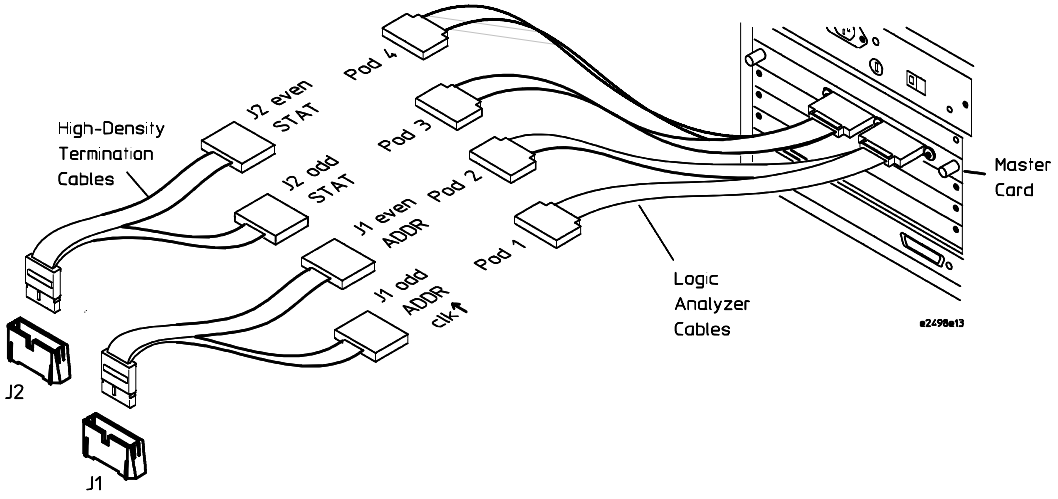
HP E5346A Cable Numbering



e2498e08

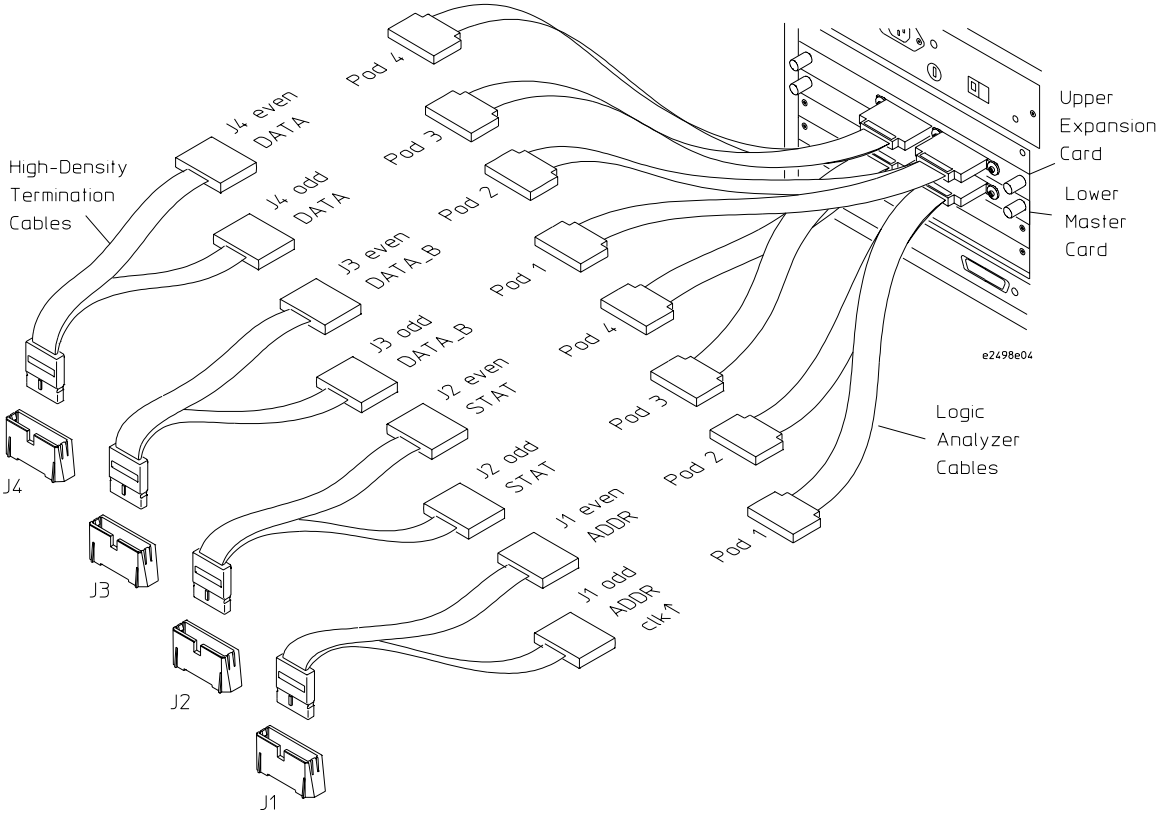
To connect to the HP 16715/16/17/18/19A logic analyzer (one card)

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



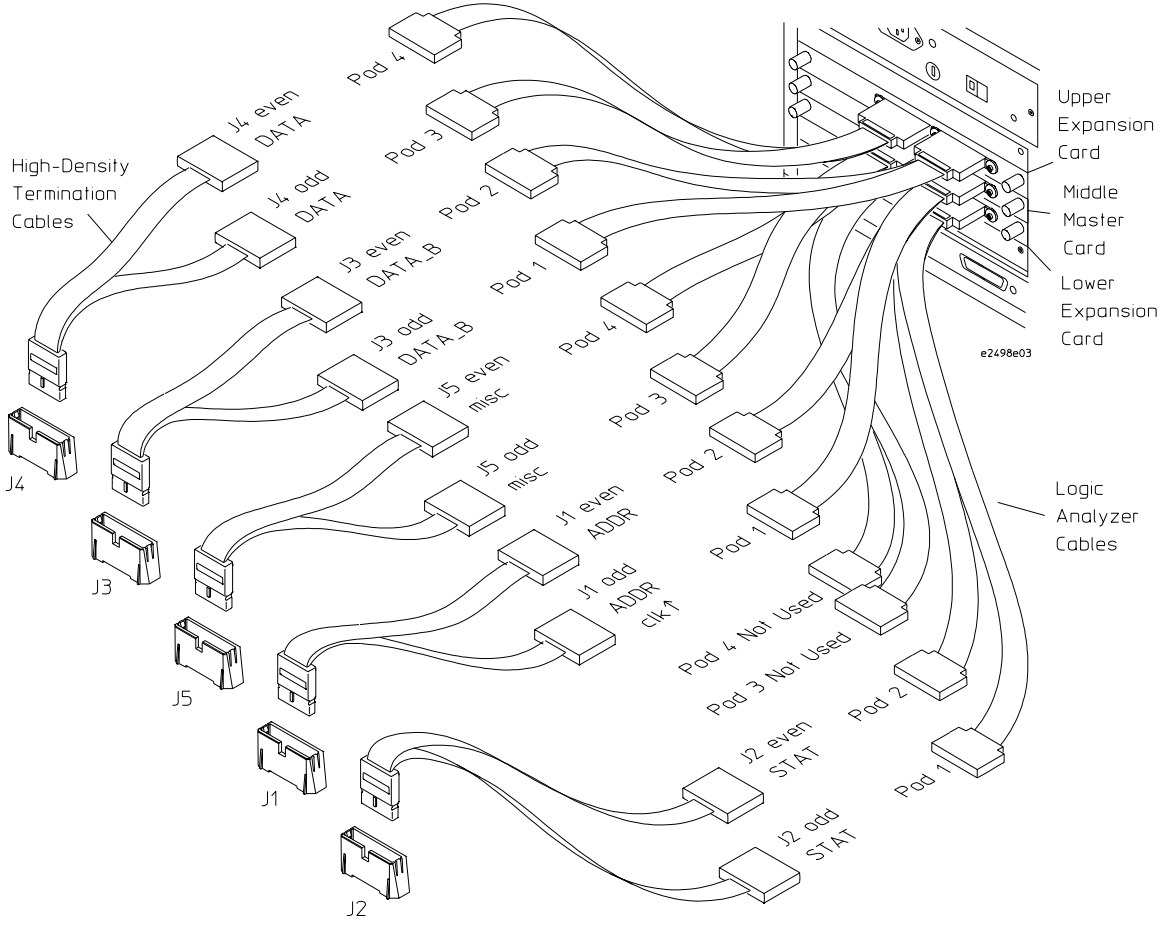
To connect to the HP 16715/16/17/18/19A logic analyzer (two cards)

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



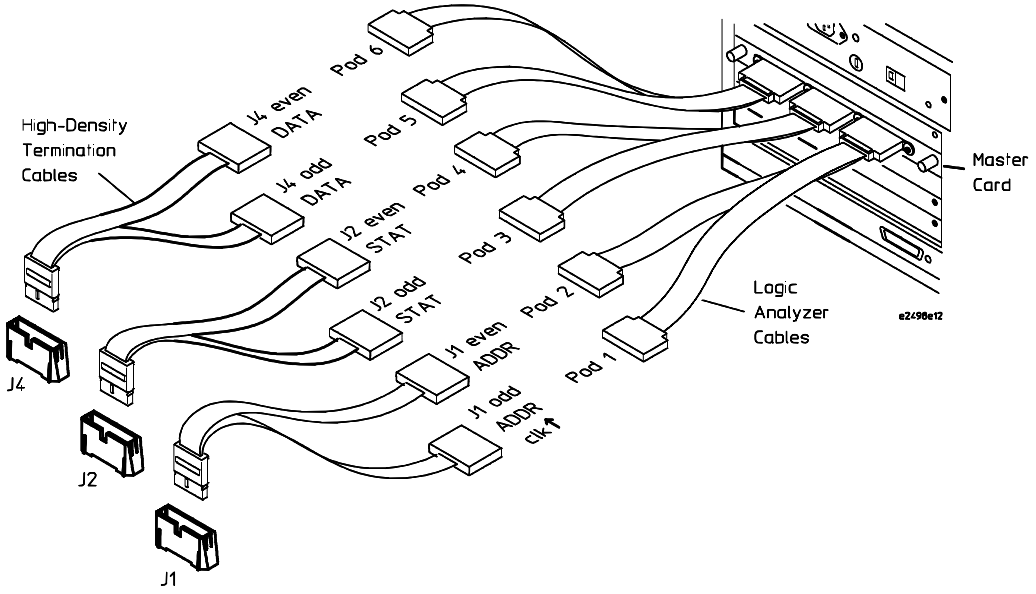
To connect to the HP 16715/16/17/18/19A logic analyzer (three cards)

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



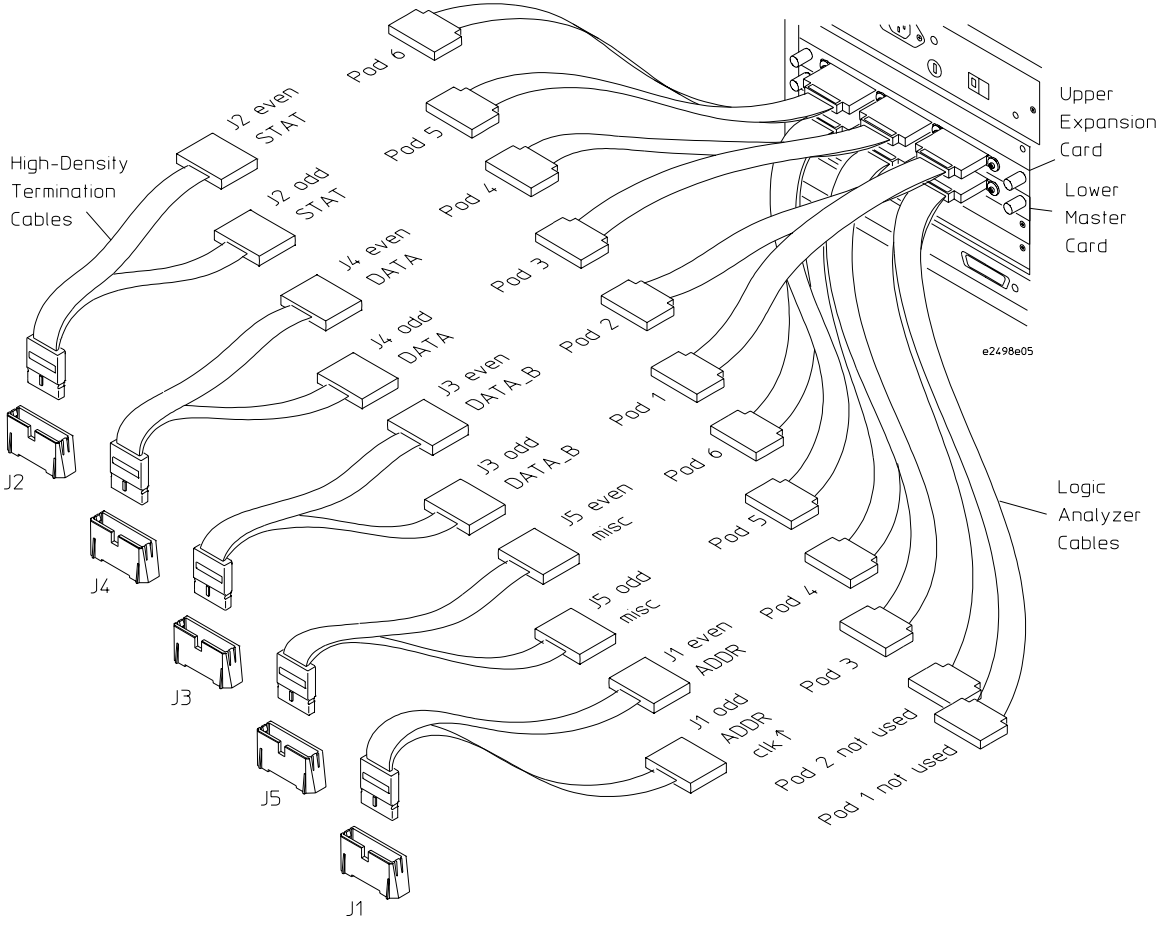
To connect to the HP 16710/11/12A logic analyzer (one card)

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



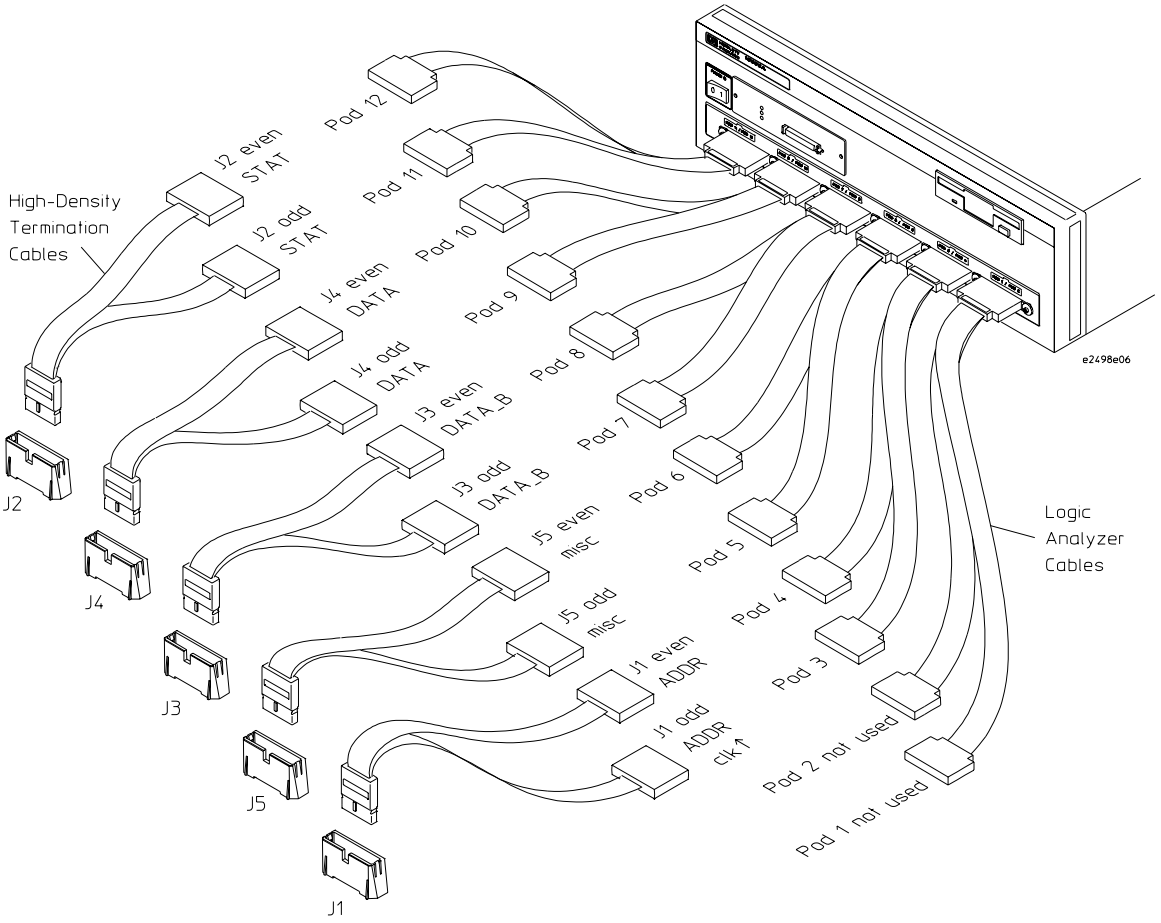
To connect to the HP 16710/11/12A logic analyzer (two cards)

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



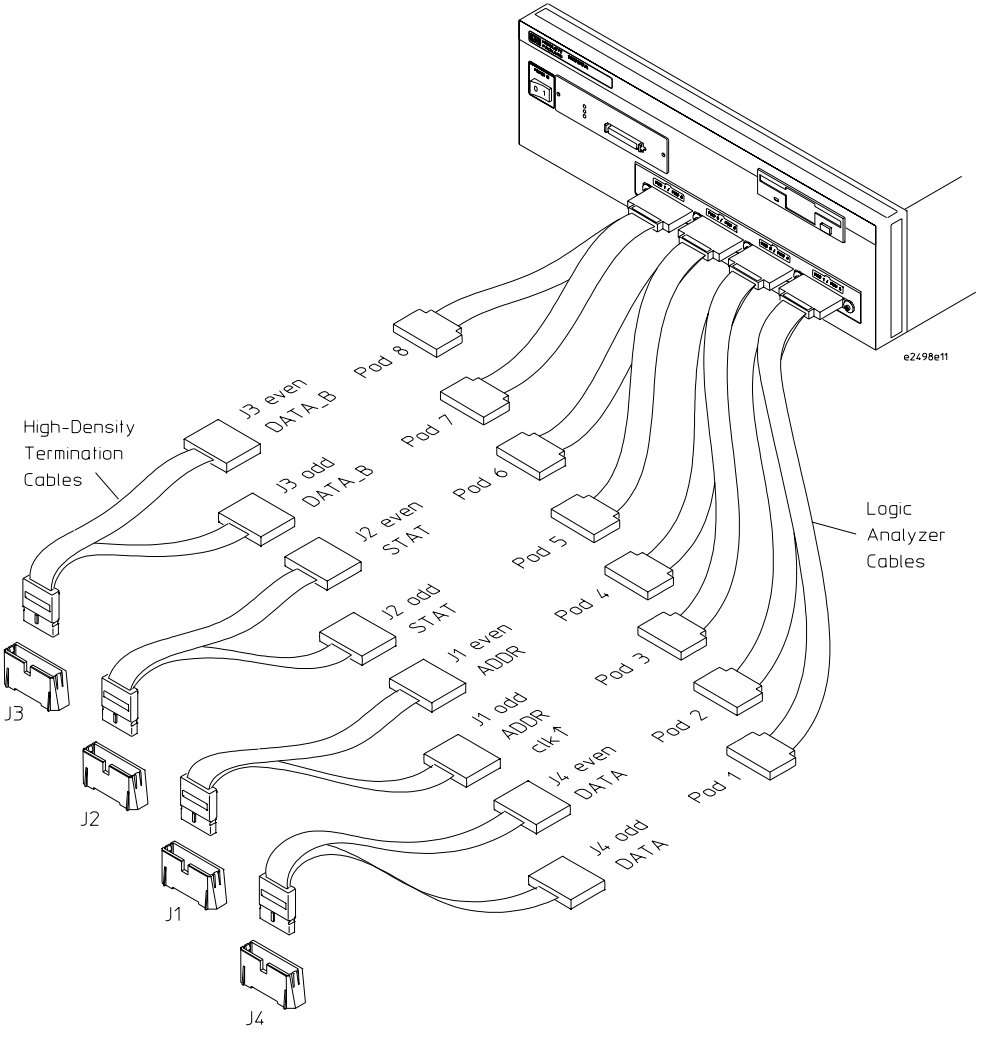
To connect to the HP 16600A logic analyzer

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



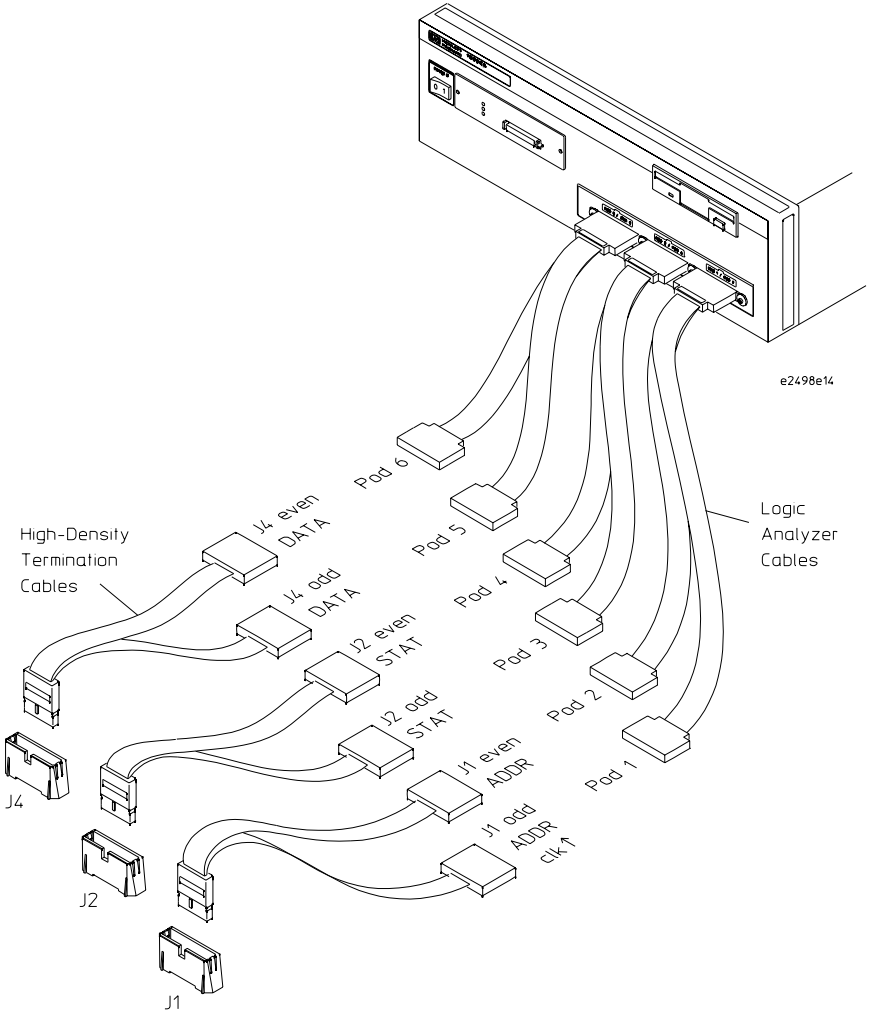
To connect to the HP 16601A logic analyzer

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



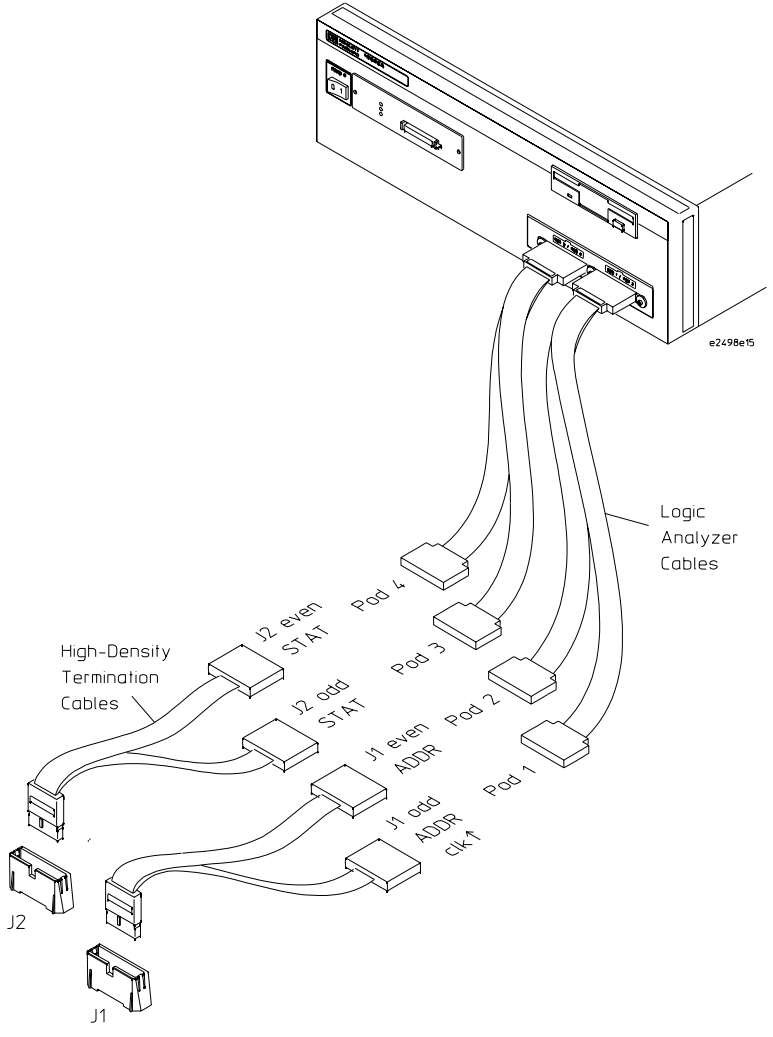
To connect to the HP 16602A logic analyzer

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



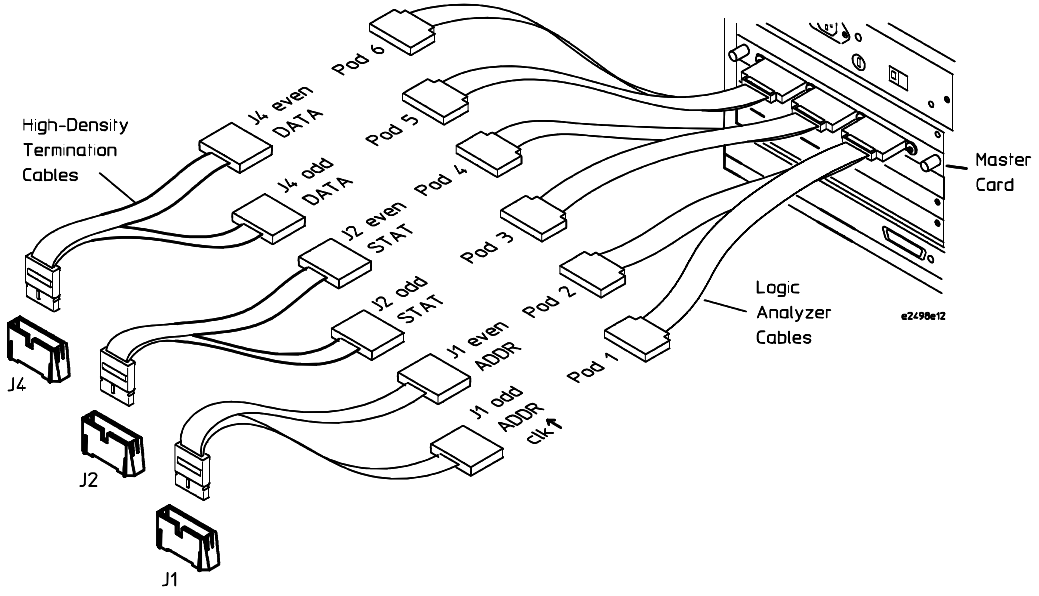
To connect to the HP 16603A logic analyzer

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



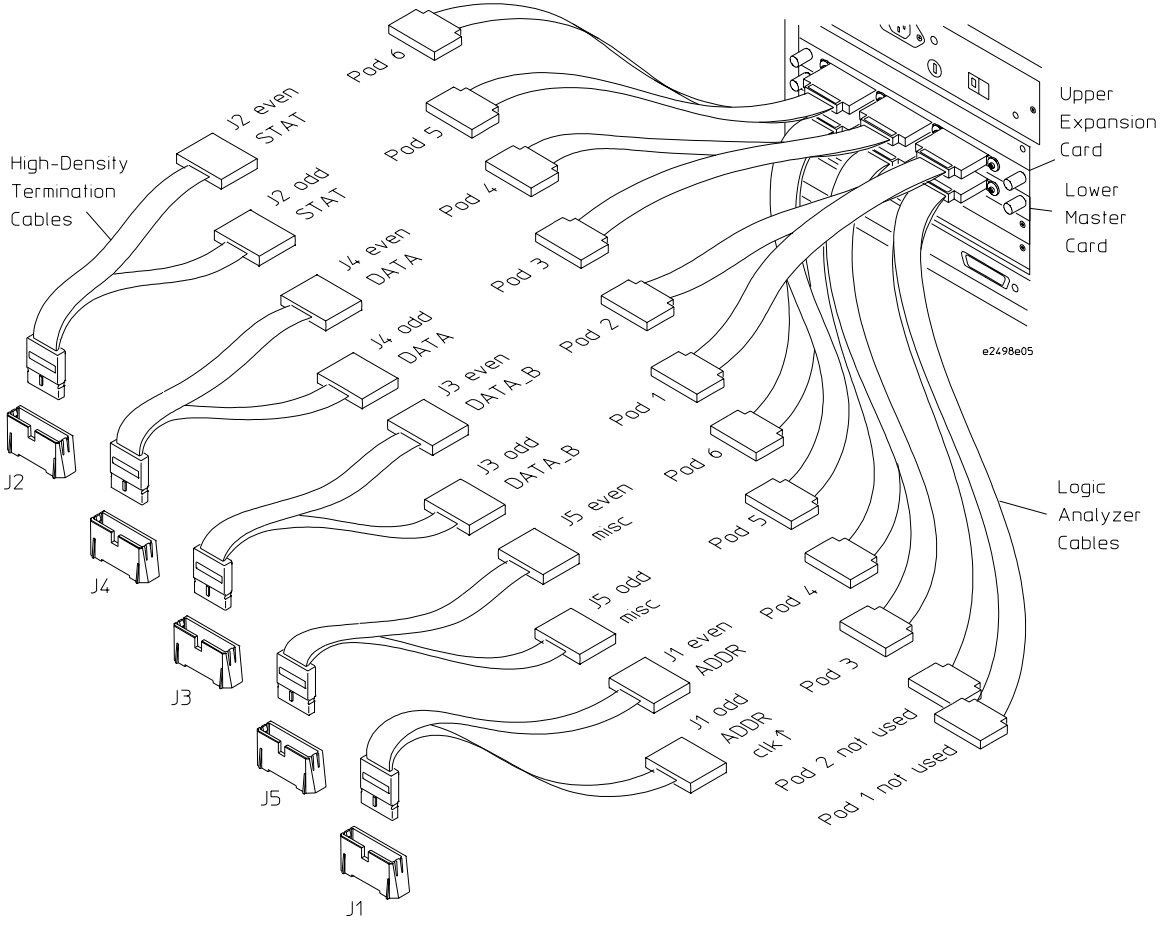
To connect to the HP 16550A analyzer (one card)

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



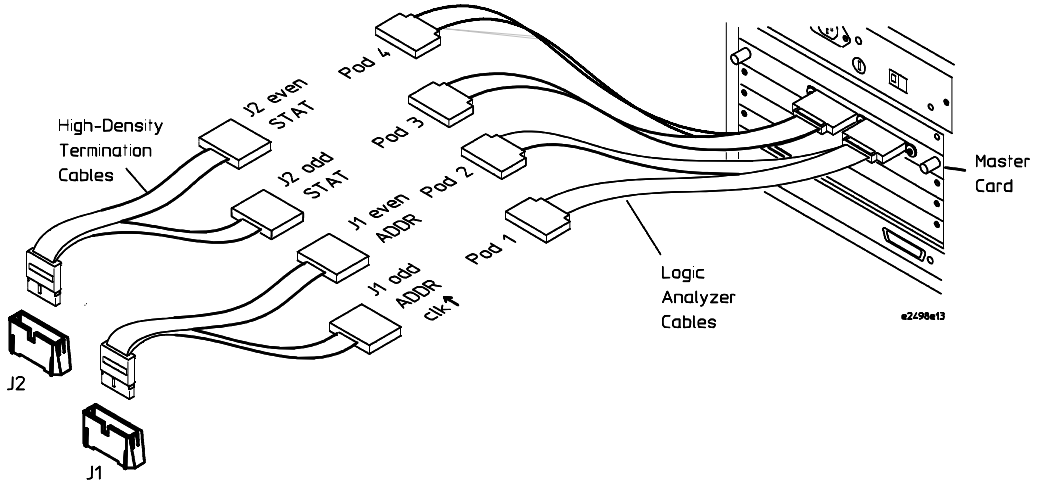
To connect to the HP 16550A analyzer (two cards)

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



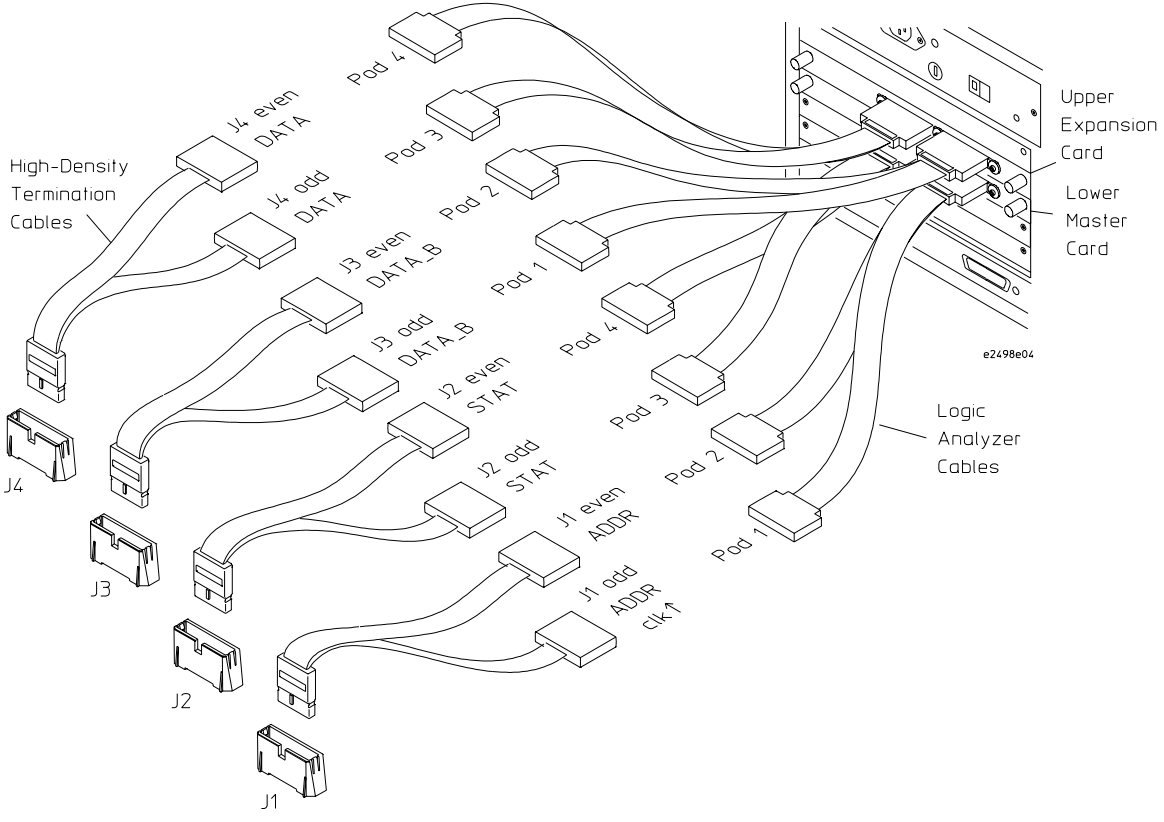
To connect to the HP 16554/55/56/57 (one-card)

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



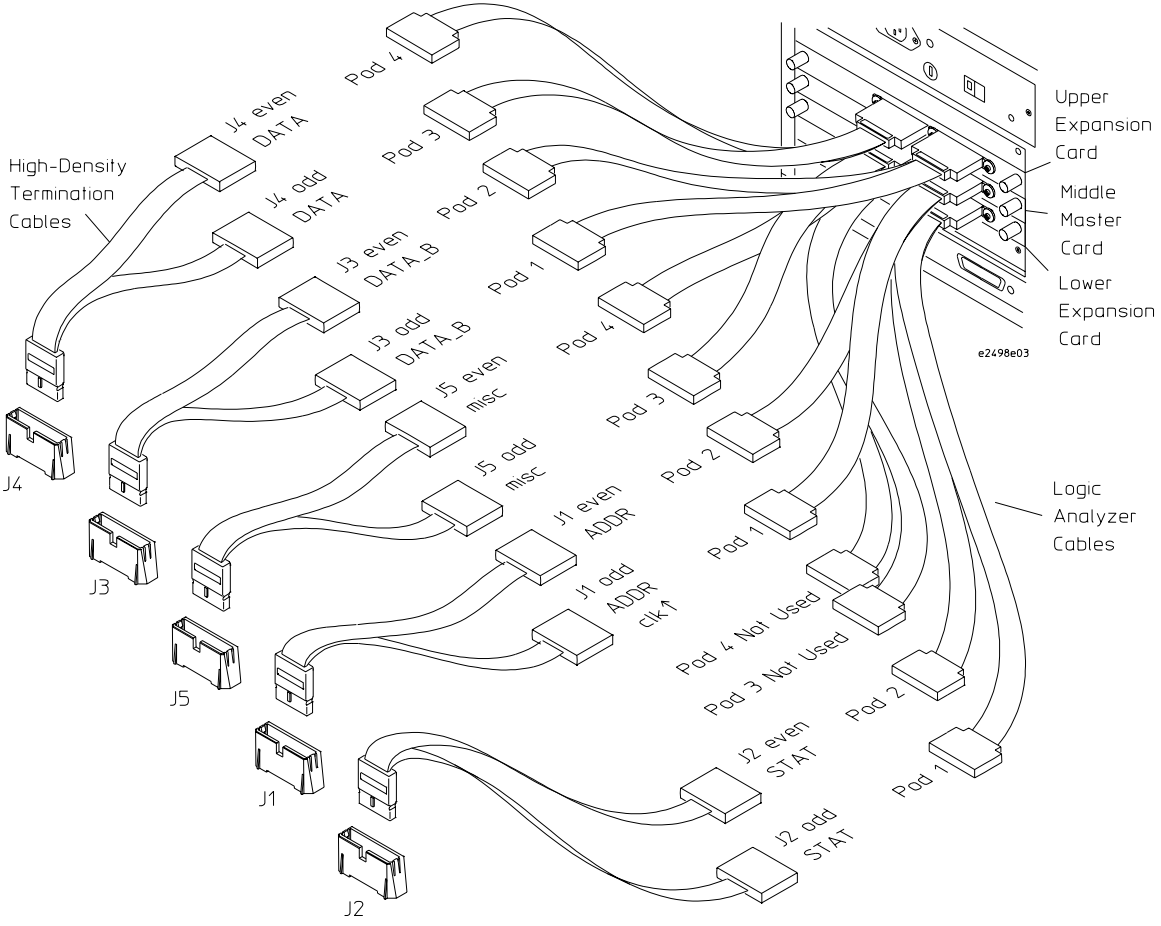
To connect to the HP 16554/55/56/57 (two-cards)

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



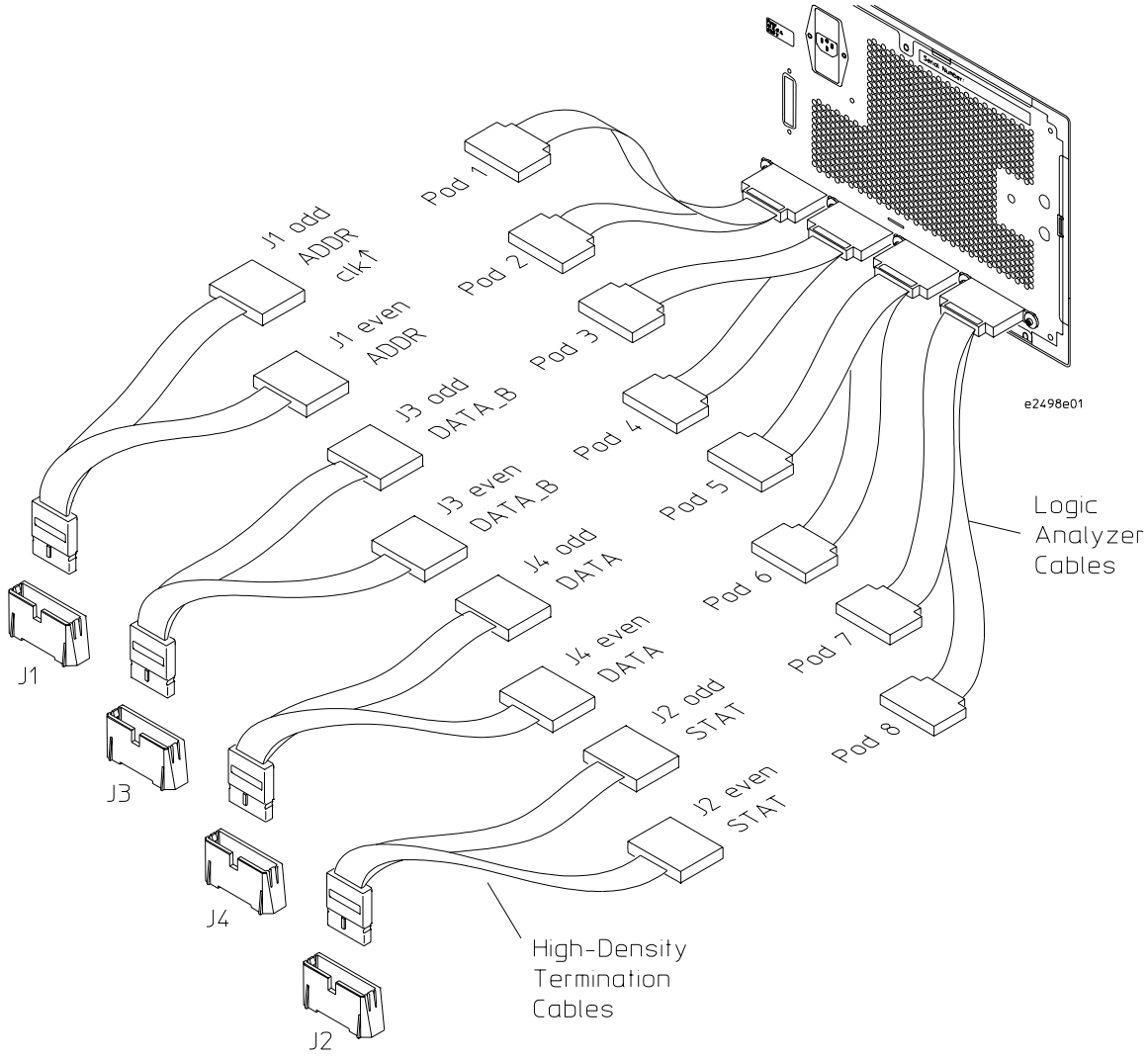
To connect to the HP 16554/55/56/57 (three-cards)

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



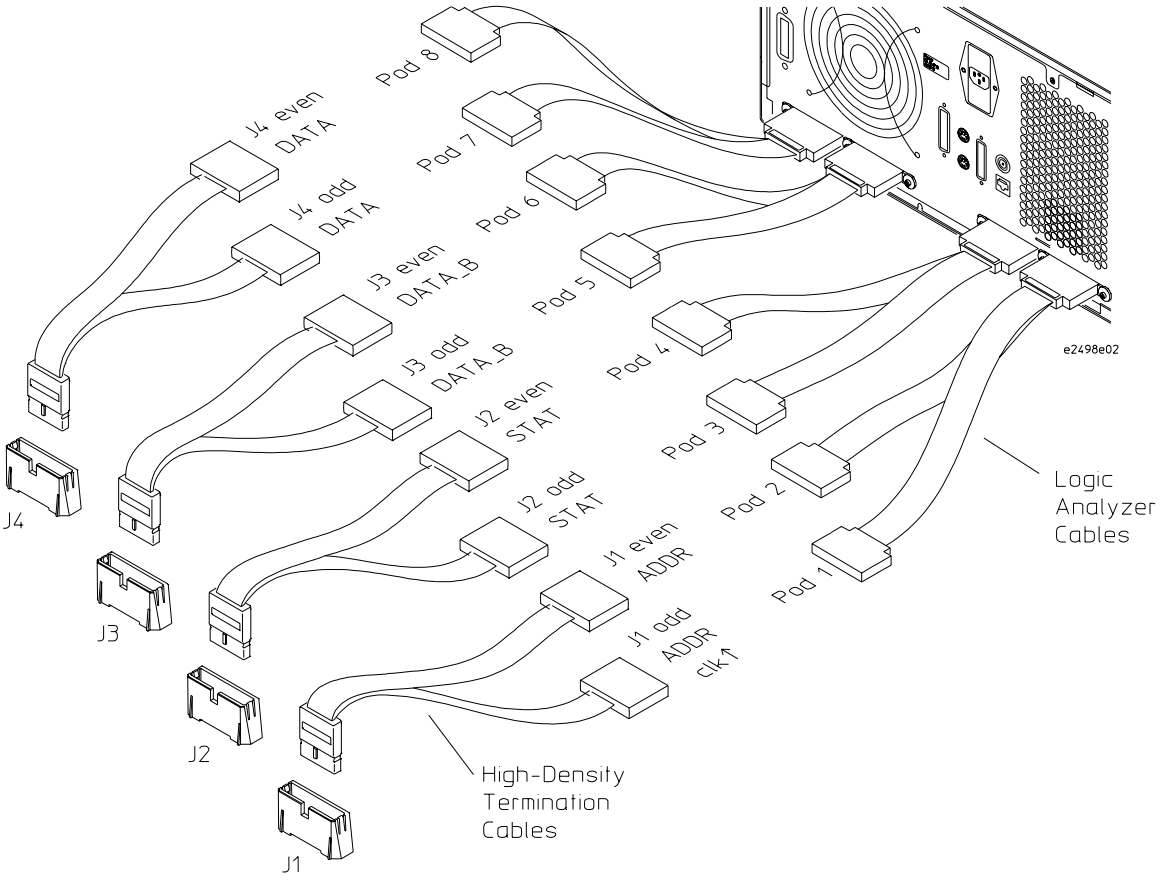
To connect to the HP 1660A/C logic analyzers

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



To connect to the HP 1670A/D logic analyzer

Use the figure below to connect the target system to the HP 1670A/D logic analyzers. Find the labels that were shipped with the high-density cables and use them to help identify the connections.



Configuring the Logic Analysis System

You configure the logic analyzer by loading a configuration file. The information in the configuration file includes:

- Label names and channel assignments for the logic analyzer
- Inverse assembler file name

The configuration file you use is determined by the logic analyzer you are using. The configuration file names are listed with the logic analyzer connection tables, and in a table at the end of this section.

The procedures for loading a configuration file depend on the type of logic analyzer you are using. There is one procedure for the HP 16600/700 series logic analysis systems, and another procedure for the HP 1660-series, HP 1670-series, and logic analyzer modules in an HP 16500B/C mainframe. Use the appropriate procedures for your analyzer.

To load configuration and inverse assembler files into HP 16600A/700A logic analysis systems from the system hard disk

If you did not use Setup Assistant, you can load the configuration and inverse assembler files from the logic analysis system hard disk.

1 Click on the File Manager icon. Use File Manager to ensure that the subdirectory /hplogic/configs/hp/ppc7xx/ exists.

> If the above directory does not exist, you need to install the POWERPC7XX Processor Support Package. Close File Manager, then use the procedure on the CD-ROM jacket to install the POWERPC7XX Processor Support Package before you continue.

2 Using File Manager, select the configuration file you want to load in the /hplogic/configs/hp/ppc7xx/ directory, then click 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 PowerPC 7XX analysis by loading the appropriate PowerPC 7XX configuration file. Loading the indicated file also automatically loads the correct inverse assembler.

3 Close File Manager.

To load an inverse assembler from the floppy disk (HP 16600A/700A)

The preferred method is to install this functionality from the CD-ROM onto the hard disk and load from the hard disk, also described in this manual.

To install a configuration and inverse assembler file from the HP E2498A floppy disk that was shipped with your HP inverse assembler product:

1. Install the floppy disk in the floppy drive on the HP 16600A/16700A-series logic analysis system mainframe.
2. In the Logic Analysis System window, click the File Manager icon.
3. In the File Manager window:
   - Set Current Disk to Flexible Disk.
   - Set Target to the analyzer you wish to configure.
   - Click the name of the desired configuration file in the Contents frame. The Contents frame lists the configuration files and inverse assembler files available on the floppy disk. These may be either DOS or LIF format files. Either format can be loaded directly into the appropriate logic analyzers.

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

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

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

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

To load configuration files—other logic analyzers

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

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

For logic analyzers that have a hard disk, you might want to create a directory such as PPC750 on the hard drive and copy the contents of the floppy onto the hard drive. You can then use the hard drive for loading files.

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

   Your configuration file choice depends on which analyzer you are using. See table on following page.
6. Execute the load operation to load the file into the logic analyzer.

   The logic analyzer is configured for PowerPC 7XX analysis by loading the appropriate PowerPC 7XX configuration file. Loading the indicated file also automatically loads the pipelined-systems inverse assembler. For information on the difference between the pipelined and nonpipelined inverse assemblers, refer to "To select a different inverse assembler."
7. If you are using the HP 16505A Prototype Analyzer, insert the appropriate "16505 Prototype Analyzer" flexible disk (pipelined or nonpipelined) into the disk drive of the prototype analyzer, and update the HP 16505A from the Session Manager. You must close your workspace to run the update.

   The HP 16505A Prototype Analyzer requires software version A.01.30 or higher to work with the HP E2498A.

Logic Analyzer Configuration Files

Analyzer Model	Analyzer Description (modules only)	Configuration File $V_{I/O}$ = 3.3V	Configuration File $V_{I/O}$ = 1.8V
16715/16/17/18/19A (one card)	167 MHz STATE 333/667 MHz TIMING	C7XXL0	C7XX_AVCL0
16715/16/17/18/19A (two card)	167 MHz STATE 333/667 MHz TIMING	C7XXL2	C7XX_AVCL2
16715/16/17/18/19A (three card)	167 MHz STATE 333/667 MHz TIMING	C7XXL3	C7XX_AVCL3
16710/11/12A (one card)	100 MHz STATE 250 MHz TIMING	C7XXF1	C7XX_AVCF1
16710/11/12A (two card)	100 MHz STATE 250 MHz TIMING	C7XXF2	C7XX_AVCF2
16600A	na	C7XXF2	C7XX_AVCF2
16601A	na	C7XXF2	C7XX_AVCF2
16602A	na	C7XXF1	C7XX_AVCF1
16603A	na	C7XXF0	C7XX_AVCF0
16550A (one card)	100 MHz STATE 500 MHz TIMING	C7XXF1	C7XX_AVCF1
16550A (two card)	100 MHz STATE 500 MHz TIMING	C7XXF2	C7XX_AVCF2
16554A (one card)	0.5M SAMPLE 70/250 MHz LA	C7XXM0	C7XX_AVCM0
16554A (two card)	0.5M SAMPLE 70/250 MHz LA	C7XXM2	C7XX_AVCM2
16554A (three card)	0.5M SAMPLE 70/250 MHz LA	C7XXM3	C7XX_AVCM3
16555A/D (one card)	1.0M SAMPLE 100/500 MHz LA	C7XXM0	C7XX_AVCM0
16555A/D (two card)	1.0M SAMPLE 110/500 MHz LA	C7XXM2	C7XX_AVCM2
16555A/D (three card)	1.0M SAMPLE 110/500 MHz LA	C7XXM3	C7XX_AVCM3
16556A/D (one card)	1.0M SAMPLE 100/400 MHz LA	C7XXM0	C7XX_AVCM0
16556A/D (two card)	1.0M SAMPLE 110/400 MHz LA	C7XXM2	C7XX_AVCM2

Analyzer Model	Analyzer Description (modules only)	Configuration File $V_{I/O}$ = 3.3V	Configuration File $V_{I/O}$ = 1.8V
16556A/D (three card)	1.0M SAMPLE 110/400 MHz LA	C7XXM3	C7XX_AVCM3
16557D (one card)	1.0M SAMPLE 135/500 MHz LA	C7XXM0	C7XX_AVCM0
16557D (two card)	1.0M SAMPLE 135/500 MHz LA	C7XXM2	C7XX_AVCM2
16557D (three card)	1.0M SAMPLE 135/500 MHz LA	C7XXM3	C7XX_AVCM3
1660A/AS/C/CS	na	C7XXJ1	na
1670A/D	na	C7XXM2	na

NOTE: Use the Setup Assistant for logic analyzer configuration when possible. See page 23 for instructions for using the Setup Assistant.

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

To select a different inverse assembler

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

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

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

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

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

To load a different inverse assembler file in the other logic analyzers, repeat steps 1 - 6 in "To load configuration files—other logic analyzers," but for step 5 select the inverse assembler.

4

Analyzing the PPC7XX with an HP 16600A/16700A-series Logic Analyzer

This chapter describes modes of operation for the HP E2455B analysis probe. It also describes data, symbol encodings, and information about the inverse assembler.

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

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

Modes of Operation

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

State-per-ack mode

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

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

State-per-clock mode

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

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

Timing mode

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

1. Click on the logic analyzer icon.
2. Select Setup... from the menu.
3. Click on the Sampling tab.
4. Select the Timing Mode button.

Modes of Analysis

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

Inverse assembly analysis

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

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

Cache-on trace reconstruction

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

For cache-on trace reconstruction, set the operating mode to State-per-ack and enable the branch trace mode by setting the MSR.BE bit 22.

Logic Analyzer Configuration

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

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

Format menu

This section describes the organization of PPC7XX signals in the logic analyzer's Format menu.

The configuration files contain predefined format specifications. These format specifications include all labels for monitoring the microprocessor. The tables on the following pages show the signals used in the STAT label and the predefined symbols set up by the configuration files.

The HP 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.

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

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

Status Encoding

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

The inverse assembler uses STAT bits TC0, TSIZ0...2, TT0...3, TBST, TA, AACK, ARTRY, DRTRY, ABB, TEA, and TS. The signal-to-connector tables in the "Hardware Reference" chapter list all the PPC7XX signals probed and their corresponding analyzer channels.

Status Bit Description

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

Status Bit	Description
TEA-	The memory system may assert TEA to indicate a transfer error, e.g. an unmapped part of the address space.
TT 0:3	The Transfer Type signals indicate the direction and purpose of a bus transaction.
Atomic(TT0)	Usually, TT0 asserted indicates an atomic (e.g., stwcx.) transfer.
R/-W (TT1)	TT1 is high for a read, low for a write.
Invldt (TT2)	Usually, the PowerPC 7XX asserts TT2 to indicate that the corresponding cache line should be invalidated by other processors.
A Only (TT3)	TT3 high indicates that there is data associated with the current address. TT3 low usually indicates an address-only transaction.
TC 0:1	The Transfer Code outputs provide further information about the current transfer. For a read, they indicate whether instructions or operands are being fetched.
TBST-	When asserted, TBST- indicates a four-beat burst transfer of eight words.
TSIZ 0:2	Indicates the size for the data transfer in conjunction with TBST-.
WT-	Write Through indicates a write-through transaction that should be pushed through local caches to shared memory.
CI-	Cache Inhibit indicates that the 7XX will not cache a read.
GBL-	Global indicates the transaction is global, i.e., should be snooped by all other caching devices on the bus.
SRESET-	A falling-edge input causes the 7XX to undergo a soft reset.
HRESET-	Input asserted causes the 7XX to undergo a hard reset.
CKSTP-	Input asserted causes the 7XX to undergo machine check processing.
INT-	Input indicates an external interrupt is pending.
CHECKSTOP	Output indicates that the 7XX has entered the machine check state, i.e., stopped.
QREQ-	The PowerPC 7XX asserts Quiescent Request to indicate that it wants to be, or is, in a snooze mode.

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

Predefined Logic Analyzer Symbols

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

Symbol Description

Label	Symbol	Encoding
acks	idle	1111
	ARTRY	xxx0
	DRTRY	0xxx
	TA AACK	x00x
	AACK	xx0x
	TA	x0xx
R/-W	rd	1
	wr	0
TSIZ	burst	xxx0
	8 byte	0001
	1 byte	0011
	2 byte	0101
	3 byte	0111
	4 byte	1001
	5?byte	1011
	6?byte	1101
	7?byte	1111
TT	Kill Block	0110
	Wr Graphics	1010
	Rd Graphics	1110
	Clean Block	0000
	Write	0001
	Wr/Kill	0011
	Read	0101
	Rd/Flush	0111
	Wr/Atomic Flush	1001
	Read Atomic	1101
	Rd/Flush Atomic	1111
	?Flush Block	0010
	?DSYNC	0100
	?eieio	1000
	?(reserved)	1011
	?TLB Invalidate	1100
STAT	inst fetch	xxxx xxxx xxxx xxx1 xxxx 1xxx xxxx 0xxx

The least significant bit of the TSIZ label is the TBST- signal. The symbols prefixed by "?" represent signal outputs defined by the PowerPC architecture but not asserted by the PowerPC 7XX. An instruction fetch is indicated by AACK asserted (address & qualifiers valid), R/-W (TT1) asserted for read, and TC0 asserted.

Trigger dialog

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

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

Qualifying Stored Data

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

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

Configuring for State-per-clock mode

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

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

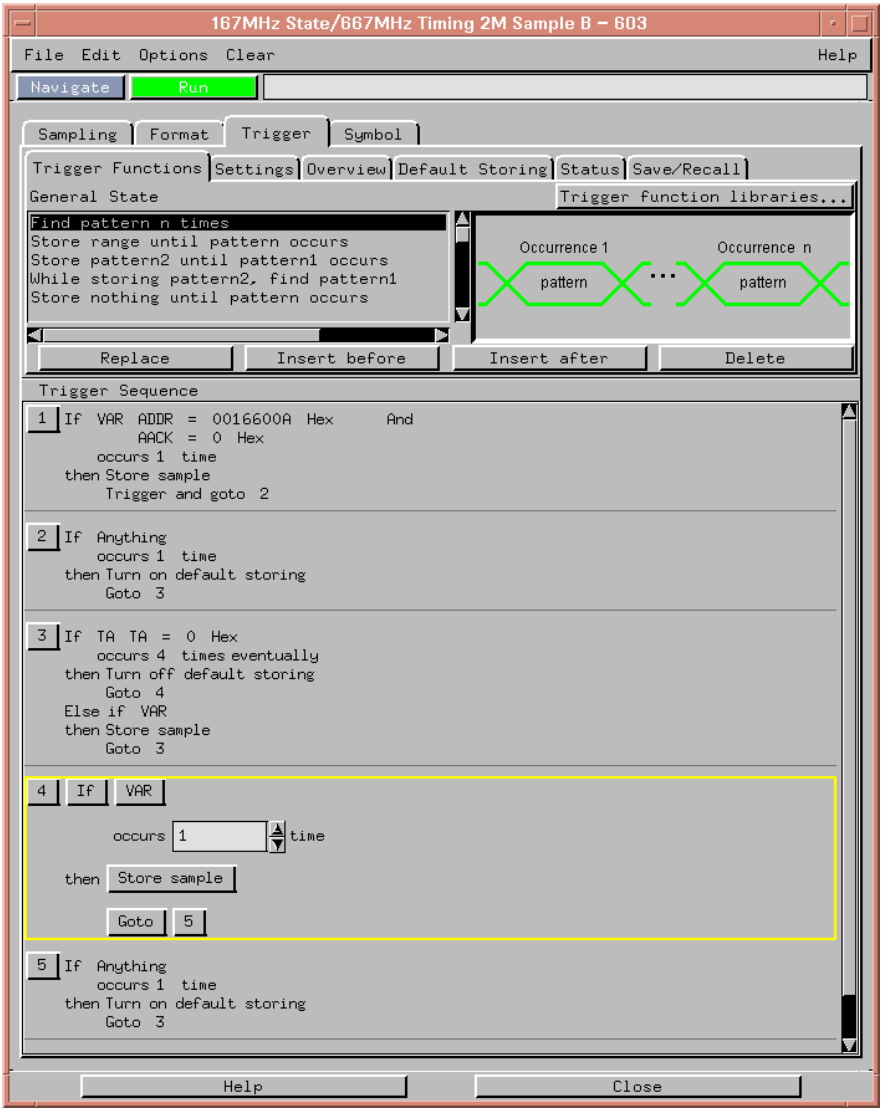
Capturing an address

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

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

Capturing Isolated Addresses

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



Using the Inverse Assembler

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

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

Using Cache-On Trace Reconstruction

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

The branch exception is located at 0x00000D00 for an exception prefix MSR.IP=0 or 0xFFF00D00 for an exception prefix MSR.IP=1. The interrupt routine writes the branch target address SRR0 to the tracking address (location in RAM which is non-cached or write-through mode is enabled for that memory block) so that the IA can track the program flow. Also, the tracking address must be on a word boundary.

example branch exception routine:

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

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

Customers who wish to nest interrupts must save and restore the SRR0 special purpose register before writing it out to the tracking address. Also, the customer must write out the exception address at the beginning of the exception.

example program exception routine:

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

To enable cache-on trace reconstruction:

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

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

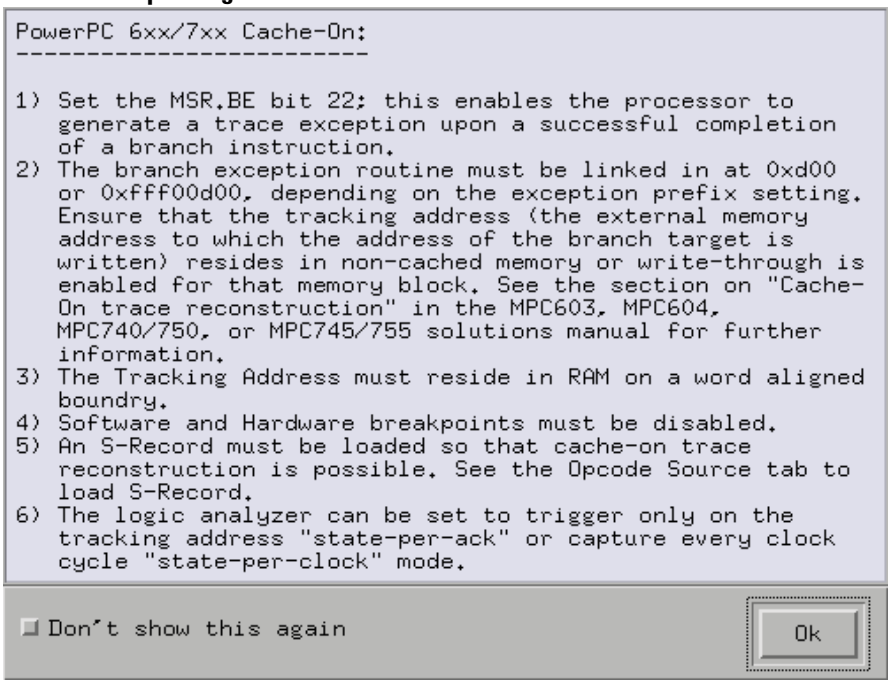
   In the Opcode Source tab
4. Load an S-Record executable file

Minimizing effects of cache-on trace on system performance

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

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

Cache-on help dialog



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

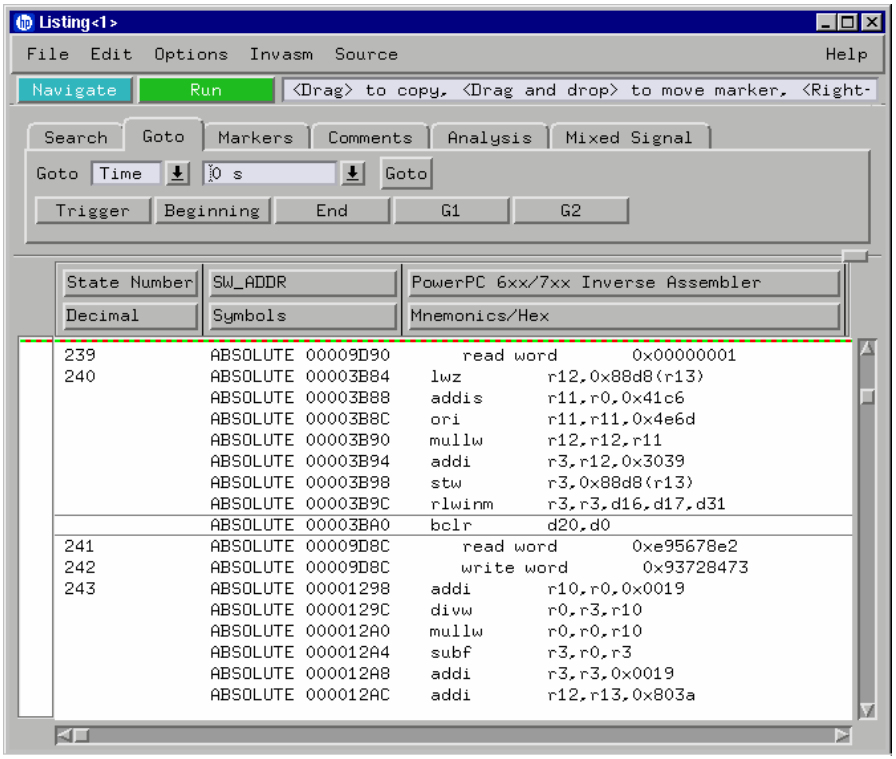
NOTE: When using HP run-control the user must issue an "rstssm" (reset soft stop mode) to disable soft stop mode. Also, this command must be issued whenever registers are modified, since soft stop mode is re-enabled automatically.

Enabling branch exception disassembly

The following trace shows cache-on execution using branch trace exception disassembly. See page 87 for an explanation of this feature.

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

Cache-on trace, S-Record executable file loaded, data bus connected, tracking address 0x00000100



Inverse Assembler Modes of Operation

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

Inverse Assembler Modes of Operation

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

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

To use the Invasm menu

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

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

Loading the Inverse Assembler

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

Setting the Inverse Assembler Preferences

Why the configuration is necessary

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

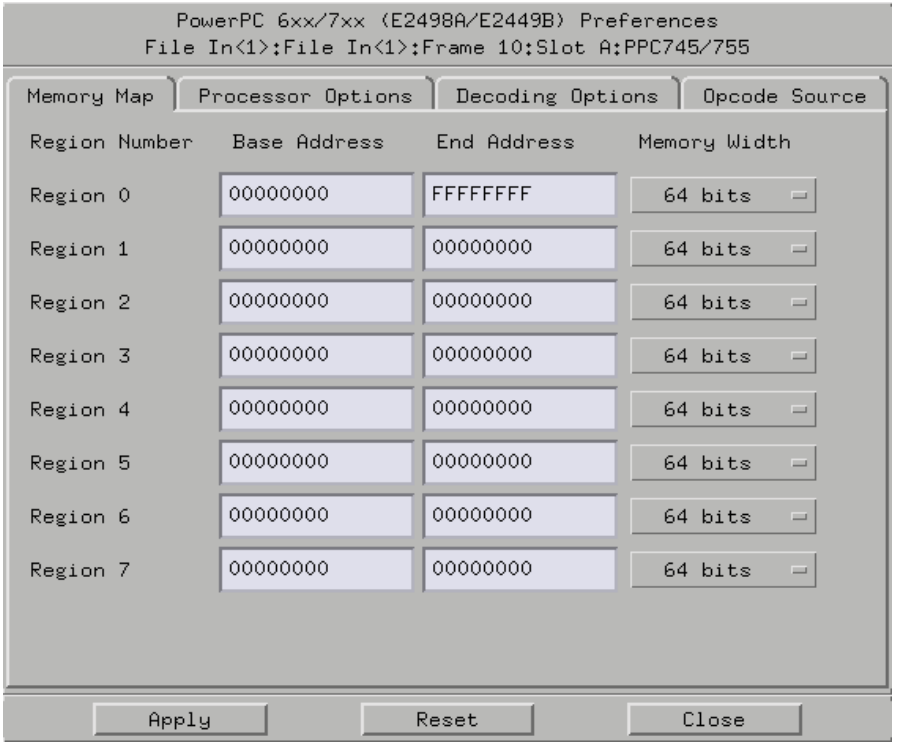
The memory controller operates by mapping every address to one of eight memory regions. Each memory region can be set up to drive different external signals, to have different write permissions, etc. The memory regions are numbered from 0 to 7. Memory region 0 has the highest priority and region 7 has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

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

To set the memory map preferences

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

Inverse Assembler Preferences Dialog



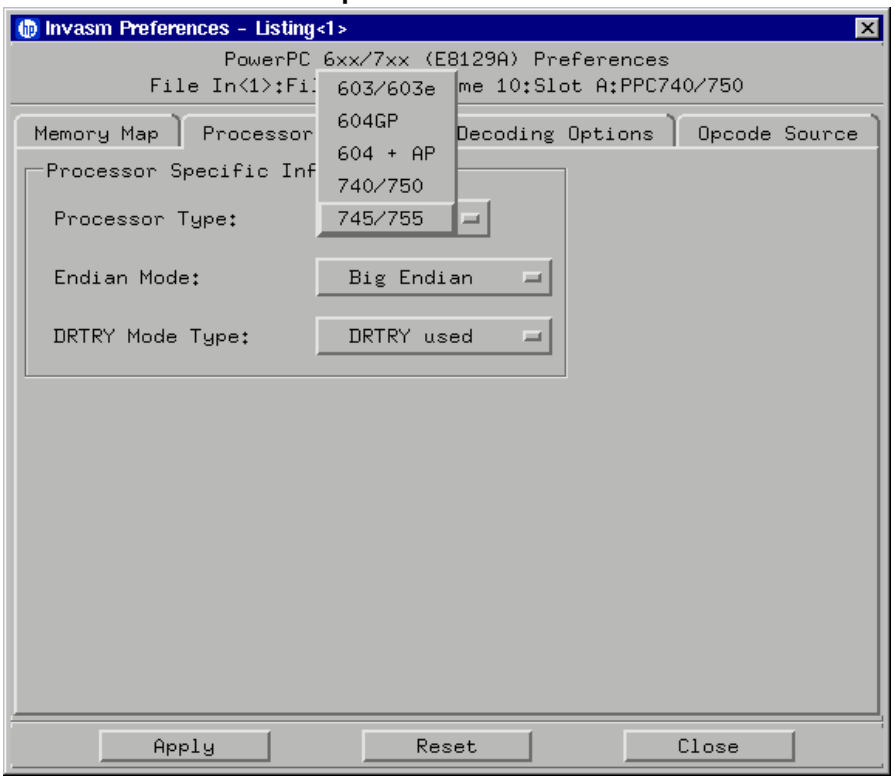
Click Apply when you have finished configuring the memory map.

To set the processor options preferences

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

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

Inverse Assembler Processor Options



Endian Mode

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

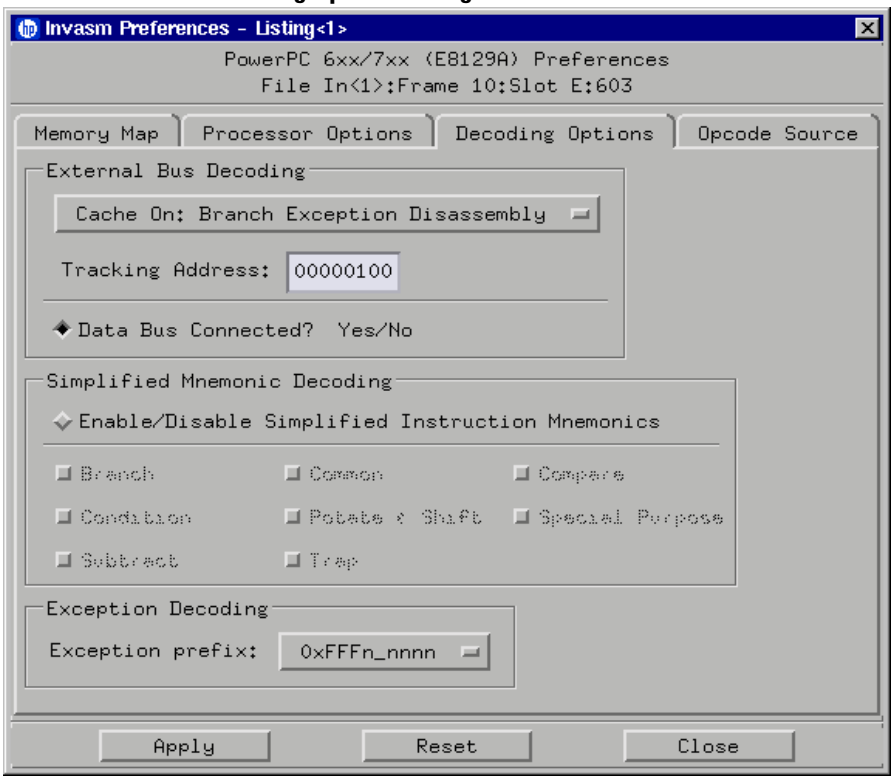
Little endian mode causes the instruction word from DL0...31 (DATA_B label; external address xxx4) to be dispatched before the instruction word from DH0...31 (DATA label; external address xxx0). It also causes byte and half-word reads and writes to appear on the opposite side of the bus and swaps the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

DRTRY Mode

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

To set the decoding options preferences

Inverse Assembler Decoding Options Dialog



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

Data Bus Connected. Read and write states are always indicated regardless of whether the data bus is connected. However, when the data bus is connected, read/write data will also be displayed. See “Inverse Assembler Modes of Operation” on page 91

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

- Conditional traps and branches decode the condition mnemonically when possible. For some conditions which have no conventional mnemonics (for example, "signed less than or unsigned greater than"), the condition field is displayed in binary.
- The L bit is omitted as a compare operand. Instead, compares are decoded as "cmpw" (or "?cmpd").
- "Add immediate" instructions with a negative immediate operand are decoded as subtract immediate ("subi").
- "Subtract from" instructions subf and subfc are decoded as subtract instructions sub and subc with the operands exchanged so that "sub r3 r4 r5" is mnemonically interpreted as "r3 = r4 - r5."
- ori r0 r0 0000 is decoded as "nop".
- Add immediate and add immediate shifted instructions, addi and addis, with a null source register are decoded as load immediate and load immediate shifted, li and lis.
- or instructions with identical source registers are decoded as move register, mr.
- nor instructions with identical source registers are decoded as not register, not.
- xor and eqv instructions with identical source and destination registers are decoded as clear and set, clr and set, respectively.
- The cror, crnor, crxor, and creqv instructions map analogously to crmv, crnot, crclr, and crset.
- When the mtcrf instruction field mask specifies the entire cr, it is decoded as mtcr.

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

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

is to shift right word immediate, e.g.

```
srwi    r30  r30  16.
```

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

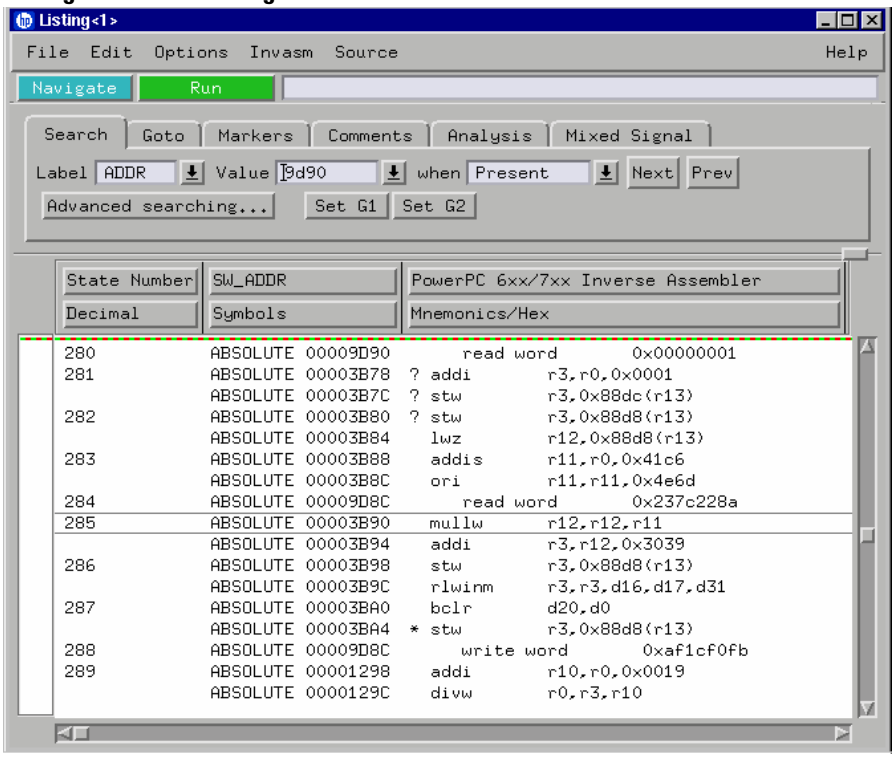
Mnemonic	Decoded As
rlwimi (rotate left word immediate then mask insert)	inslwi insert from left immediate insrwi insert from right immediate
rlwinm (rotate left word immediate then AND with mask)	rotlwirotate left immediate rotrwirotate right immediate slwishift left immediate srwishift right immediate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

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

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

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

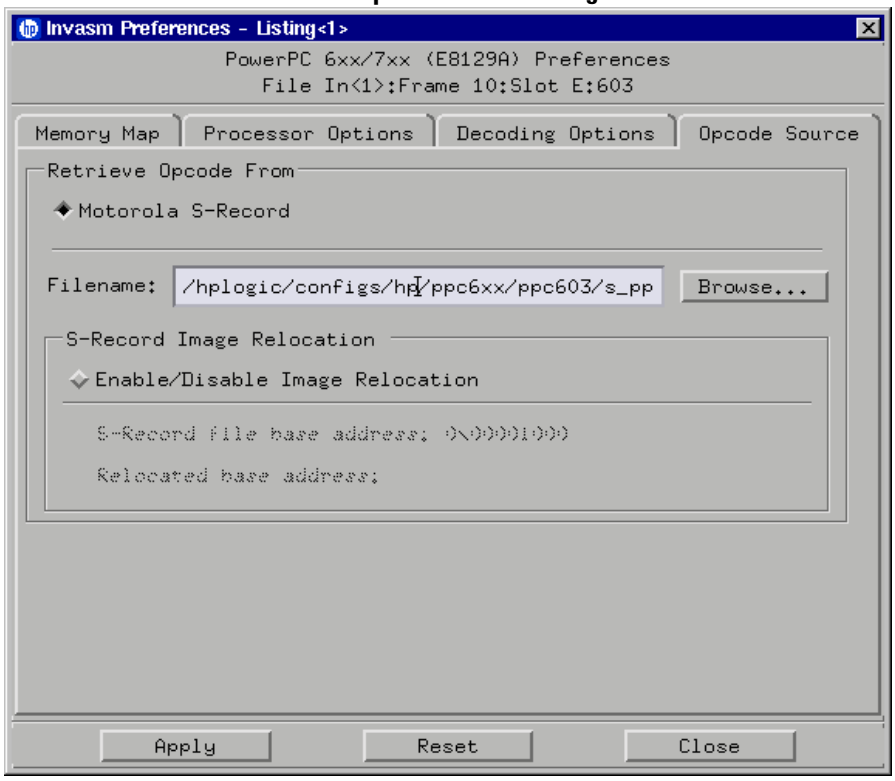
Listing Window Showing Trace with Data Bus Connected: Cache Off



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

To set the opcode source preferences

Inverse Assembler Preferences Opcode Source Dialog



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

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

Display Filtering

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

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

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

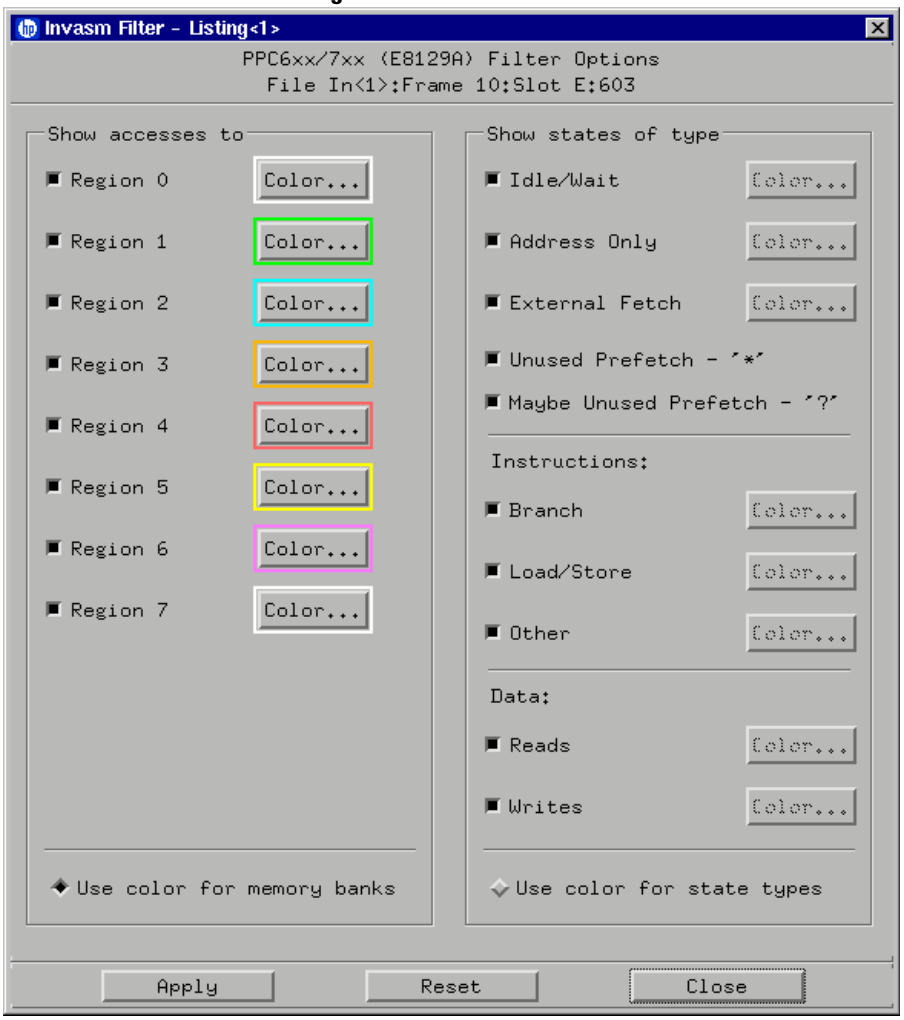
Second, you can isolate particular operations by suppressing all other operations. For example, you can show branches, with all other states suppressed, allowing quick analysis of branch instructions.

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

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

The following figure shows the inverse assembler filter dialog.

Inverse Assembler Filter Dialog



Options

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

To enable/disable the instruction cache on the PPC7XX

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

To disable the cache with the emulation module:

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

Register values for controlling the cache

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

To disable the cache with code:

- Disable the instruction cache with the following code:

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

- To also disable the data cache use:

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

- To invalidate and disable both caches use:

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

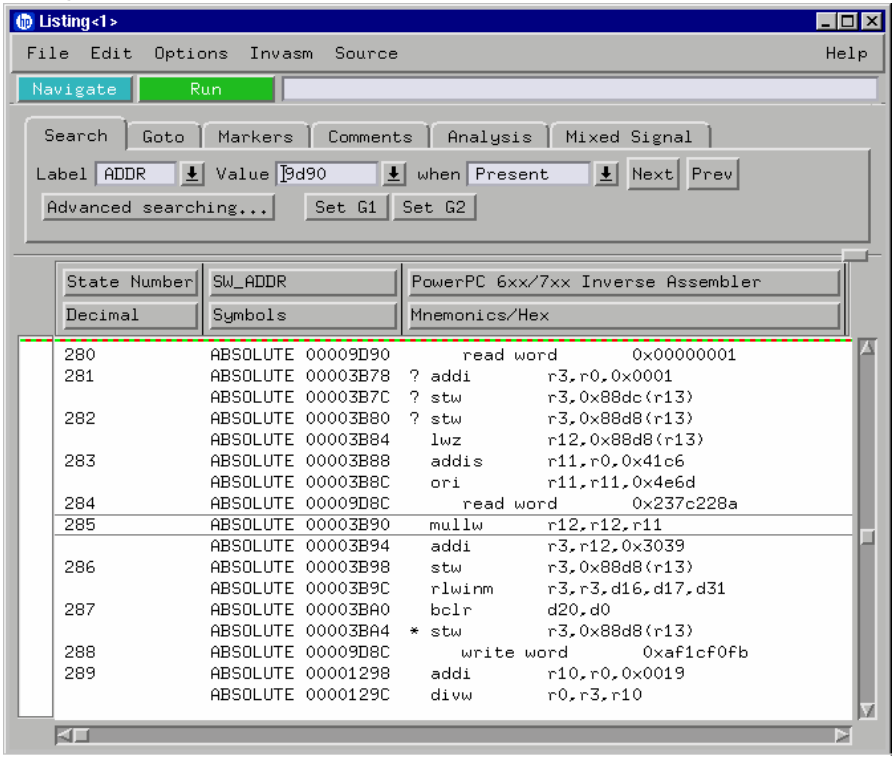
- Enable the instruction cache with the following code:

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

To display captured state data

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

Listing Menu.



On the HP 16600A/16700A-series logic analysis systems, the entire synthesized address appears under the label “SW_ADDR”. The actual address bits presented by the PowerPC 7XX may be observed under the ADDR label.

Inverse assembler output format

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

Interpreting Data

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

Most numerical data is displayed in hexadecimal, for example, “stwu r1,0xfff8(r1).” Bit numbers and shift counts are displayed in decimal with a dot suffix, for example, “cror 31. 31. 31.”

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

The inverse assembler decodes the full PowerPC instruction set architecture, including 64-bit mode instructions and optional instructions not implemented on the PowerPC 7XX. When these unimplemented opcodes are encountered, the listing displays “illegal opcode.”

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

Branch Instructions

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

SW_ADDR Label

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

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

Overfetch Marking

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

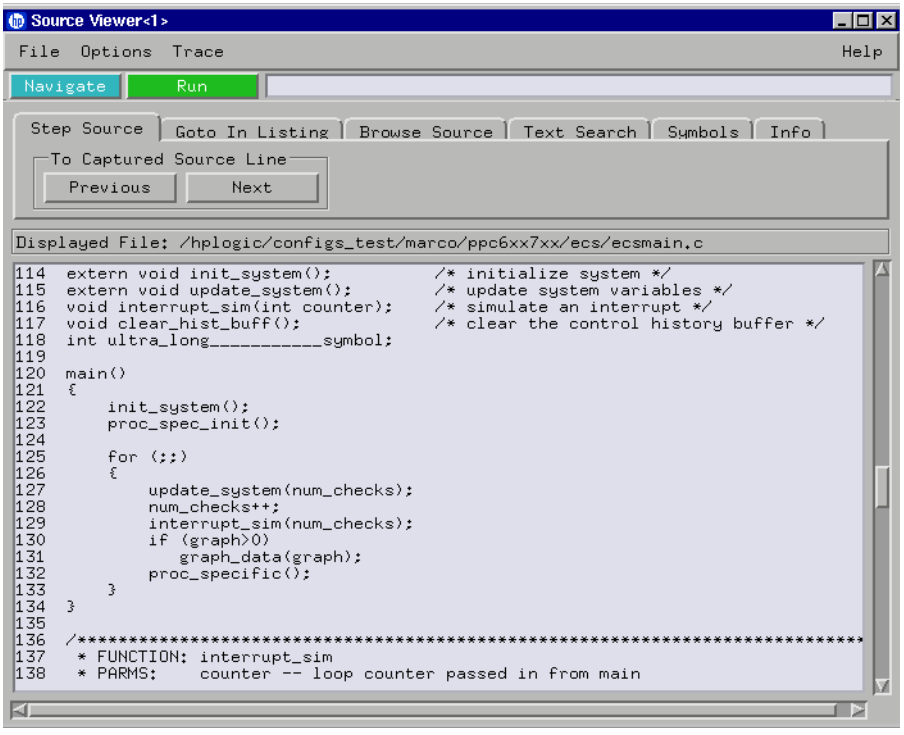
- When the 7XX executes a branch instruction, the instructions between the branch and the branch target are not executed. These instructions are indicated with an asterisk "*", or if the bus trace is ambiguous, with an interrogation point "?". If the instruction cache is enabled, the branch target may already be in the cache and will not be fetched over the bus. The remaining cache line containing the branch will be marked as overfetch.

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

Displaying Data with the HP B4620B Source Correlation Tool Set

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

Source Correlation Tool Set Data



5

Analyzing the PPC7XX with an HP 1660A/1670A/16500B/C-series Logic Analyzer

This chapter describes modes of operation for the HP E2455B analysis probe. It also describes data, symbol encodings, and information about the inverse assembler.

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

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

Modes of Operation

The HP E2455B analysis probe can be used in three different analysis modes: State-per-ack, State-per-clock, or Timing. The following sections describe these modes and how to configure the logic analyzer for each mode.

State-per-ack mode

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

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

State-per-clock mode

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

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

Timing mode

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

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

Logic Analyzer Configuration

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

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

Format menu

This section describes the organization of PPC7XX signals in the logic analyzer's Format menu.

The configuration files contain predefined format specifications. These format specifications include all labels for monitoring the microprocessor. The tables on the following pages show the signals used in the STAT label and the predefined symbols set up by the configuration files.

The HP 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.

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

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

Status Encoding

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

The inverse assembler uses STAT bits TC0, TSIZ0...2, TT0...3, TBST, TA, AACK, ARTRY, DRTRY, ABB, TEA, and TS. The signal-to-connector tables in the "Hardware Reference" chapter list all the PPC7XX signals probed and their corresponding analyzer channels.

Status Bit Description

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

Status Bit	Description
TEA-	The memory system may assert TEA to indicate a transfer error, e.g. an unmapped part of the address space.
TT 0:3	The Transfer Type signals indicate the direction and purpose of a bus transaction.
Atomic(TT0)	Usually, TT0 asserted indicates an atomic (e.g., stwcx.) transfer.
R/-W (TT1)	TT1 is high for a read, low for a write.
Invldt (TT2)	Usually, the PowerPC 7XX asserts TT2 to indicate that the corresponding cache line should be invalidated by other processors.
A Only (TT3)	TT3 high indicates that there is data associated with the current address. TT3 low usually indicates an address-only transaction.
TC 0:1	The Transfer Code outputs provide further information about the current transfer. For a read, they indicate whether instructions or operands are being fetched.
TBST-	When asserted, TBST- indicates a four-beat burst transfer of eight words.
TSIZ 0:2	Indicates the size for the data transfer in conjunction with TBST-.
WT-	Write Through indicates a write-through transaction that should be pushed through local caches to shared memory.
CI-	Cache Inhibit indicates that the 7XX will not cache a read.
GBL-	Global indicates the transaction is global, i.e., should be snooped by all other caching devices on the bus.
SRESET-	A falling-edge input causes the 7XX to undergo a soft reset.
HRESET-	Input asserted causes the 7XX to undergo a hard reset.
CKSTP-	Input asserted causes the 7XX to undergo machine check processing.
INT-	Input indicates an external interrupt is pending.
CHECKSTOP	Output indicates that the 7XX has entered the machine check state, i.e., stopped.
QREQ-	The PowerPC 7XX asserts Quiescent Request to indicate that it wants to be, or is, in a snooze mode.

Predefined Logic Analyzer Symbols

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

Symbol Description

Label	Symbol	Encoding
acks	idle	1111
	ARTRY	xxx0
	DRTRY	0xxx
	TA AACK	x00x
	AACK	xx0x
	TA	x0xx
R/-W	rd	1
	wr	0
TSIZ	burst	xxx0
	8 byte	0001
	1 byte	0011
	2 byte	0101
	3 byte	0111
	4 byte	1001
	5?byte	1011
	6?byte	1101
	7?byte	1111
TT	Kill Block	0110
	Wr Graphics	1010
	Rd Graphics	1110
	Clean Block	0000
	Write	0001
	Wr/Kill	0011
	Read	0101
	Rd/Flush	0111
	Wr/Atomic Flush	1001
	Read Atomic	1101
	Rd/Flush Atomic	1111
	?Flush Block	0010
	?DSYNC	0100
	?eieio	1000
	?(reserved)	1011
	?TLB Invalidate	1100
STAT	inst fetchxxxx xxxx xxxx xxx1 xxxx 1xxx xxxx 0xxx	

The least significant bit of the TSIZ label is the TBST- signal. The symbols prefixed by "?" represent signal outputs defined by the PowerPC architecture but not asserted by the PowerPC 7XX. An instruction fetch is indicated by AACK asserted (address & qualifiers valid), R/-W (TT1) asserted for read, and TC0 asserted.

Trigger dialog

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

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

Qualifying Stored Data

The trigger dialog determines what will be acquired by the analyzer and when it will be acquired. The HP E2455B software pre-configures a storage qualification term to exclude wait and idle states from the analyzer's memory.

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

Configuring for State-per-clock mode

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

Change storage qualification from ≠ **idle** to **anystate**.

Capturing an address

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

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

Using the Inverse Assembler

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

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

To disable the instruction cache on the PPC7XX

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

To disable the cache with the emulation module:

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

Register values for controlling the cache

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

To disable the cache with code:

- Disable the instruction cache with the following code:

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

- To also disable the data cache use:

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

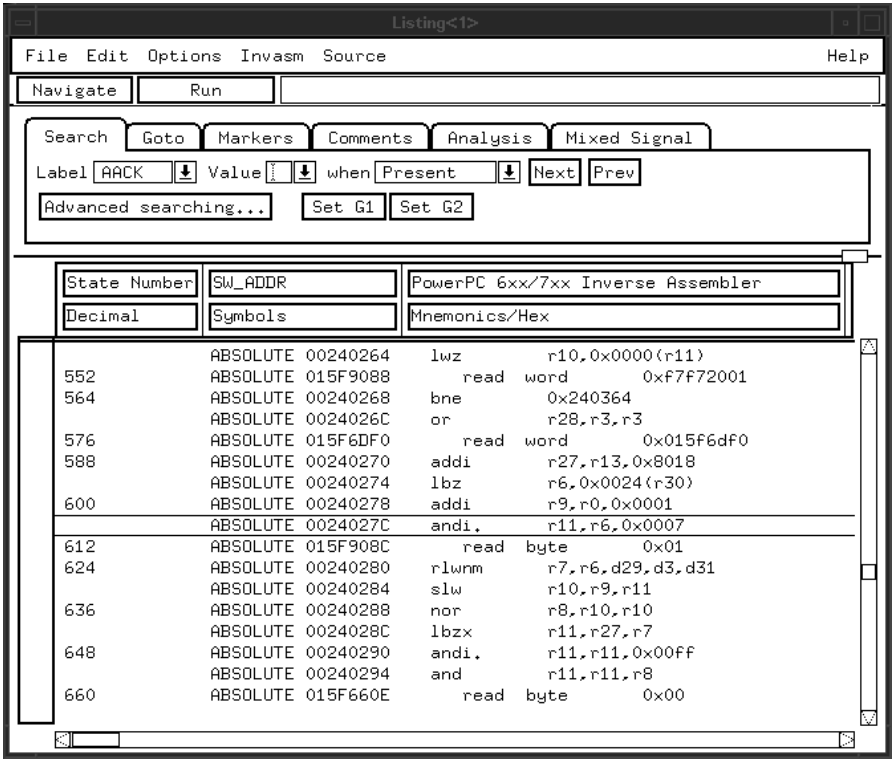
- To invalidate and disable both caches use:

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

To display captured state data

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

Listing Menu.



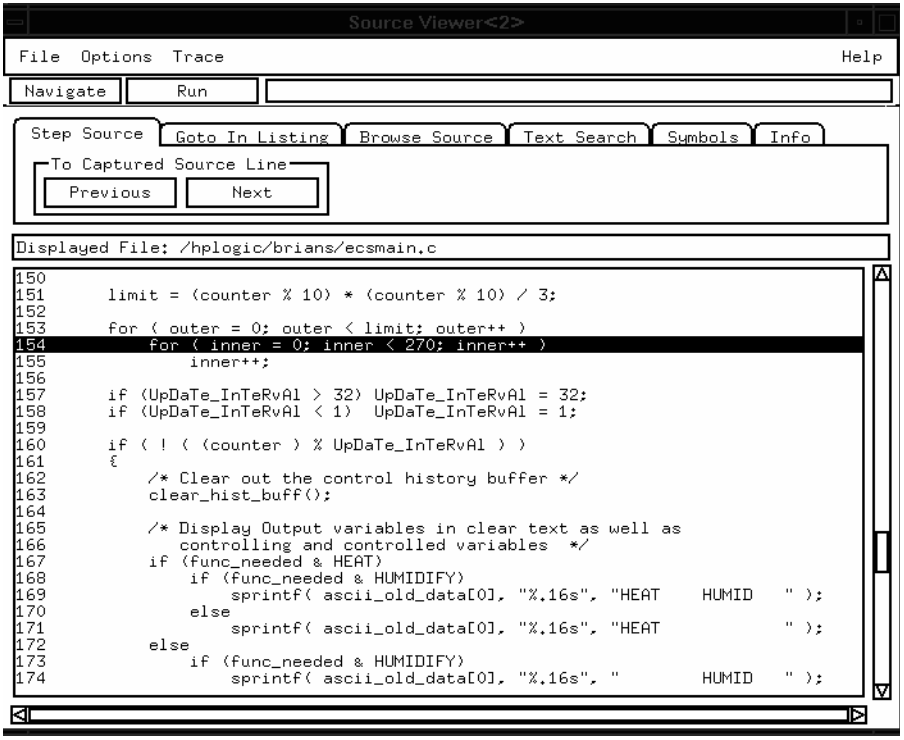
HP 16500C

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

Displaying Data with the HP B4620B Source Correlation Tool Set

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

Source Correlation Tool Set Data.



Inverse assembler output format

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

Interpreting Data

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

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

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

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

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

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

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

Branch Instructions

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

Overfetch Marking

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

- When the 7XX executes a branch instruction, the instructions between the branch and the branch target are not executed. These instructions are indicated with an asterisk "*", or if the bus trace is ambiguous, with an interrogation point "?". If the instruction cache is enabled, the branch target may already be in the cache and will not be fetched over the bus. The remaining cache line containing the branch will be marked as overfetch.

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

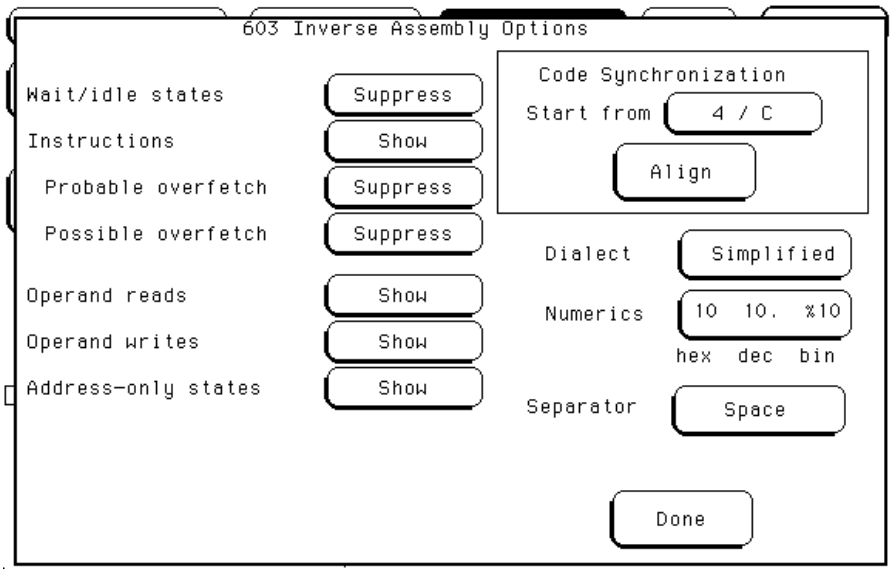
Little-Endian Mode

The inverse assembler is designed to support the native big-endian mode of operation on the PowerPC 7XX. When operating in little-endian mode, the 7XX uses a technique known as “address munging” to convert internal little-endian addresses into external big-endian addresses. Internal and external addresses may differ from one another in the three least significant bits.

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

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

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



Display Filtering

Display filtering is similar to the description on page 102.

Code Synchronization

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

Extended Mnemonics

PowerPC assemblers support a number of extended mnemonics for some popular assembly language instructions as described in the PPC7XX User's Manual. The HP E8129A and HP E2455B inverse assemblers support the following extensions:

- Conditional traps and branches decode the condition mnemonically when possible. For some conditions which have no conventional mnemonics (for example, "signed less than or unsigned greater than"), the condition field is displayed in binary.
- The L bit is omitted as a compare operand. Instead, compares are decoded as "cmpw" (or "?cmpd").
- "Add immediate" instructions with a negative immediate operand are decoded as subtract immediate ("subi").
- "Subtract from" instructions subf and subfc are decoded as subtract instructions sub and subc with the operands exchanged so that "sub r3 r4 r5" is mnemonically interpreted as "r3 = r4 - r5."
- ori r0 r0 0000 is decoded as "nop".
- Add immediate and add immediate shifted instructions, addi and addis, with a null source register are decoded as load immediate and load immediate shifted, li and lis.
- or instructions with identical source registers are decoded as move register, mr.
- nor instructions with identical source registers are decoded as not register, not.
- xor and eqv instructions with identical source and destination registers are decoded as clear and set, clr and set, respectively.
- The cror, crnor, crxor, and creqv instructions map analogously to crmv, crnot, crclr, and crset.
- When the mtcrf instruction field mask specifies the entire cr, it is decoded as mtcr.

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

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

is to shift right word immediate, e.g.

```
srwi    r30  r30  16.
```

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

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

Using the Inverse Assembler

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

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

6

Symbols and Source Code in the Analyzer

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

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

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

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

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

Symbols

Symbols represent values in measurements. For example, the symbol INTERRUPT might represent the value 1FF04000 found on the ADDR label, the address where your interrupt handler begins. You can trigger a measurement on the occurrence of a symbol, and you can have the logic analyzer show symbols in place of values in Listing displays.

Three symbol sources may be used in the logic analyzer:

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

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.

Predefined PPC7XX symbols

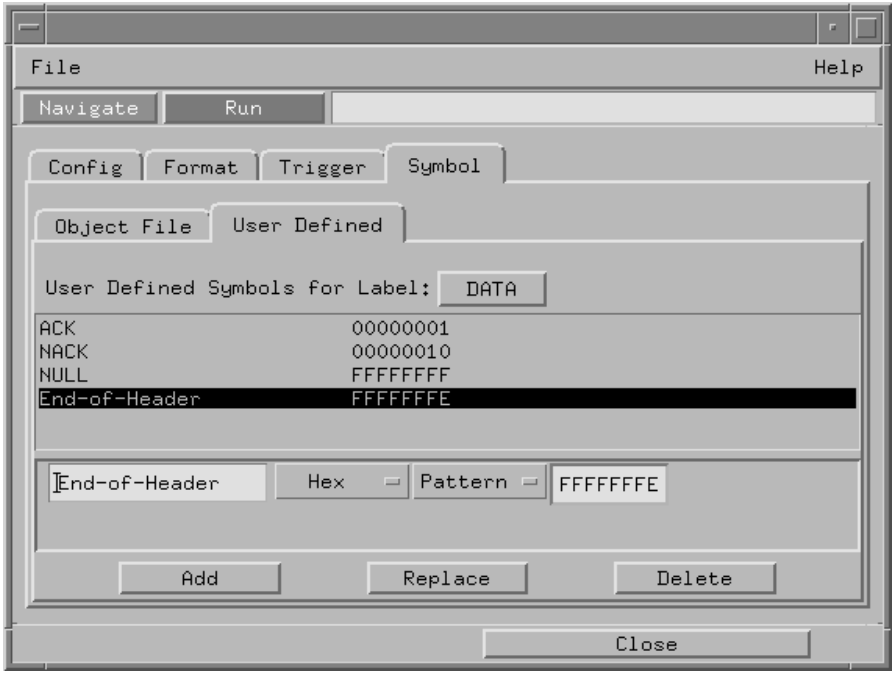
If you're using an HP inverse assembler for the PPC7XX microprocessor, the logic analyzer configuration files include predefined symbols.

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

The predefined PPC7XX symbols are listed on page 83.

User-defined symbols

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



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

Symbol use requirements

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

An accurate bus trace

The HP E2498A inverse assembler provides PPC7XX microprocessor data when the logic analyzer is properly connected to the target system.

Direct address translation

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

An inverse assembler for trace lists

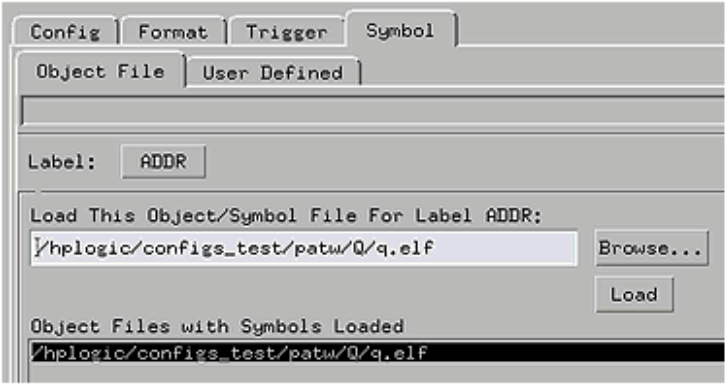
The PPC7XX inverse assembler decodes captured data into program counter (PC) addresses (also known as software addresses) and assembly language mnemonics. Refer to the previous chapter on PPC7XX inverse assembly.

A symbol file

You need an object file containing symbolic debug information in a format the logic analyzer understands. Alternatively, you can use a General Purpose ASCII (GPA) symbol file (see page 241).

To use object file symbols in the HP 16600A/700A

To load object file symbols for address values in the HP 16600A/16700A-series logic analysis system, open the logic analyzer module's Setup window and select the Symbol tab; then, select the Object File tab. Make sure the label is ADDR. From this dialog you can select object files and load their symbol information.



When you load object file symbols into a logic analyzer, a database that correlates symbols and line numbers to addresses in the object file is generated. The Symbol Selector dialog allows you to view the database so you can find a symbol to use in place of a hexadecimal value when defining trigger patterns, trigger ranges, and so on.

If your language tool is not one of those listed on page 139, you can create a symbol file in the General-Purpose ASCII (GPA) file format (refer to the "General-Purpose ASCII (GPA) File Format" chapter).

See Also

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

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

Compilers for PPC7XX

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

Object File Formats

Language System & Version	Format
Diab Data version 4.1a	ELF/DWARF

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

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

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

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

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

See Also

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

Diab Data Compiler Options

The following options should be used:

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

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

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

More information is available on the World Wide Web at:

http://www.diabdata.com

Source Code

The HP B4620B Source Correlation Tool Set lets you:

- View the high-level source code associated with captured data.
- 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.

If you purchased a solution, the HP B4620B Source Correlation Tool Set was included. Otherwise, the source correlation tool set is available as an add-on product for the HP 16600A/16700A-series logic analysis system and must be licensed before you can use it (see the System Admin dialogs for information on licensing).

See Also

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

Requirements for source correlation

The source correlation tool set works with many microprocessors and their embedded software development environments.

However, the overall effectiveness of the source correlation tool set will vary to some degree depending on the specific development environment it is being used in. The following areas affect the performance of the source correlation tool set for different development environments:

- Proper probing and inverse assembly.

  All the information needed to reconstruct the complete address bus of the target system must be acquired by the logic analyzer. When the target system is properly connected to the logic analyzer, the HP E2498A inverse assembler meets this requirement.

  The inverse assembler may need to reconstruct any incomplete address bus information and/or filter out any unexecuted instructions. Also, the Memory Management Units must perform direct address translation.

  When displaying the next or previous instances of a source line, the Source Viewer display uses the PC or SW_ADDR (Software Address) label generated by the inverse assembler.

- Object file symbols.

  The source correlation tool set requires that symbols be loaded into the logic analyzer (refer to the section titled “To use object file symbols in the HP 16600A/700A” on page 138.

  The compiler needs to produce an object file format that is readable by the logic analyzer; otherwise; a general-purpose ASCII (GPA) format file needs to be generated.

- Access to source code files.

  The source correlation tool set requires that you give the logic analysis system access to your program's high-level source files (either by NFS mounting the file system that contains the source files or by copying source files to the logic analysis system disk).

Inverse assembler generated PC (software address) label

In the HP 16600A/16700A-series logic analysis system, the PPC7XX inverse assembler generates a "PC" label. The PC label is displayed as another column in the Listing tool. This label is also known as the Software Address generated by the inverse assembler.

The "Goto this line in listing" commands in the HP 16600A/16700A-series logic analysis system perform a pattern search on the PC label in the Listing display (when an inverse assembler is loaded). Because the inverse assembler is called for each line that is searched, the search can be slow, especially with a deep memory logic analyzer.

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

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

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 HP 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 HP logic analyzer networking must be compatible with the user's network environment. HP 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.

Triggering on Symbols and Source Code

When setting up trigger specifications to capture PPC7XX execution:

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

Using trigger alignment

You should use an 8-bit alignment to avoid missed triggers. The PPC7XX has a 64-bit data bus. Instructions for the PPC7XX are 32 bits long and must be located on even address boundaries. This means that an instruction will often be fetched as the lower 32 bits of one 64-bit memory cycle. When this happens, the address of the instruction in the lower 32 bits of the fetch will not be seen on the address bus. If a trigger was set to occur on this instruction's address, the trigger will not be found by the logic analyzer. For example:

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

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

To help avoid these missed triggers, the trigger dialogs for symbol addresses allow you to "Align" the address to a 1-, 2-, 4-, or 8-byte boundary. Alignment affects the least significant address bits of the trigger specification, either setting them to a "don't care" or "zero" value, depending on the logic analyzer.

Set the alignment for program fetches to the width of the program memory in bytes. For the PPC7XX with 64-bit (8-byte) wide program memory, use 8-byte alignment. Eight-byte alignment will change the least significant three bits ($2^3 = 8$) of the trigger. Address 84 with 8-byte alignment results in a trigger address range of 80 through 87 for some logic analyzers (3 don't care bits), or an address of 80 on the other analyzers (three 0 bits). Note that either of these triggers would catch line #14 in the example above.

To correlate relocatable code using the address offset

You need to adjust the source correlation tool set to compensate for relocatable code segments or memory management units that produce fixed code offsets. The offset field in the trigger menu allows you to offset the symbol address. Entering the appropriate address offset will cause the source correlation tool set to reference the correct symbol information for the relocatable or offset code.

To adjust for prefetches, use a trigger offset of 8 (prefetch queue depth) to avoid triggering on prefetched instructions. Note that this is not a foolproof scheme, since this may result in a missed trigger if a branch takes place between the base address and the offset address. For the PPC7XX, an offset of 8 is large enough to overcome the prefetch queue.

Be aware of prefetches and adjust your triggering to compensate for them. Note that the PPC7XX has a good Branch Prediction Unit (PBU), and often, when executing loops, the processor will NOT fetch instructions beyond the end of the loop. This means that on most loops, an offset may not be required to avoid a false trigger.

Using storage qualification

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

The source correlation tool set can exhibit long responses to requests for the next source line if the current trace listing corresponds to code from a library that is not in the source code search path. Logic analyzer storage qualification can be used to avoid capturing library code routines.

7

Connecting and Configuring the Emulation Module

Connecting and Configuring the Emulation Module

This chapter shows you how to connect the emulation module to the target system and how to configure the emulation module and target processor.

NOTE: The emulation module is compatible with the PowerPC 740/750, but not the PowerPC 745/755.

Overview

Here is a summary of the steps for connecting and configuring the emulation module:

1. Make sure the target system is designed to work properly with the emulation module (See page 154).
2. Install the emulation module in your logic analysis system, if necessary (See page 159).

   Use the Setup Assistant to guide you through steps 3-6 (See page 23). Use this manual for additional information, if desired.
3. Connect the emulation module to your target system using the 50-pin cable and the TIM (See page 164).
4. Update the firmware of the emulation module, if necessary (See page 166).
5. Verify communication between the emulation module and the target (See page 167).
6. Configure the emulation module (See page 168).
7. Test the connection between the emulation module and the target (See page 181).
8. Connect a debugger to the emulation module, if applicable (See page 183).

See Also See Chapter 8, "Using Debuggers (with the Emulation Module)," beginning on page 183 for information on configuring the emulation module with a debugger.

Using the Emulation Control Interface

The Emulation Control Interface in your HP 16600A/700A-series logic analysis system allows you to control an emulator (an emulation module or an emulation probe).

As you set up the emulation module, you will use the Emulation Control Interface to:

- Update firmware (which reloads or changes the processor-specific personality of the emulator).
- Change the LAN port assignment (rarely necessary).
- Run performance verification tests on the emulator.

The Emulation Control Interface allows you to:

- Run, break, reset, and step the target processor.
- Set and clear breakpoints.
- Read and write registers.
- Read and write memory.
- Read and write I/O memory.
- View memory in mnemonic form.
- Read and write the emulator configuration.
- Download programs (in Motorola S-Record, Intel Hex, plain binary, ELF object, COFF object, or IEEE695 object format) to the target system RAM or ROM.
- View emulator status and errors.
- Write and play back emulator command script files .

If you have an emulation probe, this interface also allows you to configure the LAN address of the emulation probe.

Using the logic analysis system's intermodule bus does not require the Emulation Control Interface to be running. If the emulation module icon is in the Intermodule window, then it will be able to send and receive signals. Therefore if you are using a debugger, you can use an analyzer to cause a break.

Using a debugger with the Emulation Control Interface is not recommended because:

- The interfaces can get out of synchronization when commands are issued from both interfaces. This causes windows to be out-of-date and can cause confusion.
- Most debuggers cannot tolerate another interface issuing commands and may not start properly if another interface is running.

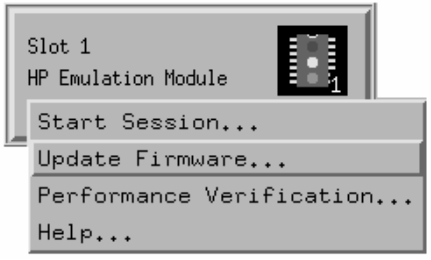
See Also

All of the Emulation Control Interface windows provide online help with a **Help** button or a **Help→On this window** menu selection. Refer to the online help for complete details about how to use a particular window.

To start the Emulation Control Interface from the main System window

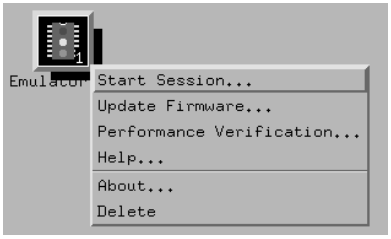
1 In the System window, click the emulation module icon.

2 Select Start Session....



To start the Emulation Control Interface from the Workspace window

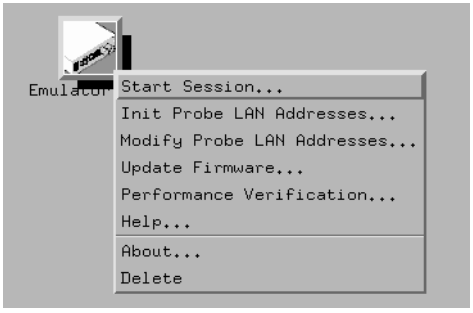
1. Open the Workspace window.
2. Drag the Emulator icon onto the workspace.
3. Right-click on the Emulator icon, then select Start Session....



To start the Emulation Control Interface from the Workspace window for an emulation probe

If you have a stand-alone emulation probe connected to the logic analysis system via LAN, use the Emulation Probe icon instead of the Emulation Module icon.

1. Open the Workspace window.
2. Drag the Emulation Probe icon onto the workspace.
3. Right-click on the Emulation Probe icon, then select Start Session....



4. In the Session window, enter the IP address or LAN name of the emulation probe, then click Start Session.

Designing a Target System for the Emulation Module

The following sections describe design considerations for your target system to operate properly with the emulation module.

Target System Requirements for PowerPC 7XX

MMU Support

When the MMU is enabled in the PowerPC hardware, and the HP Emulator is configured for effective addresses, all memory addresses given to the emulator are assumed to be effective addresses (logical addresses). The emulator uses the MMU block address translation (BAT) registers, segment registers, hash tables, and other special-purpose MMU registers to compute each corresponding physical address. The requested memory operation is then performed using the physical address.

Operational notes:

- The emulator attempts to perform address translation only if the MSR[IR] and/or the MSR[DR] bits are set (=1) AND the emulator is configured to do translation (cf address=effective). The emulator configuration may be changed using the cf command:
  - cf address=effective (power up default value)
  - cf address=physical
- If both the MSR[IR] and MSR[DR] are set, the emulator will perform address translations by first searching the IBATs and then the DBATS, if no match is found in the IBATs. Note that the PowerPC silicon allows the IBAT and DBAT registers to specify overlapping effective address ranges. Avoid defining overlapping ranges. These make debugging more difficult because the emulator can use the IBATs to translate addresses intended for the DBATs.
- If an effective address is not found in the MMU translation tables, the emulator will return an error and will not perform the requested operation.
- Cache coherency is maintained during emulator MMU translations.

- Be sure the translation enable/disable condition is the same when you set and clear breakpoints. If a breakpoint is set while translation is enabled and then cleared while translation is disabled, the result will be erroneous and unpredictable. This is also true if a breakpoint is set while translation is disabled and then cleared while translation is enabled.
- The emulator ignores read-only restrictions defined in the MMU. (i.e. The emulator may attempt to write to memory that has been defined by the MMU as read-only.)
- MMU translation is automatic and transparent to debuggers connected to the emulator.

Unsupported modes

Target systems which use any of the following modes of operation are not currently supported:

- Address parity is not generated on external address bus operations. Accesses to devices that check parity will fail.

$\overline{\text{QACK}}$ signal

If the target development board does not use the $\overline{\text{QACK}}$ signal , the board must have a pull down resistor to pull this signal low. This allows the PowerPC to enter the debug state. Recommended value: 1KΩ or less.

TDO, TDI, TCK, TMS and $\overline{\text{TRST}}$ signals

TDO, TDI, TCK, TMS and $\overline{\text{TRST}}$ signal traces between the JTAG connector and the PPC7XX must be less than 3 inches long. If these signals are connected to other nodes, the other nodes must be daisy chained between the JTAG connector at one end and the PowerPC microprocessor at the other end. These signals are sensitive to crosstalk and must not be routed along active signals such as clock lines on the target board.

The TDI, TCK, TMS and $\overline{\text{TRST}}$ signals must not be actively driven by the target system when the JTAG port is being used.

Reset signals

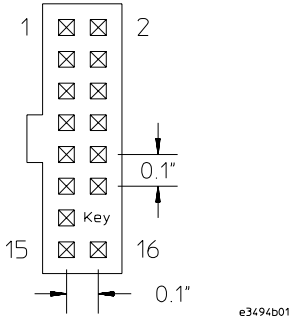
The $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ and $\overline{\text{TRST}}$ signals from the JTAG connector must be logically ORed with the $\overline{\text{HRESET}}$, $\overline{\text{SRESET}}$ and $\overline{\text{TRST}}$ signals that connect to the processor on the target system. They cannot be "dotted" or "wire-ORed" on the board. The ORed signals should only reset the processor and no other devices on the target system.

The HP emulation module adds capacitance to all target system signals routed to the JTAG connector. This added capacitance may reduce the rise time of the $\overline{\text{SRESET}}$ or the $\overline{\text{HRESET}}$ signal beyond the processor specifications. If so, the target may need to increase the pull-up current on these signal lines.

PowerPC JTAG interface connections and resistors

The target system must have a 16-pin male 2x8 header connector with the following dimensions:

JTAG Header Connector (top view)



Position 14 of the connector on the target system must not contain a pin. The cable supplied with the emulation module can only be installed if pin 14 has been removed from the header.

Refer to the table on the next page to see the signals that must be connected to pins on the JTAG Header Connector.

Place the JTAG Header Connector as close as possible to the processor to ensure signal integrity.

PowerPC 7XX Connections

Header Pin Number	Signal Name	I/O	Board Resistor
1	TDO	Out	
2	Not connected		
3	TDI	In	1kΩ pulldown
4	$\overline{\text{TRST}}$	In	10kΩ pullup
5	Not connected		
6	+POWER[1]		1kΩ series[2]
7	TCK	In	10kΩ pullup
8	Not connected		
9	TMS	In	10kΩ pullup
10	Not connected		
11	$\overline{\text{SRESET}}$	In	10kΩ pullup
12	Not connected		
13	$\overline{\text{HRESET}}$	In	10kΩ pullup
14	KEY		
15	$\overline{\text{CSTP\_OUT}}$	Out	1kΩ pullup
16	GND		
	$\overline{\text{QACK}}$ [3]	In	1kΩ pulldown
	L2_TEST_CLK	In	10kΩ pullup
	L1_TEST_CLK	In	10kΩ pullup
	$\overline{\text{LSSD\_MODE}}$	In	10kΩ pullup
	ARRAY_WR	In	10kΩ pullup

[1] The +POWER signal is sourced from the target system and is used as a reference signal. It should be the power signal being supplied to the processor (either +3.3V or +5V). It does not supply power to the HP emulation module.

[2] This 1kΩ series resistor provides short circuit current limiting protection only. If the resistor is present, it should be 1kΩ or less.

[3] If the target system does not use this signal, the board must have a 1kΩ pulldown resistor connected to this pin. This signal allows the HP emulation module to force the processor into soft stop mode. If the target system does use this signal, it should provide logic so that $\overline{\text{QACK}}$ goes low in response to a $\overline{\text{QREQ}}$.

Installing the Emulation Module

Your emulation module may already be installed in your logic analysis system. If you need to install an emulation module yourself, follow the instructions on the pages which follow.

CAUTION:

These 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.

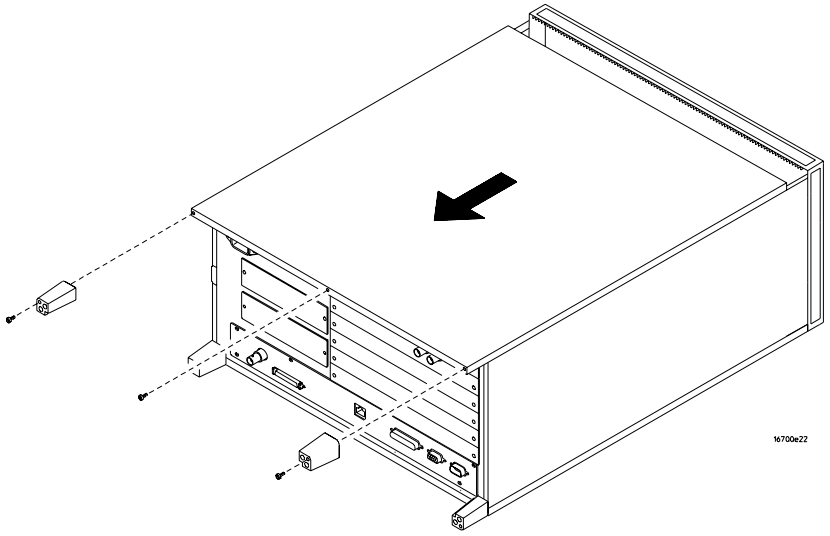
Electrostatic discharge can damage electronic components. Use grounded wrist straps and mats when you handle modules.

To install the emulation module in an HP 16700A-series logic analysis system or an HP 16701A expansion frame

You will need T-10 and T-15 Torx screwdrivers (supplied with the emulation module).

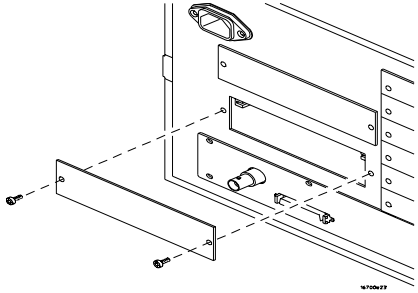
1. Turn off the logic analysis system and REMOVE THE POWER CORD.

   Remove any other cables (including mouse or video monitor cables).
2. Turn the logic analysis system frame upside-down.
3. Remove the bottom cover.

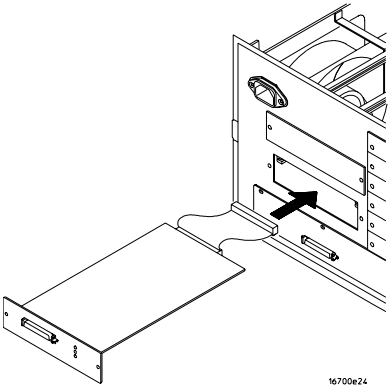


4. Remove the slot cover.

   You may use either slot.

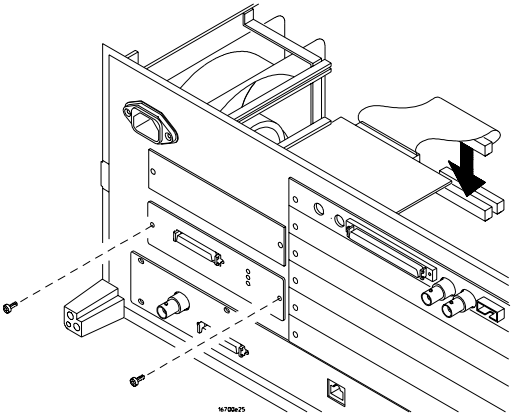


5 Install the emulation module.



6 Connect the cable and re-install the screws.

You may connect the cable to either of the two connectors. If you have two emulation modules, note that many debuggers will work only with the "first" module: the one toward the top of the frame ("Slot 1"), plugged into the connector nearest the back of the frame.



7 Reinstall the bottom cover, then turn the frame right-side-up.

8 Plug in the power cord, reconnect the other cables, and turn on the logic analysis system.

The new emulation module will be shown in the system window.

See Also

See page 166 for information on giving the emulation module a "personality" for your target processor.

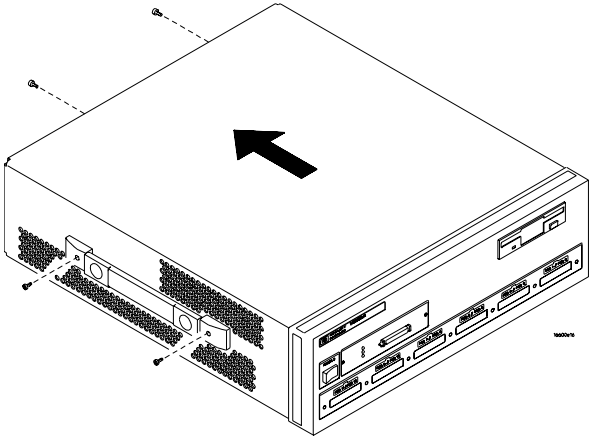
To install the emulation module in an HP 16600A-series logic analysis system

You will need T-8, T-10, and T-15 Torx screwdrivers (supplied with the emulation module).

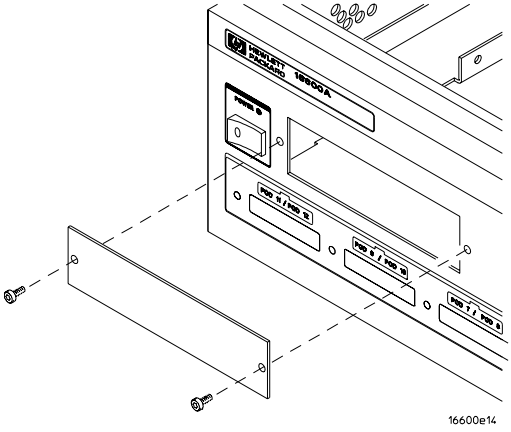
1 Turn off the logic analysis system and REMOVE THE POWER CORD.

Remove any other cables (such as probes, mouse, or video monitor).

2 Slide the cover back.

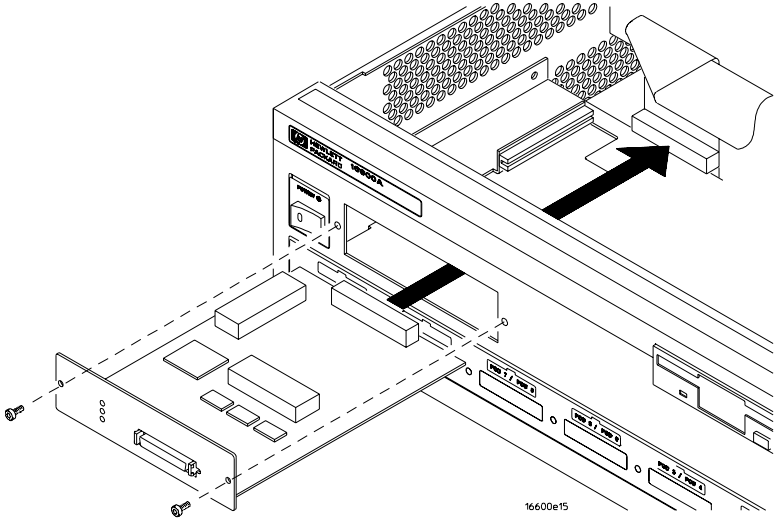


3 Remove the slot cover.



4 Install the emulation module.

5 Connect the cable and re-install the screws.



6 Reinstall the cover.

Tighten the screws snugly (2 N-m or 18 inch-pounds).

7 Plug in the power cord, reconnect the other cables, and turn on the logic analysis system.

The new emulation module will be shown in the system window.

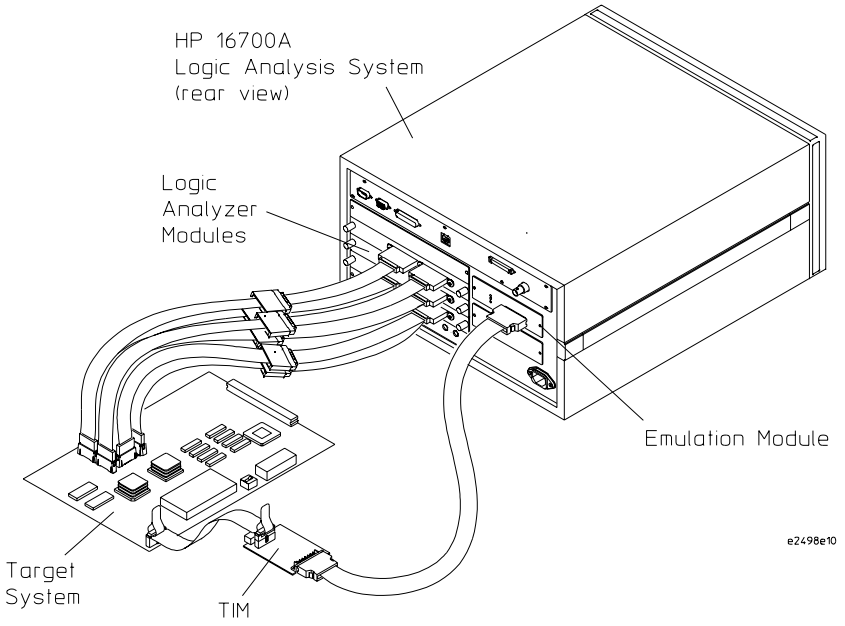
See Also See page 166 for information on giving the emulation module a "personality" for your target processor.

To test the emulation module

If this is the first time that you have used the emulation module, you should run the built-in performance verification test before you connect to a target system. Refer to page 285 for information on performance verification.

Connecting the Emulation Module to the Target System

Connect the emulation module and TIM to a target system directly through a JTAG connector on the target board. Use the procedure on the following page.



After you have connected the emulation module to your target system, you may need to update the firmware in the emulation module.

See Also

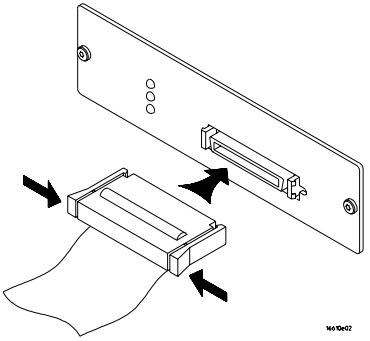
For information on designing a JTAG port on your target board, see page 154.

For a list of the parts supplied with the emulation module, see page 28.

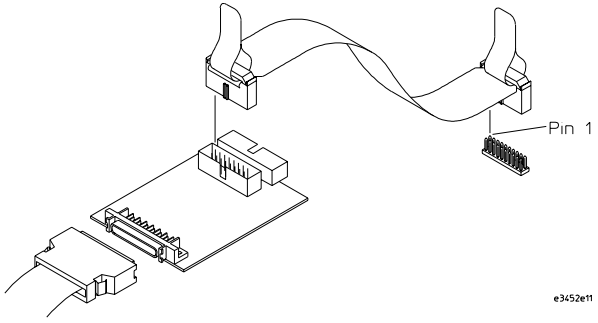
To connect to a target system using a JTAG port

The emulation module can be connected to a target system through a 16-pin JTAG port connector. The emulation module should be connected to a 16-pin male 2x8 header connector on the target system using the 16-conductor cable assembly provided.

1. Turn off the target system and disconnect it from all power sources.
2. Plug one end of the 50-pin cable into the emulation module.



3. Plug the other end of the 50-pin cable into the target interface module.
4. Plug one end of the 16-pin cable into the target interface module.
5. Plug the other end of the 16-pin cable into the JTAG port on the target system.



6. Turn on the power to the logic analysis system and then the target system.

See Also

See "Designing a Target System for the Emulation Module" on page 154 for information on designing a target system for use with the emulation module.

To Update Firmware

After you have connected the emulation module to your target system, you may need to update the firmware to give it the right "personality" for your processor. You must update the firmware if:

- The emulation module is being connected to a new analysis probe or TIM, or
- The emulation module was not shipped already installed in the logic analysis system, or
- You have an updated version of the firmware from HP.

To update the firmware:

1. End any run control sessions which may be running.
2. In the Workspace window, remove any Emulator icons from the workspace.
3. Install the firmware onto the logic analysis system's hard disk, if necessary.
4. In the system window, click the emulation module and select Update Firmware.
5.
6. In the Update Firmware window, select the firmware version to load into the emulation module.
7. Click Update Firmware.

   In about 20 seconds, the firmware will be installed and the screen will update to show the current firmware version.

See Also

"Installing Software" beginning on page 33 for instructions on how to install the firmware files on the hard disk.

To display current firmware version information

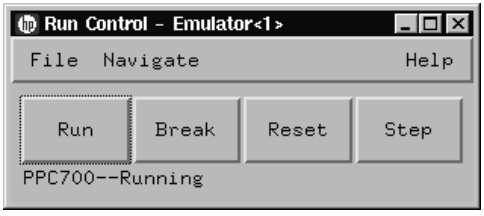
- In the Update Firmware window, click Display Current Version.

  There are usually two firmware version numbers: one for "Generics" and one for the personality of your processor.

To verify communication between the emulator and target system

1. Turn on the target system.
2. Start the Emulation Control Interface.

   If the electrical connections are correct, and if the emulator firmware and analysis probe or TIM match your target processor, the Run Control window should be displayed:



Configuring the Emulation Module

The emulation module has several user-configurable options. These options may be customized for specific target systems and saved in configuration files for future use.

The easiest way to configure the emulation module is through the Emulation Control Interface in an HP 16600A or HP 16700A logic analysis system.

If you use the Emulation Control Interface, please refer to the online help in the Configuration window for information on each of the configuration options.

Other ways to configure the emulation module are by using:

- the emulation module's built-in terminal interface
- your debugger, if it provides an "emulator configuration" window which can be used with this HP emulation module

What can be configured

There are two categories of configuration items: general configuration and cache configuration.

The default powerup configuration will generally work with many target systems if the cache is turned off.

If the instruction and data caches are both turned off, the cache configuration items are meaningless and can be ignored.

The following option can be configured using built-in commands:

- Restriction to real-time runs.

The built-in "help cf" command also lists the following options, which are provided only for compatibility with standalone emulation probes:

- BNC break in behavior.
- BNC trigger out behavior.

General Configuration

- JTAG clock speed
- Reset operation
- Memory read delays
- Memory write delays
- Parity bit information

Cache Configuration

- Memory read operation
- Data memory write operations
- Instruction memory write operations

To configure using the Emulation Control Interface

The easiest way to configure the emulation module is to use the Emulation Control Interface.

1. Start an Emulation Control Interface session.

   In the system window, click the Emulation Control Interface icon, and then select "Start Session...".

2. Open a Configuration window.

   Select "Configuration..." from the Emulation Control Interface icon or from the Navigate menu in any Emulation Control Interface window.

3. Set the configuration options, as needed.

   The configuration selections will take effect when you close the configuration window or when you move the mouse pointer outside the window.

4. Save the configuration settings.

   To save the configuration settings, open the File Manager window and click **Save...**.

See Also

Help →Help on this window in the Configuration window for information on each of the configuration options.

Help in the Emulation Control Interface menu for help on starting an Emulation Control session.

To configure using the built-in commands

If you are unable to configure the emulation module with the Emulation Control Interface or a debugger interface, you can configure the emulation module using the built-in "terminal interface" commands.

1. Connect a telnet session to the emulation module over the LAN.

   For example, on a UNIX system, for an emulation module in Slot 1 enter:

   `telnet` *`LAN_address`* `6472`
2. Enter `cf` to see the current configuration settings.
3. Use the `cf` command to change the configuration settings.

See Also

Enter `help cf` for help on the configuration commands.

For information on connecting using telnet, and for information on other built-in commands, see page 271.

Example

To see a complete list of configuration items, type "help cf". This command displays:

```
  cf - display or set emulation configuration

     cf                - display current settings for all config items
     cf <item>         - display current setting for specified <item>
     cf <item>=<value> - set new <value> for specified <item>
     cf <item> <item>=<value> <item> - set and display can be combined

   help cf <item>    - display long help for specified <item>

   --- VALID CONFIGURATION <item> NAMES ---
     rrt       - Restrict to real-time runs
     reset     - Configure reset actions
     speed     - Set JTAG clock
     mrdop     - Configure mem read operation
     dmwrop    - Configure D mem write operation
     imwrop    - Configure I mem write operation
     mrddel    - Set memory read delay
     mwrdel    - Set memory write delay
     breakin   - Select BNC break input option
     trigout   - Select BNC trigger output option
     parity    - Enable/disable data parity
 M>
```

To see a more detailed description of any configuration item, use the command "help cf <item>". For example:

```
M>help cf rrt

   Restrict to real-time runs

     cf rrt=yes
     cf rrt=no

   If yes (and while the processor is running the user program), any
   command that requires the processor to be stopped will be rejected.
```

```
   For example 'reg' and 'm'.

   If no, commands that require the processor to be stopped will
   actually stop the processor, execute then resume running the
   processor.

 M>
```

To see a list of the current configuration settings, use "cf":

```
M>cf
   cf rrt=yes
   cf reset=runrom
   cf speed=1
   cf mrdop=mm
   cf dmwrop=mm
   cf imwrop=upd_dcu
   cf mrddel=0
   cf mwrdel=0
   cf breakin=off
   cf trigout=fixhigh
   cf parity=off
 M>
```

To configure using a debugger

Because the HP emulation module can be used with several third-party debuggers, specific details for sending the configuration commands from the debugger to the emulation module cannot be given here. However, all debuggers should provide a way of directly entering terminal mode commands to the emulation module. Ideally, you would create a file that contains the modified configuration entries to be sent to the emulation module at the beginning of each debugger session.

See Also

See Chapter 8, "Using Debuggers (with the Emulation Module)," beginning on page 183 for more information, or consult your debugger manual.

To configure restriction to real-time runs

Real-time runs configuration

Value	Emulation module configured for	Built-in command
no	Allows commands which break to the monitor. Examples include commands which display memory or registers. These commands break to the monitor to access the target processor, then resume the user program.	cf rrt=no
yes	No commands are allowed which break to the monitor, except "break," "reset," "run," or "step." The processor must be explicitly stopped before these commands can be performed. (Default)	cf rrt=yes

If your debugger allows displaying or modifying memory or registers while the processor is running, you must set rrt=no in order to use this feature.

To configure the JTAG clock speed (communication speed)

The HP emulation module needs to be configured to communicate at a rate which is compatible with your target processor. The JTAG Clock speed is independent of processor clock speed. In general, speed=1 can always be used and provides the best performance. With some target systems that have additional loads on the JTAG lines or with target systems that do not quite meet the requirements described in the "Designing a Target System" chapter (page 154), setting speed to a slower setting may enable the module to work.

Processor clock speed configuration

Value	Processor clock (TCK) is at least	Built-in command
1	10 MHz (default)	cf speed=1
2	5 MHz	cf speed=2
3	2.5 MHz	cf speed=3
4	1.25 MHz	cf speed=4
5	625 kHz	cf speed=5
6	312 kHz	cf speed=6
7	156 kHz	cf speed=7

To configure reset operation

The reset configuration item controls what kind of reset is performed and what state the processor will be in after the reset.

Reset configuration

Value	Effect of a reset from the emulation module	Built-in command
runrom	Reset the processor and cause it to start running user code at address FFF00100H.(Default)	cf reset=runrom
rom	Reset the processor and cause it to stop at address 0FFF00100H.	cf reset=rom
runram	Reset the processor and cause it to start running user code at address 00000100H.	cf reset=runram
ram	Reset the processor and cause it to stop at address 00000100H.	cf reset=ram
jtag	Just reset the JTAG interface on the processor. The processor itself will not be reset. This may help in some cases where communications are lost, however all the other reset settings reset the JTAG interface as part of the reset sequence so this setting will only rarely be useful.	cf reset=jtag

To set memory read delays

The memory read delay setting delays the number of microseconds specified during memory reads. It is provided for accessing slow devices like memory mapped IO.

- To set the memory read delay using the built-in terminal interface, use the `cf mrddel=`<*delay in usec*> command.

The <*delay in usec*> must be in the range 0-10000000. Set this delay to the smallest number possible for best performance. It delays all reads by the number of microseconds specified.

Default: cf mrddel=0

To set memory write delays

The memory write delay setting delays memory writes by the number of microseconds specified. It is provided for accessing slow devices like memory mapped IO.

- To set the memory write delay using the built-in terminal interface, use the `cf mwrdel=`<*delay in usec*> command.

The <*delay in usec*> must be in the range 0-10000000. Set this to the smallest number possible for best performance.

Default: cf mwrdel=0

To generate parity bits on memory operations

The PowerPC processor generates parity bits on both address and data lines when running user code. When used in debug mode these bits must be generated separately slowing down memory operations. Since memory operations on the PowerPC are slow as it is and many target systems do not check parity, parity is only generated if requested.

Parity configuration

Value	Emulation module configured for	Built-in command
off	Do not generate the parity bits for memory operations from the emulation module. This provides better performance, but will not work correctly when accessing devices that check the parity bits.(Default)	cf parity=off
on	Generate the parity bits for memory operations. Currently, only parity bits for the memory data lines are generated. Parity bits on the address lines are not. This may change in future firmware versions.	cf parity=on

To configure the memory read operation

The memory read operation configuration entry defines how the memory and cache interact during a memory read operation. If both instruction and data caches are turned off (bits ICE and DCE in the register HID0 are zero), this configuration setting has no effect and a memory read will always return the contents of physical memory.

Memory read configuration

Value	emulation module configured for	Built-in command
mm	A memory read from an address that is valid in either the data or instruction cache will return the contents of the cache. Memory reads from addresses not valid in either cache will return the contents of the physical memory.(Default)	cf mrdop=mm
phys	A memory read will always return the contents of physical memory.	cf mrdop=phys

Using the mrdop=phys setting with the cache enabled may show data that is no longer valid. Use this setting only for solving cache problems where you really need to see the contents of physical memory. For general operation, the "mm" setting should always be used.

The instruction cache in PPC740 and PPC750 is encoded. The emulator will decode the content of the instruction cache before displaying it. However, the emulator will only decode valid instructions. Invalid instructions in the cache will be displayed in coded form, which might not match the content of memory.

To configure data memory write operations

Although the PowerPC processor has one contiguous physical memory address space that can hold both data and instructions, it has separate caches for instructions and data. These separate caches must be considered in order to keep the caches and memory coherent during memory write operations. These settings are only used for memory write operations. Code download always writes to physical memory and disables any cache entries containing addresses written for improved performance. Some host interfaces use the code download mode for all memory write operations so this setting may or may not have any effect on your debugger.

Only the memory write command allows specifying instruction or data memory operations. This may not be provided by your debugger interface. If not specified, memory write operations are always instruction memory.

If the data cache is disabled, a data memory write will always write to physical memory and this configuration setting is ignored.

Memory write configuration

Value	emulation module configured for	Built-in command
mm	A data writes to addresses that are valid in the data cache will write the value only to the cache and mark the cache line modified as "dirty" which will indicate to the cpu that the cache line must be written to memory. A data write that is not valid in the data cache will only be written to physical memory.(Default)	cf dmwrop=mm
thru	A data memory write to an address that is valid in the data cache will write to both cache and physical memory. If the address is not valid in the cache, only physical memory will be modified.	cf dmwrop=thru
bypass	A data memory write will only be written to physical memory ignoring the cache.	cf dmwrop=bypass

The **cf dmwrop=bypass** setting should be used with extreme caution because dirty cache entries may be written by the processor over the new data value written to memory by the emulation module.

To configure instruction memory write operations

Although the PowerPC processor has one contiguous physical memory address space that can hold both data and instructions, it has separate caches for instructions and data. These separate caches must be considered in order to keep the caches and memory coherent during memory write operations. Code download always writes to physical memory and disables any cache entries containing addresses written for improved performance. Some host interfaces use the code download mode for all memory write operations so this setting may or may not have any effect on your debugger.

Only the memory write command allows specifying instruction or data memory operations. Access to this may not be provided by your debugger interface. If not specified, memory write operations are always instruction memory.

If the instruction and data caches are both disabled, an instruction memory write will always write to physical memory and this configuration setting is ignored. If the instruction cache is disabled, instruction memory writes will always write to physical memory and the data cache will be either updated or bypassed depending on this configuration setting.

This configuration setting controls the behavior of both caches when doing instruction memory writes so that instruction memory writes can be used for all memory operations if desired.

Instruction memory write configuration

Value	Emulation module configured for	Built-in command
upd_dcb	This stands for instruction cache update, data cache bypass. An instruction memory write to an address that is valid in the instruction cache will write the value to both the instruction cache and memory. The data cache will be bypassed even if the address is valid in the data cache.	cf imwrop=upd_dcb
upd_dcu	This stands for update instruction cache and update data cache. An instruction memory write to an address that is valid in both caches will write the value to both caches and physical memory. (Default)	cf imwrop=upd_dcu
inv_dcb	This stands for instruction cache invalidate and data cache bypass. An instruction memory write will invalidate the instruction cache if valid and write only to physical memory. The data cache is not modified even if valid.	imwrop=inv_dcb
inv_dcu	This stands for instruction cache invalidate and data cache update. An instruction memory write will invalidate the instruction cache if valid and write to physical memory. The data cache will also be updated if the address is valid in the data cache	imwrop=inv_dcu

Setting imwrop to upd_dcb or inv_dcb should be used with caution since dirty cache entrys in the data cache may overwrite the memory just modified by the HP emulation module.

Testing the emulator and target system

After you have connected and configured the emulator, you should perform some simple tests to verify that everything is working.

See Also

"Troubleshooting the Emulation Module" on page 267 for information on testing the emulator hardware.

To test memory accesses

1. Start the Emulation Control Interface and configure the emulator, if necessary.
2. Open the Memory window.
3. Write individual locations or fill blocks of memory with patterns of your choosing.

   The access size is the size of memory access that will be used to write or read the memory values.
4. Use the Memory I/O window to stimulate I/O locations by reading and writing individual memory locations.

To test with a running program

To more fully test your target, you can load simple programs and execute them.

1. Compile or assemble a small program and store it in a Motorola S-Record or Intel Hex file.
2. Use the Load Executable window to download the program into RAM or flash memory.
3. Use the Breakpoints window to set breakpoints. Use the Registers window to initialize register values.

The new register or breakpoint values are sent to the processor when you press the Enter key or when you move the cursor out of the selected register field.

4. In the Run Control window, click Run.
5. Use the Memory Mnemonic window to view the program and use the Memory window to view any output which has been written to memory.

8

Using Debuggers (with the Emulation Module)

Several prominent companies design and sell state-of-the-art source debuggers that work with the HP emulation module and emulation probe.

Benefits of using a debugger

The debugger will enable you to control the execution of your processor from the familiar environment of your debugger. Using a debugger lets you step through your code at the source-code level.

With a debugger connection, you can set breakpoints, single-step through source code, examine variables, and modify source code variables from the debugger interface. The debugger can also be used to download executable code to your target system.

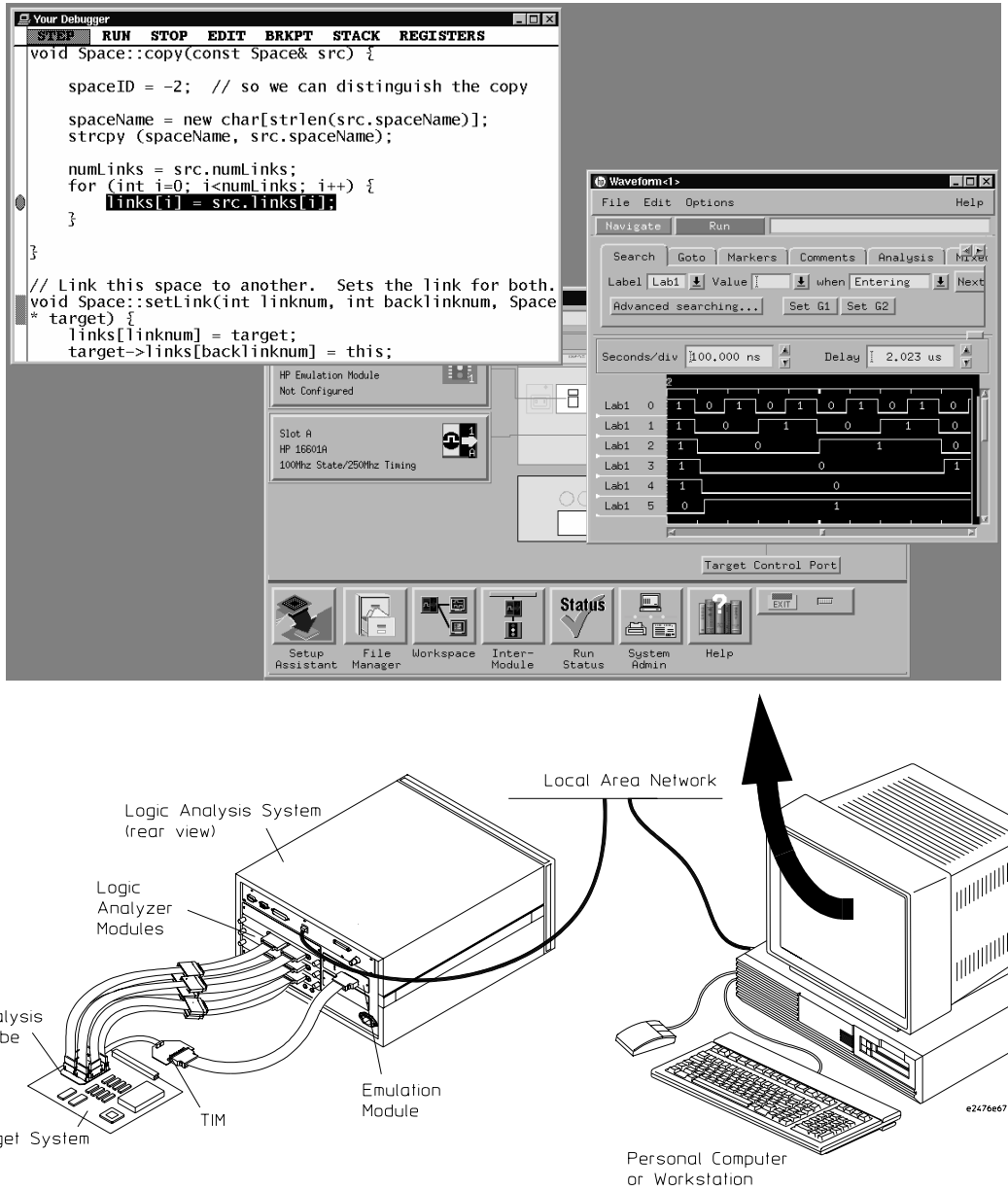
Using a debugger to connect the emulation module allows the entire design team to have a consistent interface from software development to hardware/ software integration.

Debugger interfaces must be ordered directly from the debugger vendor.

Compatibility with other logic analysis system tools

You can use your logic analysis system to collect and analyze trace data while you use your debugger. If you are using an X windows workstation or a PC with an X terminal emulator, you can display the logic analyzer windows right next to your debugger.

Here is an example of what the display on your PC or workstation might look like.



Minimum requirements

To use a debugger with the emulation module, you will need:

- A debugger which is compatible with the emulation module
- A LAN connection between the PC or workstation that is running the debugger, and the HP 16600A or HP 16700A logic analysis system
- A web browser, X windows, or an X terminal emulator (such as Reflection X on a PC). This is required if you wish to use the logic analysis system user interface on your PC or workstation screen, along with the debugger.

Is your debugger compatible with the emulation module?

Ask your debugger vendor whether the debugger can be used with an HP emulation module or HP emulation probe (also known as a "processor probe" or "software probe").

LAN connection

You will use a LAN connection to allow the debugger to communicate with the emulation module.

Compatibility with the Emulation Control Interface

Do not use the logic analysis system's Emulation Control Interface and your debugger at the same time.

Setting up Debugger Software

The instructions in this manual assume that your PC or workstation is already connected to the LAN, and that you have already installed the debugger software according to the debugger vendor's documentation.

To use your debugger with the emulation module, follow these general steps:

- Connect the emulation module to your target system. See page 150.
- Connect the logic analysis system to the LAN. See page 188.
- Export the logic analysis system's display to your PC (see page 191)or workstation (see page 190), or use a web browser (see page 190).
- Configure the emulation module. See page 168.
- Begin using your debugger.

If you use the Emulation Control Interface to configure the emulation module, remember to end the Emulation Control Interface session before you start the debugger.

CAUTION: Do not use the Emulation Control Interface at the same time as a debugger.

The Emulation Control Interface and debuggers do not keep track of commands issued by other tools. If you use both at the same time, the tools may display incorrect information about the state of the processor, possibly resulting in lost data.

See Also Refer to the documentation for your debugger for more information on connecting the debugger to the emulation module.

To connect the logic analysis system to the LAN

Information on setting up a LAN connection is provided in the online help or installation manual for your logic analysis system.

Your debugger will require some information about the LAN connection before it can connect to the emulation module. This information may include:

- IP address (Internet address) or LAN name of the logic analysis system.
- Gateway address of the logic analysis system.
- Port number of the emulation module.

Port numbers for emulation modules

Port number	Use for
Debugger connections	
6470	Slot 1 (First emulation module in an HP 16600A/700A-series logic analysis system)
6474	Slot 2 (Second emulation module in an HP 16700A-series system)
6478	Slot 3 (Third emulation module in an expansion frame)
6482	Slot 4 (Fourth emulation module in an expansion frame)
Telnet connections	
6472	Slot 1 (First emulation module)
6476	Slot 2 (Second emulation module)
6480	Slot 3 (Third emulation module)
6484	Slot 4 (Fourth emulation module)

Write the information here for future reference:

IP Address of Logic Analysis System _____

LAN Name of Logic Analysis System _____

Gateway Address _____

Port Number of Emulation Module _____

To change the port number of an emulation module

Some debuggers do not provide a means to specify a port number. In that case, the debugger will always connect to port 6470 (the first emulation module). If you need to connect to another module, or if the port number of the first module has been changed, you must change the port number to be 6470.

To view or change the port number:

1. Click on the emulation module icon in the system window of the logic analysis system, then select **Update Firmware**.
2. Select **Modify LAN Port...**.
3. If necessary, enter the new port number in the **LAN Port Address** field.

   The new port number must not be 0-1000 and must not already be assigned to another emulation module.

To verify communication with the emulation module

1. telnet to the IP address.

   For example, on a UNIX system, enter "telnet <*IP_address*> 6472". This connection will give you access to the emulation module's built-in terminal interface. You should see a prompt, such as "M>".
2. At the prompt, type:

   ```
   ver
   ```

   You should then see information about the emulation module and firmware version.
3. To exit from this telnet session, type <CTRL>D at the prompt.

See Also

See the online help or manual for your logic analysis system, for information on connecting the system to the LAN and configuring LAN parameters.

See "Problems with the LAN Interface" on page 283, if you have problems verifying LAN communication.

To operate the logic analysis system using a web browser

HP 16600/700A-series logic analysis systems may be monitored and controlled using a web browser.

Information on connectivity is provided in the online help or installation manual for your logic analysis system.

To export the logic analysis system's display to a workstation

By exporting the logic analyzer's display, you can see and use the logic analysis system's windows on the screen of your workstation. To do this, you must have telnet software and X windows installed on your computer.

1. On the workstation, add the host name of the logic analysis system to the list of systems allowed to make connections:

   `xhost +<IP_address>`

2. Use **telnet** to connect to the logic analysis system.

   `telnet <IP_address>`

3. Log in as "hplogic".

   The logic analysis system will open a Session Manager window on your display.

4. In the Session Manager window, click **Start Session on This Display**.

Example

On a UNIX workstation, you could use the following commands to export the display of a logic analysis system named "mylogic":

```
$ xhost +mylogic
$ telnet mylogic
Trying...
Connected to mylogic.mycompany.com.
Escape character is '^]'.
Local flow control on
Telnet TERMINAL-SPEED option ON

HP Logic Analysis System
```

```
Please Log in as: hplogic [displayname:0]

login: hplogic
Connection closed by foreign host.
$
```

To export the logic analysis system's display to a PC

By exporting the logic analyzer's display, you can see and use the logic analysis system's windows on the screen of your PC. To do this, you must have telnet software and an X terminal emulator installed on your computer. The following instructions use the Reflection X emulator from WRQ, running on Windows 95, as an example.

1. On the PC, start the X terminal emulator software.

   To start Reflection X, click the Reflection X Client Start-up icon.

2. Start a **telnet** connection to the logic analysis system.

   Log in as "hplogic".

   For Reflection X, enter the following values in the Reflection X Client Start-up dialog:

   a In the Host field, enter the LAN name or IP address of the logic analysis system.

   b In the User Name field, enter "hplogic".

   c Leave the Password field blank.

   d Leave the Command field blank.

   e Click **Run** to start the connection.

   The logic analysis system will open a Session Manager window on your display.

3. In the Session Manager window, click **Start Session** on This Display.

To enable or disable processor caches

The Power PC 7xx processors have instruction and data caches. Debugging using an third party debugger will have the greatest performance if the caches are disabled during debugging. There are three ways to disable the caches prior to a debug session:

- Set bits 16 and 17 of register HID0 to zero (bit 0 being the MSB). This will turn off the I and D caches. Also turn off the L2 Cache by setting L2CR to zero.

  Ensure that your startup code does not reset the HID0 or L2CR registers as this could re-enable the caches.

- Issue the following probe commands:
  "cf reset=rom"
  "rst" ("rst" will turn off all caches)

  Ensure that your startup code does not reset the HID0 register after the "rst" command as this could re-enable the caches.

- Keep the caches enabled but tell the HP emulation module to bypass them. To do this, issue the probe commands:

  "cf mrdop=phys" (so only physical memory is read)
  "cf dmwrop=bypass" (to bypass the updating of the data cache) reference all addresses with the @dmem modifier.

  Example:
  M> cf mrdop=phys
  M> cf dmwrop=bypass
  M> m -d4 -a4 0.. (this will read physical memory only)
  M> m -d4 -a4 0@dmem=12345678 (this will write physical memory only)

  When caches are bypassed, all memory accesses occur out of physical memory and the cache information is ignored. **This means that cache coherency is not maintained.**

If cache handling is not modified using one of the above three methods, execution with the third party debugger may be slower due to the HP emulation module making sure the cache information stays coherent with physical memory.

Using the Green Hills debugger

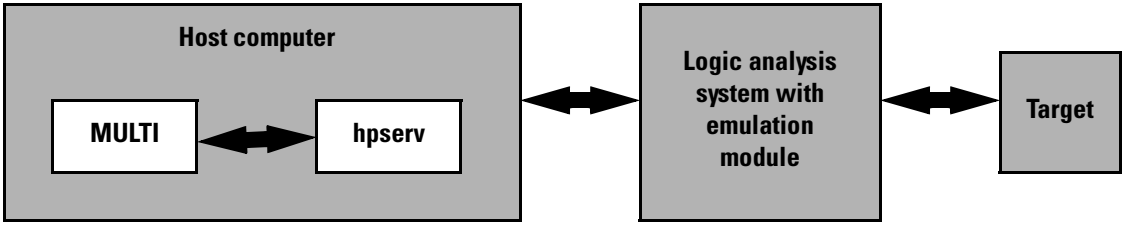
Compatibility

Version 1.8.8.A of the MULTI Development Environment from Green Hills Software, Inc. is one of several debuggers that connect to the HP emulation module.

This section provides information that is specific to using MULTI with the HP emulation module. It is intended to be used in conjunction with the MULTI documentation provided by Green Hills Software.

Overview

MULTI connects to an emulation module through the Green Hills host-resident program (hpserv).



To get started

1 Check that your Emulation Module is programmed with firmware for a PPC7XX processor.

Go to the system window of the logic analyzer interface and verify that the Emulation Module icon is described as a “Motorola PPC700 PowerPC Emulator”. If it is not, follow the instructions on page 166 to update the firmware.

2 Build the executable.

If you have the demo software shipped with the Green Hills debugger, follow these steps:

a Prepare the executable.

Go to the hpdemo subdirectory where you installed MULTI. Copy the mbx700.lnk file to user.lnk.

b Start MULTI.

On Unix, enter "multi".

On Windows, double-click the Green Hills icon.

c Set up the MULTI software environment:

- Replace the project default.bld (in the Builder dialog box next to the project button) with hpdemo/default.bld and press ENTER.
- Make sure the target button on the MULTI window says "PPC".
- In the Builder window, double-click ecs.bld.

The box next to the Debug button should display “ecs”. The window should list the names of the source code files.

d In the Builder menu bar, select **Options→CPU**, then set the processor type.

e In the Builder menu bar, select **Options→Advanced**, and make sure that "Output DWARF on ELF targets" option is enabled.

f Build the demo program:

- In the Builder window, click the Build icon. (Or, in the menu bar, select **Build→Build All**.)
- Close the Progress window when the "Build completed" message is displayed.

3 Connect MULTI to the emulation module.

There are two ways to connect to the emulation module:

- In the Remote box in the MULTI Builder window, enter:
  `hpserv IP_address`

  OR

- In the Builder window, click Debug to open the Debugger window, then in the Debugger window's command pane, enter:
  `remote hpserv IP_address`

Starting hpserv opens two windows: the Target window and the I/O window. Commands entered in the Target window are sent directly to the emulation module.

The I/O window sends input (stdin) to and receives output (stdout) from the target program while it is running.

Note that hpserv connects to the first emulation module (port 6470) in a logic analysis system frame. You may specify another port by using the -p option with hpserv. See page 188 for more information on port numbers.

4 Start the debugger.

If you have not opened the Debugger window yet, click **Debug** in the Builder window.

5 Configure the emulation module and target system.

Before running the target processor, you must configure the HP emulation module for your target system. For example, you may have to set the BDM clock speed, the reset operation, cache disabling, or other configuration parameters.

If you are unsure of the configuration needed for your emulation module, you can use the Configuration window in the logic analysis system's Emulation Control Interface to explore the configuration options.

Once you know the configuration settings needed for your target system, you may use one of the following methods to configure the emulation module and target system:

- Use the Configuration window in the logic analysis system's Emulation Control Interface.
- Enter "cf" commands in the Target window.
- Use an initialization script.

See the "To configure the emulation module and target system using an initialization script" section on page 197 for information on saving the configuration commands in a script.

6 Specify an initialization address for the stack pointer.

This is required if the stack pointer is neither initialized when the processor is reset nor set in the start-up code generated by the compiler. If the stack pointer address needs to be initialized:

- In the debugger's command pane, enter:
  `_INIT_SP = <address>`

OR

- In the Target window, enter:
  `reg r1=<address>`

OR

- Include the following line in an initialization script:
  `target reg r1=<address>`

7 Download the code:

> You can significantly increase download performance by disabling the caches. Disable caches by writing the appropriate bits to the HID0 register (see page 104 or page 122).

- In the Debugger window, select **Remote→LoadProgram**.

  The Debugger command pane indicates that the code has been downloaded to the target.

To configure the emulation module and target using an initialization script

You can use an initialization script to configure the emulation module and set up your target system. If you will always be using the same configuration, this way will save time and reduce errors.

1. Save the configuration commands in a text file, one command per line.

   Green Hills also provides an example initialization sequence in the file MBX800.rc in the "hpdemo" directory.

2. To run the script, enter the following command in the Debugger command pane:

   `<filename`

Example

Create a file with the following lines:

```
remote hpserv hplogic1
 target cf reset=soft
 _INIT_SP=0x10000
```

Save the file in the MULTI start-up directory and name it hpserv.rc. To run the script, enter the following command in the Debugger command pane:

```
<<hpserv.rc
```

When run, this script will:

- Connect to the target through the emulation module in a logic analysis system frame called "hplogic1".
- Set the reset level to soft.
- Initialize the stack pointer.

To perform common debugger tasks

- To display registers, click the **regs** button in the Display window.
- To set a breakpoint, click on the source code line where the breakpoint is to be located.
- To clear a breakpoint, click again on the source line.
- To step through code, click **next**.
- To run from the current PC, click **go**.
- To toggle the display between source code and source code interlaced with assembly code, click **assem**.
- To load program symbols, reset the PC, reset the stack pointer, and run from the start, click **restart**.

To send commands to the emulation module

MULTI communicates to the emulation module using the emulation module's "terminal interface" commands. MULTI automatically generates and sends the commands required for normal operation. If you want to communicate directly with the emulation module during a debug session, you may do so using "terminal interface" commands through the Target window (which comes up when hpserv is brought up). You can also enter these commands from the Debugger window's command pane by preceding the command with the "target" command.

To view commands sent by MULTI to the emulation module

The communication between MULTI and the emulation module can be viewed by running hpserv in a logging mode:

```
remote hpserv -dc -a  -o <filename> <emulation module name>
```

The options -dc and -da log both asynchronous and console messages and the -o <*filename*> directs these messages to a log file called <*filename*>. When using this option, disconnect from hpserv (to flush out the file) and then you may view <*filename*> to see what commands MULTI sent to the emulation module.

NOTE: logging commands in this way may result in a VERY large file. Beware of the disk space it may require.

To reinitialize the system

If you suspect that the emulation module is out of sync with the MULTI debugger, you may want to reinitialize it. Perform the steps below to accomplish reinitialization:

1. In the Target window, type:

   ```
   init -c
   ```

2. Repeat steps 5 through 8 in the "Getting started" section to configure the emulation module.

To disconnect from the emulation module

- In the Debugger window, select Remote→**Disconnect**.

  The Debugger command pane indicates that the debugger has disconnected from the emulation module.

Error conditions

"!ERROR 800! Invalid command: bcast" usually means that there is not a target interface module (TIM) connected to the emulation module or the emulation module does not have firmware for the PPC700 family. Verify that the emulation module is connected to the target. Next, go to the system window of the logic analyzer interface and verify that the Emulation Module icon (stop-light) is described as a Motorola PPC700 PowerPC Emulator. If it is not, follow the steps on page 166 to update the firmware in the Emulation module for PPC700 processors.

"command socket connection failed: WSAECONNREFUSED: connection refused" usually means the emulation module is not at port #6470 on the Logic Analysis System.

See Also

See also the *Green Hills MULTI Software Development Environment User's Guide*.

See also *Using MULTI with the Hewlett-Packard Processor Probe* from Green Hills Software, Inc.

See also the Green Hills web site: **http://www.ghs.com**

See Chapter 7, "Connecting and Configuring the Emulation Module," beginning on page 149 for more information on configuration options and the "cf" command.

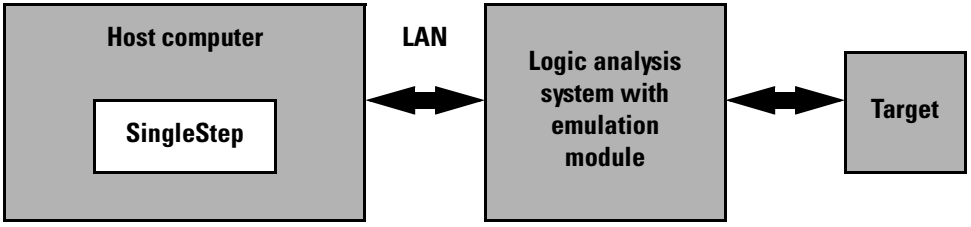
Using the Software Development Systems debugger

Compatibility

Version 7.2 of the SingleStep debugger from Software Development Systems, Inc. is one of several debuggers which connect to the HP emulation module.

This section provides information that is specific to using SingleStep with the HP emulation module. It is intended to be used in conjunction with the SingleStep documentation provided by SDS.

Overview



Startup behavior

The following actions are performed at the start of a session and when you select **File→Debug:**

- If the target reset option is selected, the target is reset and programmed with the register values in the configuration file (<*filename*>.cfg).
- Hardware breakpoints are disabled.
- Software breakpoints are enabled.
- All breakpoints are cleared.
- main() _exit breakpoints are set, if that option is selected.

Getting started

1 Check that your Emulation Module is programmed with firmware for a PPC700 processor:

Go to the system window of the logic analyzer interface and verify that the Emulation Module icon is described as a "PowerPC 700 Emulator". If it is not, follow the instructions on page 166 to update the firmware.

2 Connect to the emulation module:

a Start SingleStep running on your PC or workstation.

b When the small Debug dialog box appears in the middle of the screen, click the Connection tab and then enter the IP address of the HP logic analysis system which contains the emulation module.

If the Debug dialog box is not visible, select **File→Debug.**

Note: SingleStep is hard-coded to connect to the emulation module at port 6470 of the logic analysis system frame. See page 189 for more information on port numbers.

3 Initialize the target system.

The target system must have various registers and memory locations initialized before it can access RAM and before SingleStep can download an application. Normally, code in the target's boot ROM performs this initialization. However, when SingleStep resets the target, it immediately places the processor in debug mode. Any initialization code which may exist on the target board has not been run.

SingleStep provides a way for target initialization to occur without running application code through the use of the "_config" alias. _config is used to define a list of commands that will be used to initialize the target after a reset. The _config alias should be defined in the sstep.ini file (in the "cmd" directory); it can list the actual commands used to initialize the executable, or it may point to a file of type .cfg which contains the actual initialization commands. The config aliases contained in sstrp.ini are provided by SDS for some common targets. Refer to the SDS SingleStep User's Guide for information about additional initialization techniques.

An alternate way of creating the _config alias is to use the Target Configuration tab in the "Debug" dialog box. The "Debug" dialog method and the sstep.ini method are mutually exclusive. Use one or the other, but not both.

Initialization of the target (that is, execution of the _config alias) will not actually occur until the "Debug" dialog is successfully exited.

4 Set up the download and execution options in the Options tab of the Debug dialog.

5 Download the application and run:

> You can significantly increase download performance by disabling the caches. Disable caches by writing the appropriate bits to the HIDO register (see page 104 or page 122).

Select the File tab and enter the application file name. Exit the "Debug" dialog box by clicking OK.

Emulation module initialization and target initialization occur every time the "Debug" dialog is terminated via the OK button. A summary of the actions taken by SingleStep is given here:

- Initialize the emulation module with the communication speed specified in the "Debug" dialog.
- If "reset target" was selected then execute the commands specified by the _reset alias. The _reset alias should be used to specify commands that are specific to initializing the processor. It is executed each time the processor is reset. The value of the _reset alias can be viewed by issuing a "alias _reset" from the command window.
- Execute the commands specified by the _config alias. The _config alias should be used to specify commands that are specific to initializing (configuring) the target system. It is executed each time the processor is reset and each time the debug dialog is exited. The value of the _config alias can be viewed by issuing an "alias _config" from the command window.
- If "load image" was selected then download the application and set the PC based on object module file contents.
- If "execute until main" was selected then set a breakpoint at main() and run.

To send commands to the emulation module

To view commands sent by SingleStep

SingleStep communicates to the emulation module using the emulation module's "terminal interface" commands. SingleStep automatically generates and sends the commands required for normal operation. This communication between SingleStep and the emulation module can be observed by entering the following command in the SingleStep command window:

```
control -ms
```

To send commands

"Terminal interface" commands may be sent directly to the emulation module from the SingleStep command window or included in SingleStep's .cfg or .dbg command files.

Commands should be enclosed in double quotes and given the prefix: control -c.

Examples

To see see what is defined to happen at reset you would issue the following command in the SingleStep command window:

```
control -c "cf reset"
```

To change the reset definition you would issue the following command in the command window:

```
control -c "cf reset=runrom"
```

For more information about "terminal interface" commands see page 168.

Error conditions

"!ERROR 800! Invalid command: bcast" usually means that there is not a target interface module (TIM) connected to the emulation module or the emulation module does not have firmware for the MPC700 family. Verify that the emulation module is connected to the target through a TIM. Next, go to the system window of the logic analyzer interface and verify that the Emulation Module icon (stop-light) is described as a Motorola MPC700 PowerPC Emulator. If it is not, follow the steps on page 166 to update the firmware in the Emulation module for MPC700 processors.

"command socket connection failed: WSAECONNREFUSED: connection refused" usually means the emulation module is not at port #6470 on the Logic Analysis System. See step 2 of the getting started section above.

"unrecognized hostname" usually means that the debugger is unable to establish communication with the emulator. Verify communication to the emulation module by doing a ping to the logic analyzer. If you are unable to ping the logic analyzer refer to page 283 for more information.

See Also

The SDS web site: **http://www.sdsi.com**

The *SDS SingleStep Users Guide*.

The configuration section beginning on page 168 for more information on configuration options and the "cf" command.

9

Coordinating Logic Analysis with Processor Execution

This chapter describes how to use an analysis probe, an emulation module, and other features of your HP 16600A or HP 16700A logic analysis system to gain insight into your target system.

What are some of the tools I can use?

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

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

Which assembly-level listing should I use?

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

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

Which source-level listing should I use?

Different tools display source code for different uses:

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

Where can I find practical examples of measurements?

The Measurement Examples section in the online help contains examples of measurements which will save you time throughout the phases of system development: hardware turn-on, firmware development, software development, and system integration.

A few of the many things you can learn from the measurement examples are:

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

To find the measurement examples, click on the Help icon in the logic analysis system window, then click on "Measurement Examples."

Triggering the Emulation Module from the Analyzer

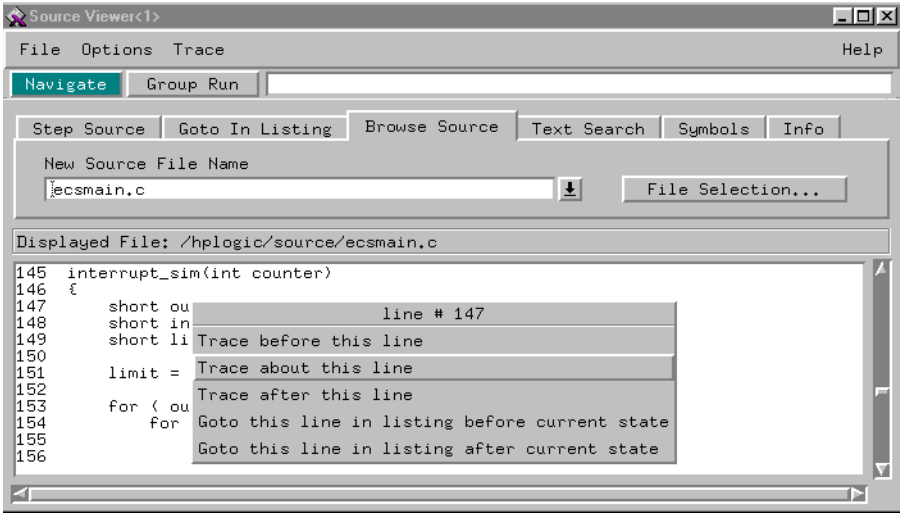
The logic analyzer may be used to signal the emulation module to stop (break) the target processor. This is done from either the Source Viewer window or the Intermodule window. If you are using the HP B4620B Source Correlation Tool Set, using the Source Viewer window is the easiest method.

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

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

1. Click on the logic analyzer module icon in the System window, and choose **Source Viewer...**.
2. In the Source Viewer window, click on the line of source code where you want to set the trigger, then select **Trace about this line**.

The logic analyzer trigger is now set.



3 Select **Trace→Enable - Break Emulator On Trigger.**

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

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

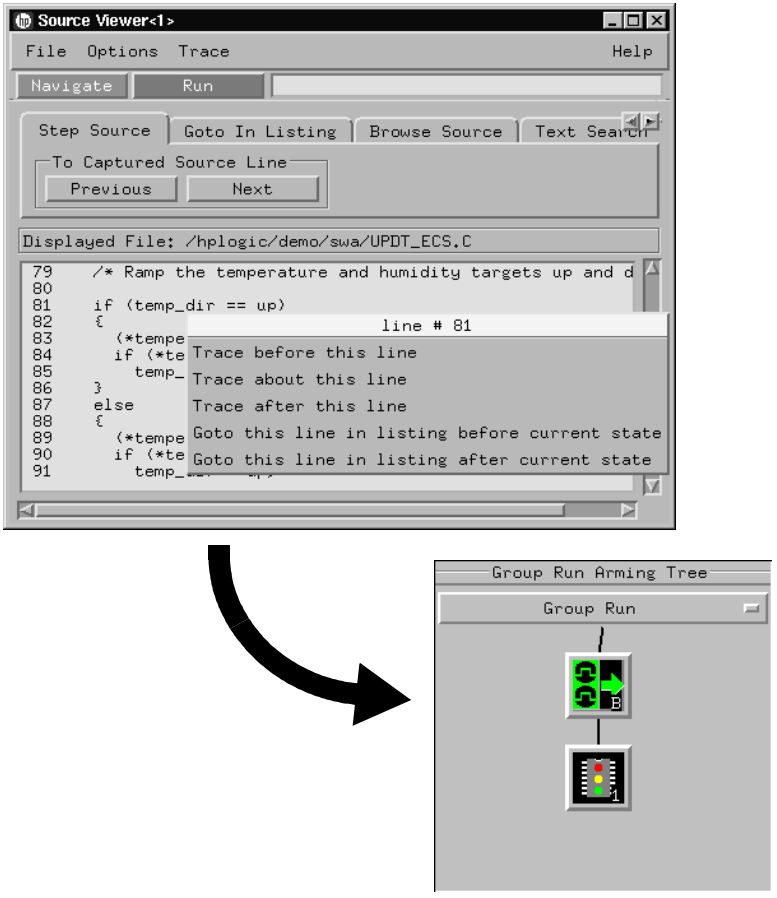
4 Click **Group Run** in the Source window (or other logic analyzer window).

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

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

Use the Intermodule window if you do not have the HP B4620B Source Correlation Tool Set or if you need to use a more sophisticated trigger than is possible in the Source Viewer window.

1. Create a logic analyzer trigger.
2. Click on the **Intermodule** icon in the System window.
3. In the Intermodule window, click the emulation module icon, then select the analyzer which is intended to trigger it.



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

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

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

See Also

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

To minimize the "skid" effect

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

To minimize the skid effect:

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

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

The amount of skid will depend on the processor's execution speed and whether code is executing from the cache. See "To configure the JTAG clock speed (communication speed)" on page 174 for information on how to configure the clock speed.

To stop the analyzer and view a measurement

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

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

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

1. The analyzer triggers.
2. The trigger ("Break In") is sent to the emulation module.
3. The emulation module stops the user program which is running on the target processor. The processor enters a background debug monitor.
4. Because the processor has stopped, the analyzer stops receiving a qualified clock signal.
5. If the trigger position is "End", the measurement will be completed.
6. If the trigger position is not "End", the analyzer may continue waiting for more states.
7. The user clicks **Stop** in a logic analyzer window, which tells the logic analyzer to stop waiting, and to display the trace.

Tracing until the processor halts

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

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

Some possible uses for this measurement are:

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

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

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

To capture a trace before the processor halts

HP 16600A/16700A

1 Set the sampling to **state mode** and the trigger condition to **Run until user stop**.

Now proceed to step 2 under "All HP logic analysis systems".

HP 1660/70 **HP 16500B/C**

1 In the configuration dialog, set the machine type to **state**, and set the logic analyzer to trigger on **nostate**.

Now proceed to step 2 under "All HP logic analysis systems".

All HP logic analysis systems

2 Set the trigger point (position) to **End**.

3 In a logic analyzer window, click **Run**.

4 In the Emulation Control Interface or debugger click **Run**.

5 When the target processor halts, click **Stop** in the logic analyzer window to complete the measurement.

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

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

Triggering the Logic Analyzer from the Emulation Module

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

If you are only using a state analyzer to capture the processor bus then it will be much simpler to use “Tracing until processor halts” as described on page 216.

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

The emulation module trigger signal

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

The In Monitor trigger signal can be caused by:

- The most common method to generate the signal is to click **Run** and then click **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 click **Reset** and then click **Break**. Going from the reset state of the processor to the In Monitor state will generate the signal. Some processors that do not remain in reset will not generate an In Monitor signal in the reset to break transision.
- In addition, an In Monitor signal is generated any time a debugger or other user interface reads a register, reads memory, sets breakpoints or steps. Care must be taken to not falsely trigger the logic analyzers that are listening to the In Monitor signal.

Group Run

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

The following setups can operate independently of Group Run:

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

Here are some examples:

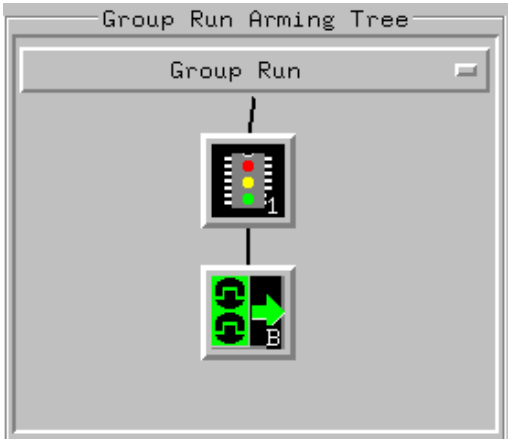
- If Group Run is armed from Port In and an emulation module is connected to Group Run, then any Port In signal will cause the emulation module to go into monitor. The Group Run button does not have to be clicked for this to operate.
- If two emulation modules are connected together so that one triggers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

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

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

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

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



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

Debuggers can cause triggers

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

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

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

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

If you define a trigger on some state and the debugger happens to read the same state, then you may falsely trigger your analyzer measurement.

In summary, when you are making an analysis measurement be aware that the debugger could be impacting your measurement.

To trigger the analyzer when the processor halts - timing mode

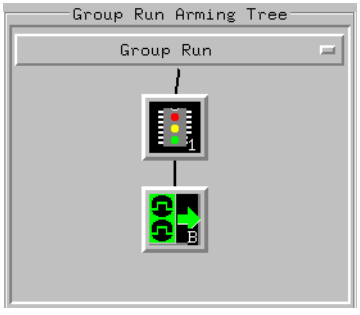
If your processor halts unexpectedly, and you would like to see timing information on your bus prior to the halt, set up this measurement.

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

NOTE: If you only need state information leading up to a processor halt, and timing information is not important, use the procedure called "To capture a trace before the processor halts" on page 216. It is much simpler.

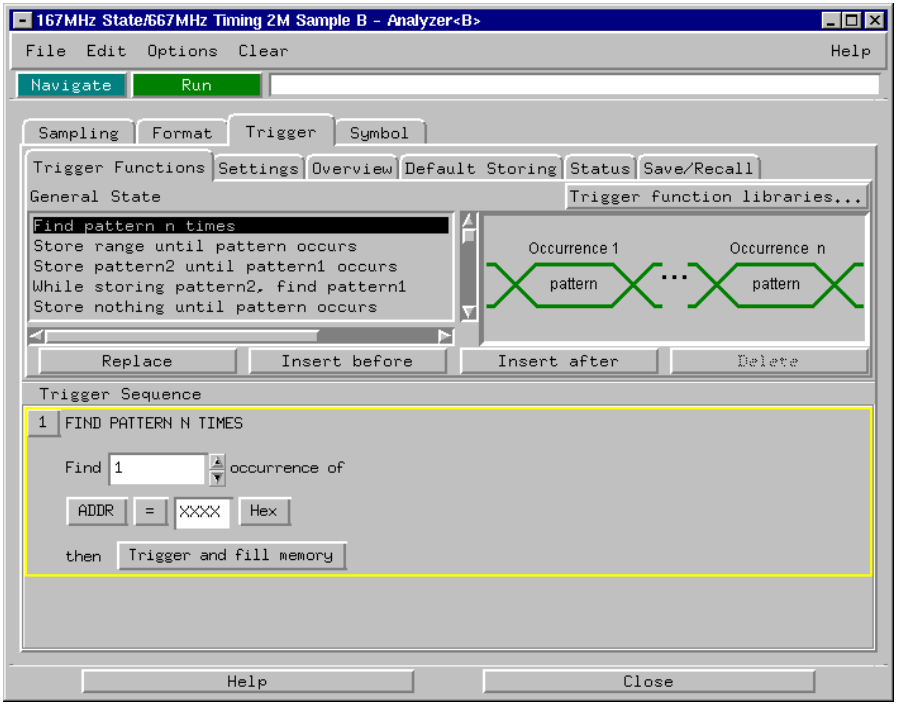
HP 16600A/16700A with VisiTrigger

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



Now continue to step 2.

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



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

4 Click **Group Run** to start the analyzer(s).

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

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

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

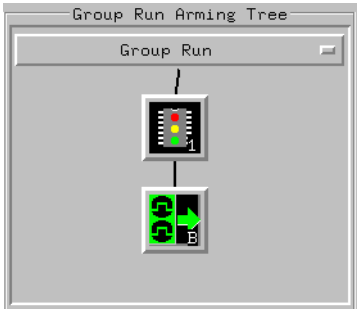
The logic analyzer will store states until the processor halts.

To trigger the analyzer when the processor reaches a breakpoint

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

NOTE: If you are only using a state analyzer to capture the processor bus then it will be much simpler to use "Tracing until processor halts" as described on page 216.

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



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

4. Set the breakpoint.

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

5. Click **Group Run** to start the analyzer(s).

6 Click **Run** in the Emulation Control Interface or use your debugger to start the target processor running.

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

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

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

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

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

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

10

Hardware Reference

This chapter contains additional reference information including the specifications and characteristics for the target system when using the inverse assembler software, specifications and characteristics for the emulation probe, and signal mapping for HP E2498A software. It consists of the following information:

- Inverse assembler reference
- Emulation module reference

Operating characteristics

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

Operating Characteristics	
Microprocessor Compatibility	PowerPC 740, 745, 750, 755
Microprocessor Clock Speed	70 MHz for the HP 1670A and HP 16554A Logic Analyzers 100 MHz for the HP 16600A, HP 16601A, HP 16550A, HP 16554D, HP 16555/56/57, HP 1660, and HP 1670D Logic Analyzers
Logic Analyzers Supported	HP 1660A/AS/C/CP/CS, HP 1670A/D, HP 16550A (two cards), HP 16554A/55A/56A (two or three cards), HP 16555D/56D/57D (two or three cards), HP 16600A, HP 16601A.
Probes Required	Eight 16-channel probes are required for disassembly. Two additional 16-channel probes are available.
Signal Line Loading	Typically 100 kOhm plus 10 pF.
Setup/Hold Requirement	For all signals, the logic analyzers require a minimum combined setup/hold window. For the HP 16600-series logic analysis system, the combined setup/hold must be at least 4.5 ns (such as 0/4.5, 1.0/3.5. etc). For all other logic analyzers, the combined window must be at least 3.5 ns.

Inverse assembler—signal-to-connector mapping

The following tables show the electrical signal-to-connector mapping required by the HP E2498A Inverse Assembler Software.

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

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

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

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

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

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

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

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

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

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

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

Emulation module—operating characteristics

The following operating characteristics are not specifications, but are typical operating characteristics for the HP 16610A emulation module and PPC7XX target interface module.

Operating Characteristics	
Microprocessor Compatibility	PPC740 and PPC750 microprocessors.
Environmental Characteristics (Temperature, Altitude, Humidity, Pollution Degree)	The emulation module meets the environmental characteristics of the logic analysis system in which it is installed. For indoor use only.

Emulation module—electrical characteristics

NOTE: The emulation module is compatible with the PowerPC 740/750, but not the PowerPC 745/755.

Maximum Ratings

Characteristics for the PowerPC 740/750 emulation module	Symbol	Min	Max
TDO, $\overline{\text{CKSTP\_OUT}}$	V_{ih}	2.0 V	5.5 V
	V_{il}		0.8 V
	I_i		±1 μA
	C_{in}		15 pF
TDI, TCK, TMS, $\overline{\text{TRST}}$ [1]	V_{oh} @ I_{oh} = -32 mA	2.0 V	2.8 V
	V_{ol} @ I_{ol} = 64 mA; V_{CC} = 4.5V		0.55 V
TDI, TMS, $\overline{\text{TRST}}$	C_o		25 pF
TCK	C_o		45 pF
+3.3V Power Sense [2]	V_{ih}	2.0 V	5.3 V
	V_{il}	-0.3 V	0.8 V
$\overline{\text{SRESET}}$, $\overline{\text{HRESET}}$ [3]	V_{ol} @ I_{ol} = 12 mA		0.5 V
	C_o		25 pF
TS0 - TS6, SYSCLK	C_{in}		0 pF
	V_{ih}	2.0 V	5.5 V
	V_{il}		0.8 V
	I_i		±1 μA

[1] These signals must not be actively driven by the target system when the debug-port is being used.

[2] Power Sense is used only to determine target powered status. The emulation module does not draw power from this source.

[3] Open collector outputs, pulled up to a generated voltage equivalent to the Power Sense voltage with a 2.61 k ohm pullup resistor

11

General-Purpose ASCII (GPA) Symbol File Format

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

If your compiler is not one of those listed on page 139, if your compiler does not include symbol information in the output, or if you want to define a symbol not in the object file, you can create an ASCII format symbol file.

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

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

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

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

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

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

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

Example

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

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

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

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

GPA Record Format Summary

Format

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

 [FUNCTIONS]
 func_name  start..end

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

 [SOURCE LINES]
 File: file_name
 line#  address

 [START ADDRESS]
 address

 #Comments
```

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

Example

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

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

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

 [VARIABLES]
 total    40002000  4
 value    40008000  4

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

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

SECTIONS

Format

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

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

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

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

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

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

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

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

Enable Section Relocation

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

Example

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

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

FUNCTIONS

Format

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

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

`func_name` A symbol representing the function name.

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

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

Example

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

VARIABLES

Format

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

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

`var_name` A symbol representing the variable name.

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

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

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

Example

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

SOURCE LINES

Format

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

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

`file_name` The name of a file.

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

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

Example

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

START ADDRESS

Format

```
[START ADDRESS]
 address
```

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

Example

```
[START ADDRESS]
 00001000
```

Comments

Format

```
#comment text
```

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

Example

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

12

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 Hewlett-Packard 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 256 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 analysis probe and target system.

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 airflow that meet or exceed the requirements of the microprocessor manufacturer.

Capacitive loading

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

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

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

Inverse Assembler Problems

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

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

No inverse assembly or incorrect inverse assembly

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

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

  Refer to page 94 to select the correct processor type.

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

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

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

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

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

not change the names of these labels or the bit assignments within the labels. Some analysis probes also require other data labels. See“Configuring the Logic Analysis System” on page 67 for more information.

- ❏ Verify that memory managers have been disabled.

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

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

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

  For instructions on how to disable the cache, see page 104 or page 122.

  If the cache is enabled, follow the instructions in “Using Cache-On Trace Reconstruction” on page 87.

  If the cache is enabled, ensure that an S-Record is loaded and that the preferences are set correctly (see page 96.)

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

- ❏ Verify that the endian selection is correct in the processor options preferences menu (see page 94).

  Unlike most processors, the PPC7XX can run in either little-endian or big-endian mode. To verify the format of the data, the Machine State Register (MSR) must be examined. The least significant bit (bit 0 according to Motorola convention, or bit 31 according to IBM convention) indicates the mode of the processor. A value of 1 indicates little-endian mode. A value of 0 indicates big-endian mode.

Inverse assembler will not load or run

You need to ensure that you have the correct system software loaded on your analyzer.

- ❏ Ensure that the inverse assembler is on the same disk as the configuration files you are loading.

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

  See Chapter 2, “Installing Software,” beginning on page 33 for details.

Intermodule Measurement Problems

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

An event wasn't captured by one of the modules

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

❏ Adjust the skew in the Intermodule menu.

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

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

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

Messages

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

". . . Inverse Assembler Not Found"

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

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

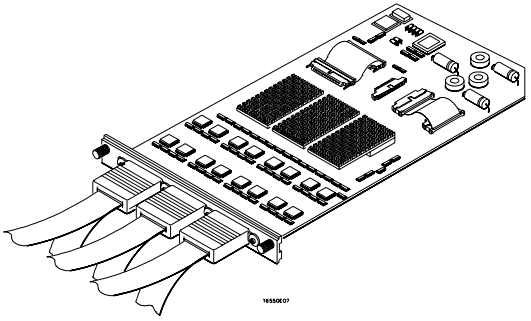
- For HP 16600A/700A-series logic analysis systems it should be in /hplogic/ia.
- For other logic analyzers it should be in the same directory as the configuration file.

See Chapter 2, "Installing Software," beginning on page 33 for details.

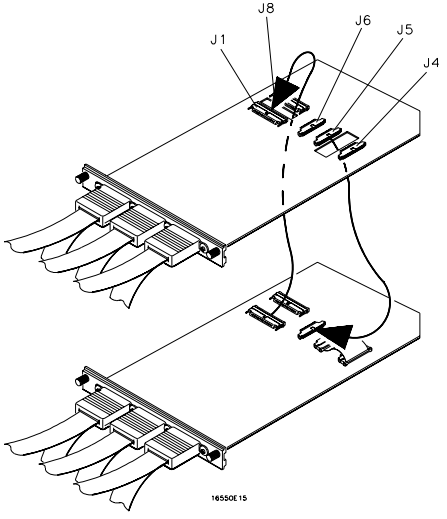
"Measurement Initialization Error"

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

Cable Connections for One-Card HP 16550A Installations



Cable Connections for Two-Card HP 16550A Installations



See Also

The *HP 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 {filename} in the disk operation menu. Selecting Load {All} will cause incorrect operation when loading most analysis probe interface configuration files.

See Also

See Chapter 2, "Installing Software," on page 33 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 analysis probe interface.

See Chapter 3 to determine the proper connections.

"Time from Arm Greater Than 41.93 ms"

The HP 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.

To Obtain Replacement Parts

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

Replaceable Parts

HP Part Number	Description
E2498-68702	Configuration and Inverse Assembler Software

13

Troubleshooting the Emulation Module

If you have problems with the emulation module, your first task is to determine the source of the problem. Problems may originate in any of the following places:

- The connection between the emulation module and your debugger
- The emulation module itself
- The connection between the emulation module and the target interface module
- The connection between the target interface module and the target system
- The target system

You can use several means to determine the source of the problem:

- The troubleshooting guide on the next page
- The status lights on the emulation module
- The emulation module "performance verification" tests
- The emulation module's built-in "terminal interface" commands

Emulation Module Troubleshooting Guide

Common problems and what to do about them

Symptom	What to do	See also
Commands from the Emulation Control Interface have no effect	Check that you are using the correct firmware.	
Commands from debugger have no effect	Use the Emulation Control Interface to try a few built-in commands. If this works, your debugger may not be configured properly. If this does not work, continue with the steps for the next symptom....	page 30, page 271
Emulation module built-in commands do not work	**1** Check that the emulation module has been properly configured for your target system.	page 30, page 168
	2 Run the emulation module performance verification tests.	page 285
	3 If the performance verification tests pass, then there is an electrical problem with the connection to the target processor OR the target system may not have been designed according to "Designing a Target System."	page 154, page 274
"Slow or missing clock" message after a logic analyzer run	Check that the target system is running user code or is in reset. (This message can appear if the processor is in background mode.)	
"Slow clock" message in the Emulation Control Interface or "c>" prompt in the built-in terminal interface	Check that the clock rate is properly configured.	page 174
Some commands fail	Check the "restrict to real-time runs" configuration	page 173

Emulation Module Status Lights

The emulation module uses status lights to communicate various modes and error conditions.

The following table gives more information about the meaning of the power and target status lights.

❍ = LED is off
● = LED is on
✳ = Not applicable (LED is off or on)

Power/Target Status Lights

Pwr/Target LEDs	Meaning
❍ Reset ❍ Break ❍ Run	No target system power, or emulation module is not connected to the target system
● Reset ❍ Break ❍ Run	Target system is in a reset state
❍ Reset ● Break ❍ Run	The target processor is executing in Debug Mode
❍ Reset ❍ Break ● Run	The target processor is executing user code
❍ Reset ● Break ● Run	Only boot firmware is good (other firmware has been corrupted)

Emulation Module Built-in Commands

The emulation module has some built-in "terminal interface" commands which you can use for troubleshooting.

You can access the terminal interface using:

- A telnet (LAN) connection
- The Command Line window in the Emulation Control Interface
- A "debugger command" window in your debugger

To telnet to the emulation module

You can establish a telnet connection to the emulation module if:

- A host computer and the logic analysis system are both connected to a local-area network (LAN), and
- The host computer has the telnet program (often part of the operating system or an internet software package).

To establish a telnet connection:

1. Find out the port number of the emulation module.

   The default port number of the first emulation module in an HP 16600A/700A series logic analysis system is 6472. The default port of a second module in an HP 16700A-series system is 6476. The default port numbers of a third and fourth module in an expansion frame are 6480 and 6484. These port numbers can be changed, but that is rarely necessary.

2. Find out the LAN address or LAN name of the logic analysis system.

3. Start the telnet program.

   If the LAN name of the logic analysis system is "test2" and you have only one emulation module installed, the command might look like this:

   ```
   telnet test2 6472
   ```

4. If you do not see a prompt, press the <Return> key a few times.

   To exit from this telnet session, type <CTRL>D at the prompt.

To use the built-in commands

Here are a few commonly used built-in commands:

Commonly used built-in commands	
b	Break—go into the background monitor state
cf	Configuration—read or write configuration options
help	Help—display online help for built-in commands
init	Initialize—init -c re-initializes everything in the emulation module except for the LAN software; init -p is the equivalent of cycling power (it will break LAN connections)
lan	configure LAN address (emulation probes only)
m	Memory—read or write memory
reg	Register—read or write a register
r	Run—start running user code
rep	Repeat—repeat a command or group of commands
rst	Reset—reset the target processor (the emulation module will wait for you to press the target's RESET button)
s	Step—do a low-level single step
ver	Version—display the product number and firmware version of the emulation module

The prompt indicates the status of the emulation module:

Emulation module prompts	
U	Running user program
M	Running in background monitor
c	Target is checkstopped
p	No target power
d	No target interface module connected to emulation module
?	Unknown state

Examples

To set register GPR0, then view GPR0 to verify that it was set, enter:

```
R>rst -m
M>reg GPR0=ffff
M>reg GPR0
  reg GPR0=0000ffff
```

To break execution then step a single instruction, enter:

```
M>b
M>s
  PC=xxxxxxxx
M>
```

To determine what firmware version is installed in the emulation module, enter:

```
M>ver
```

See Also

Use the `help` command for more information on these and other commands. Note that some of commands listed in the help screens are generic commands for HP emulators and may not be available for your product.

If you are writing your own debugger, contact HP for more information.

Problems with the Target System

This section describes how to determine whether your target system is causing problems with the operation of the emulation module.

What to check first

1 Try some basic built-in commands using the Command Line window or a telnet connection:

```
U>rst
R>
```

This should reset the target and display an "R>" prompt.

```
R>b
M>
```

This should stop the target and display an "M>" prompt.

```
M>reg GPR1
   reg GPR1=00000000
M>
```

This should read the value of the GPR1 register (the value will probably be different on your target system).

```
M>m 0..
   00000000   7c3043a6 7c2802a6 7c3143a6 4bf04111
   00000010   00000000 00000000 00000000 00000000
   00000020   00000000 00000000 00000000 00000000
   00000030   00000000 00000000 00000000 00000000
   00000040   00000000 00000000 00000000 00000000
   00000050   00000000 00000000 00000000 00000000
   00000060   00000000 00000000 00000000 00000000
   00000070   00000000 00000000 00000000 00000000
M>
```

This should display memory values starting at address 0.

```
M>s
```

This should execute one instruction at the current program counter.

If any of these commands don't work, there may be a problem with the design of your target system, a problem with the revision of the processor you are

using, or a problem with the configuration of the emulation module.

2 Check that the emulation module firmware matches your processor. To do this, enter:

```
M>ver
```

See Also See page 271 for information on entering built-in commands.

To check the debug port connector signals

- Check for the following logic levels on the target debug port.

Levels with the emulation module not connected

Header Pin	Signal Name	Level
3	TDI	Low
4	$\overline{\text{TRST}}$	High
6	+POWER	V_{DD}
7	TCK	High
9	TMS	High
11	$\overline{\text{SRESET}}$	High
13	$\overline{\text{HRESET}}$	High
15	$\overline{\text{CHECKSTOP}}$	High
16	GND	Low

Levels with the emulation module connected

Header Pin	Signal Name	I/O
1	TDO	Toggle with "es" command
3	TDI	Toggle with "es" command
4	$\overline{\text{TRST}}$	Low pulse with "rst" command
6	+POWER	V_{DD}
7	TCK	10+ MHz clock (default)
9	TMS	Low, pulse with "es" command
11	$\overline{\text{SRESET}}$	High, pulse low with "rst" command
13	$\overline{\text{HRESET}}$	High, pulse low with "rst" command
15	$\overline{\text{CHECKSTOP}}$	High
16	GND	Low

To interpret the initial prompt

The initial prompt can be used to diagnose several common problems. To get the most information from the prompt, follow this procedure:

1. Connect the emulation module to your target system.
2. Set the default configuration settings. Enter:

```
M>init -c
```

You can enter this command at any prompt. The emulation module will respond with the same information as printed by the "ver" command.

If the response is "!ERROR 905! Driver firmware is incompatible with ID of attached device"

Make sure the target interface module is connected to the cable of the emulation module, then try the "init -c" command again.

If the initial prompt is "p>"

Check pin 6 on header, 3.3V (V_{DD}).

If the initial prompt is "M>"

The processor entered debug mode without the help of the emulation module. Is another debugger connected?

If the initial prompt is "c>"

Processor is checkstopped. Something caused a machine exception before the emulation module connected or $\overline{\text{CHECKSTOP}}$ is being pulled or held low.

If the initial prompt is "?>" with "ERROR 171!"

A bad status code (0xXX) was received from the processor. Valid status is 0x01 or 0x05. Any other status indicates a bad scan of the instruction register. Check TCK, TDO, TDI, TMS, and $\overline{\text{TRST}}$_L signals. Check the firmware revision.

If the initial prompt is "U>"

The emulation module is scanning the instruction register correctly. Now you can do some more tests:

3 Enter the reset command:

```
U>rst
U>
```

The "U>" prompt is a good response that indicates $\overline{\text{SRESET}}$ and $\overline{\text{HRESET}}$ are working. Continue with "If the prompt after rst is U>".

If the prompt after rst is "?>" with "ERROR 171!"

A bad status code (0xXX) was received from the processor. Valid status is 0x01, any other status indicates bad scan of IR or failure of the reset signals. Verify TCK, TDO, TDI,TMS, and $\overline{\text{TRST}}$ are all changing state on an $\overline{\text{HRESET}}$.

If the rst command fails

Set "cf reset=rom" (no external bus cycles used in this mode), then enter the "rst" command again:

```
*>cf reset=rom
*>rst
M>
```

You can enter these commands at any prompt, shown here as "*>".

- If the prompt is "M>" with no error messages, all scans worked. We have control as long as we don't try to run code. Continue with "If you can get to the "M>" prompt.
- If an error message is displayed, verify that $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ are being driven.
- If the prompt is "c>", there was bad scanning of the data scan chain. Check processor mask revision.
- If the prompt is "U>", the processor failed to stop soft or hard. Check reset lines, mask revision, processor type and firmware version.

If the prompt after rst is "U>"

The $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ lines are working. Continue with more tests:

4 Enter the break command:

```
U>b
M>
```

If the prompt after b is "M>" with error messages

If you see: "!ERROR 145! Unable to soft stop - freezing the processor clocks" the processor is hard stopped. Check the mask revision, processor type, and firmware version. If all of these look good, then the target may not be terminating cycles (pending external bus cycles). Successive run ("r") and step ("s") commands will fail. The processor may have fetched an invalid instruction.

Check the value of the PC (IAR):

```
M>reg PC
  reg PC=xxxxxxxx
M>
```

If the value is fff00100, the processor had a problem accessing the boot ROM and crashed during boot.

Processor and/or board level reset is required to recover from "freezing" processor clocks" -- register and memory commands should still work.

If the prompt after b is "M>" with no error messages

Everything is still working correctly. Continue with more tests:

If you can get to the "M>" prompt

5 At the "M>" prompt , check register and memory access:

```
M>reg GPR0
  reg GPR0=xxxxxxxx
M>reg GPR0=12345678
M>reg GPR0
  reg GPR0=12345678
M>
```

6 If the returned value is equal to the written value, then the dd level of the chip is probably correct.

Now enter:

```
M>m -d4 -a4 0=11111111,22222222,33333333,44444444
M>m -d4 -a4 0..
   00000000   11111111 22222222 33333333 44444444
   00000010   00000000 00000000 00000000 00000000
   00000020   00000000 00000000 00000000 00000000
   00000030   00000000 00000000 00000000 00000000
   00000040   00000000 00000000 00000000 00000000
   00000050   00000000 00000000 00000000 00000000
   00000060   00000000 00000000 00000000 00000000
   00000070   00000000 00000000 00000000 00000000
M>
```

- Returned value is equal to the written value implies that memory is working.
- Returned value is not equal to the written value implies that memory control may not be initialized. Try to initialize by:

```
M>cf reset=runrom;rst;w 5
  #waiting for 5 seconds...
U>b
M>
```

a Repeat above memory test.

7 At the "M>" prompt , check the processor's revision level:

The target must support burst cache fill from where PC is pointing.

Set the PC to a location in RAM. For example:

```
M>reg PC=100
M>
```

Now enter:

```
M>reg PVR
  reg PVR=xxxxxxxx
M>
```

The returned value is in the form VVVVRRrr where VVVV is the processor's design architecture family, and RRrr is mask revision level.

VVVV:

0008 -> 740/750/745/755

For example reg PVR=00080202 means 740/750 Mask Revision 2.2.
and reg PVR=00083100 means 745/755 Mask Revision 1.0.

If you see memory-related problems

1 Set caches and translation off:

```
M>reg HID0=0
M>reg MSR=0
M>
```

If these commands fail, just try again.

2 Now enter:

```
M>m -d4 -a4 0=11111111,22222222,33333333,44444444
M>m -d4 -a4 0..
   00000000   11111111 02222222 33333333 44444444
   00000010   00000000 00000000 00000000 00000000
   00000020   00000000 00000000 00000000 00000000
   00000030   00000000 00000000 00000000 00000000
   00000040   00000000 00000000 00000000 00000000
   00000050   00000000 00000000 00000000 00000000
   00000060   00000000 00000000 00000000 00000000
   00000070   00000000 00000000 00000000 00000000
M>
```

- If you do not see correct values written in memory, try increasing memory delay (page 175).
- If the read value is not equal to the written value, the memory controller may not be set up correctly.
- If the read value is equal to the written value, but you still suspect memory problems, the emulator firmware might not be working with cache.

3 Enter:

```
M>cf reset=rom
M>rst
M>m -d4 -a4 0..
```

- Read value not equal to the written value implies that reset is tied to memory controller. Check $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ for correct connections.

4 If you have memory problems running Windows NT, you may have this problem:

- System normally runs in little endian mode
- "rst" returns processor to big endian, memory controller on target still little endian, so memory access doesn't work.

5 Hand load a little program:

```
M>m -d4 -a4 100=38210001,60000000,60000000,4bfffff4
M>reg GPR1=0
M>
```

This means: Add 1, GPR1, NOP, NOP, JMP .-4

Set the PC to this program:

```
M>reg PC=100
M>
```

Step, then check the register:

```
M>s
   PC=00000104
M>reg GPR1
   reg GPR1=00000001
M>
```

This should return"reg GPR1=00000001" .

Step some more and verify that GPR1 increments after every four steps:

```
M>s 4
   PC=00000104
M>reg GPR1
   reg GPR1=00000002
M>
```

Problems with the LAN Interface

If LAN communication does not work

If you cannot verify the connection , or if the commands are not accepted by the emulation module:

- ❏ Make sure that you wait for the power-on self test to complete before connecting.
- ❏ Make sure that the LAN cable is connected. Watch the LAN LED's on the back of the logic analysis system to see whether the system is seeing LAN activity. Refer to your LAN documentation for testing connectivity.
- ❏ Check that the host computer or debugger was configured with the correct LAN address. If the logic analysis system is on a different subnet than the host computer, check that the gateway address is correct.
- ❏ Make sure that the logic analysis system's IP address is set up correctly.

If it takes a long time to connect to the network

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

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

The subnet mask is set in the logic analysis system's System Admin window. If it then detects other subnet masks, it will generate error messages.

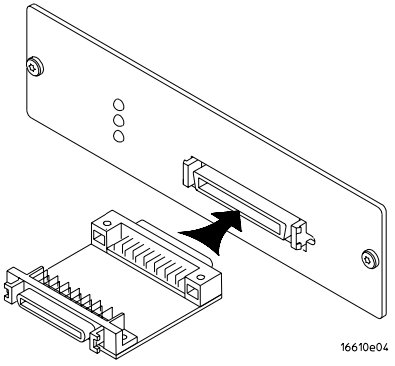
If there are many subnet masks in use on the local subnet, the logic analysis system may take a very long time to connect to the network after it is turned on.

Problems with the Emulation Module

Occasionally you may suspect a hardware problem with the emulation module or target interface module. The procedures in this section describe how to test the hardware, and if a problem is found, how to repair or replace the broken component.

To run the built-in performance verification test using the logic analysis system

1. End any Emulation Control Interface or debugger sessions.
2. Disconnect the 50-pin cable from the emulation module, and plug the loopback test board (HP part number E3496-66502) into the emulation module.



3. In the system window, click the emulation module and select Performance Verification.
4. Click Start PV.

   The results will appear onscreen.

To run complete performance verification tests using a telnet connection

1. Disconnect the 50-pin cable from the emulation module, and plug the loopback test board (HP part number E3496-66502) directly into the emulation module. Do not plug anything into the other end of the loopback test board.

   On a good system, the RESET LED will light and the BKG and USER LEDs will be out.

2. telnet to the emulation module.

3. Enter the pv 1 command.

See Also

Options available for the "pv" command are explained in the help screen displayed by typing "help pv" or "? pv" at the prompt. Note, however, that some of the options listed may not apply to your emulation module.

Examples:

If you are using a UNIX system, to telnet to a logic analysis system named "mylogic", enter:

```
telnet mylogic 6472
```

Here are some examples of ways to use the **pv** command.

To execute both tests one time:

```
pv 1
```

To execute test 2 with maximum debug output repeatedly until a ^C is entered:

```
pv -t2 -v9 0
```

To execute tests 3, 4, and 5 only for 2 cycles:

```
pv -t3-5 2
```

The results on a good system with the loopback test board connected, are as follows:

```
M>pv 1

   Testing: HPE3499C Series Emulation System
     Test  1: Powerup PV Results                              Passed!
     Test  2: Target Probe Feedback Test                      Passed!
     Test  3: Boundary Scan Master Test                       Passed!
     Test  4: I2C Test                                        Passed!
     Test  5: Data Lines Test                                 Passed!
   PASSED  Number of tests: 1           Number of failures: 0

          Copyright (c) Hewlett-Packard Co. 1987
 All Rights Reserved.  Reproduction, adaptation, or translation without prior
 written permission is prohibited, except as allowed under copyright laws.

   HPE3499C Series Emulation System
     Version:   A.07.54 22Apr98
     Location:  Generics

   HPE3454A PowerPC 700 JTAG Emulator
     Version:   A.01.07 26May98
 M>
```

You may get an error like "!ERROR 172! Bad status code (0xff) from the hard reset sequence" just before the prompt. This is because the selftest loopback connector is installed instead of being connected to a real PowerPC target system. You may also get a "?>" prompt for the same reason, and this is normal and expected. Any errors after the "PASSED Number of tests: 1 Number of failures: 0" line can be ignored.

If a performance verification test fails

There are some things you can do if a failure is found on one of these tests. Details of the failure can be obtained through using a -v option ("verbose" level) of 2 or more.

> If the particular failure you see is not listed below, contact HP for assistance.

TEST 3: Boundary Scan Master Test
TEST 4: I2C Test

If these tests are not executed, check that you have connected the loopback test board.

If these tests fail, return the emulation module to HP for replacement.

Returning Parts to Hewlett-Packard for Service

The repair strategy for this emulation solution is board replacement.

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

To return a part to Hewlett-Packard

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

   Keep any parts which you know are working. For example, if only the target interface module is broken, keep the emulation module and cables.
4. When the part has been replaced, it will be sent back to you.

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

   The HP service center can also troubleshoot the hardware and replace the failed part. To do this, send your entire measurement system to the service center, including the logic analysis system, target interface module, and cables.

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

To obtain replacement parts

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

Part numbers

Exchange Assemblies	
Part Number	**Description**
E5901A #070	Emulation module
Replacement Assemblies	
Part number	**Description**
16700-61608	Expansion cable
E3494-61604	16-pin cable
E3496-61601	50-pin cable
E3496-66502	emulation module loopback test board
E3481-61601	20-pin cable
E3452-66502	Target interface module (PPC JTAG board)

Cleaning the Instrument

If the instrument requires cleaning:

1. Remove power from the instrument.
2. Clean the instrument with a cloth that has been moistened with a mixture of mild detergent and water.
3. Make sure that the instrument is completely dry before reconnecting it to a power source.

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 In 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 module.

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 By loading new firmware and connecting a different TIM, your emulator migrates from support of one PowerPC model to support of another PowerPC model.

Emulation Module An emulation module is installed within the mainframe of a logic analyzer. It provides run control within an emulation and analysis test setup. See Emulation Probe.

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." See Emulation Module.

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.

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.

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.

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

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.

Prototype Analyzer The HP 16505A prototype analyzer acts as an analysis and display processor for the HP 16500B/C logic analysis system. It provides a windowed interface and powerful analysis capabilities. Replaced by HP 16600A/700A-series logic analysis systems.

Run Control Probe See Emulation Probe and Emulation Module.

Setup Assistant A software program that guides a user through the process of connecting and configuring a logic analyzer to make measurements on a specific microprocessor.

Shunt Connector. See Jumper.

Software Probe See Emulation Probe.

Solution HP's term for a set of tools for debugging your target system. A solution includes probing, inverse assembly, the HP 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. It is designed to perform logic analysis. 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 might be installed within its frame.

State Analysis When the logic analyzer is configured to capture data synchronously with a clock signal in the target system.

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 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 emulation module or emulation probe to signals from the debug port on a target system.

TIM See Target Interface Module.

Timing Analysis When 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

Index

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.
Hewlett-Packard Company,
3000 Hanover Street,
Palo Alto, CA 94304 U.S.A.
Rights for non-DOD U.S. Government Departments and Agencies are set forth in FAR 52.227-19 (c) (1,2).

Document Warranty

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

Hewlett-Packard 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.

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

Safety

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

Warning

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

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

Product Warranty

This Hewlett-Packard product has a warranty against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Hewlett- Packard Company 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 Hewlett- Packard.

For products returned to Hewlett-Packard for warranty service, the Buyer shall prepay shipping charges to Hewlett- Packard and Hewlett-Packard 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 Hewlett-Packard from another country.

Hewlett-Packard warrants that its software and firmware designated by Hewlett-Packard for use with an instrument will execute its programming instructions when properly installed on that instrument. Hewlett-Packard 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. Hewlett-Packard 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. Hewlett-Packard 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 Hewlett-Packard products. For any assistance, contact your nearest Hewlett-Packard Sales Office.

Certification

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

About this edition

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

Publication number
E2498-97004, December, 1999
Printed in USA.

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

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