User's Guide

Publication number E8128-97003
December 1999

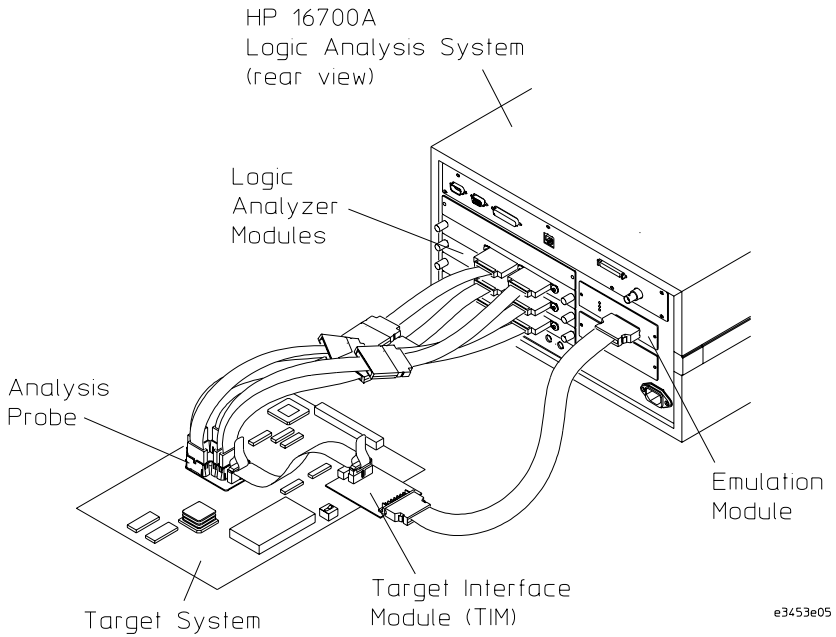
For Safety information, Warranties, and Regulatory information, see the pages behind the index.

Solutions for the Motorola MPC8240

This emulation solution lets you use the HP 16600/16700A-series logic analysis system to debug and characterize Motorola MPC8240 target systems.

Emulation Solution

The emulation solution is a bundled product consisting of an analysis probe (optional), an inverse assembler, an emulation module (and its cables and adapters), and the HP B4620B source correlation tool set.



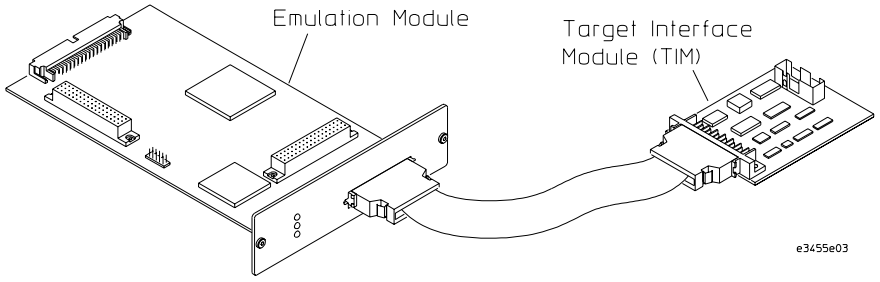
Analysis Probe and Inverse Assembler

The inverse assembler for MPC8240 processors, along with an analysis probe or a target system that has been designed with connectors for logic analyzer probes, lets you capture and display the processor's signal values with a logic analyzer. (The inverse assembler model number is HP E9611A Option 001 when ordered without an analysis probe.)

Emulation Module and Target Interface Module

The emulation module lets you use a microprocessor's built-in debugging features (like starting/stopping program execution, setting breakpoints, and modifying the contents of processor registers and target system memory). A high-level source debugger can use the emulation module to debug code running on the target system.

The target interface module (TIM) adapts the emulation module to the MPC8240 microprocessor's JTAG port.



Source Correlation Tool Set

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

In This Book

This book describes the following products:

Product Ordered	Supports	Includes
HP E9611A Option 001 inverse assembler	MPC8240	HP E8128A inverse assembler
HP E9611A Option 002 analysis probe	MPC8240	HP E8127A analysis probe and inverse assembler, BGA probing kit
HP E9511A Option 001 emulation solution	MPC8240	HP E8128A inverse assembler, HP 16610A emulation module, target interface module (TIM), and HP B4620B Source Correlation Tool Set
HP E9511A Option 002 emulation solution	MPC8240	HP E8127A analysis probe and inverse assembler, BGA probing kit, HP 16610A emulation module, target interface module (TIM), and HP B4620B Source Correlation Tool Set

Before you use this book, you should have already set up the HP 16600A/16700A-series logic analysis system, installed logic analyzer modules, and learned how to use the logic analysis system (see the logic analysis system's *Installation Guide*).

This book has five parts:

- Part1, "Installation Guide"
- Part 2, "Using the Logic Analyzer"
- Part 3, "Using the Emulation Module"
- Part 4, "Reference"
- Part 5, "Service Guide"

See Also

The logic analysis system's online help for more information on using the HP B4620B Source Correlation Tool Set.

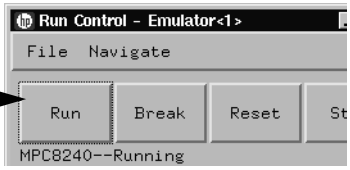
Use the Setup Assistant

Click here to connect the logic analyzer cables, and automatically load the correct configuration files. See page 22.

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.

If you cannot easily find the information you need, send email to documentation@col.hp.com. Your comments will help HP improve future manuals. (This address is for comments only; contact your local HP representative if you need technical support.)

Contents

Part 5 Service Guide 279

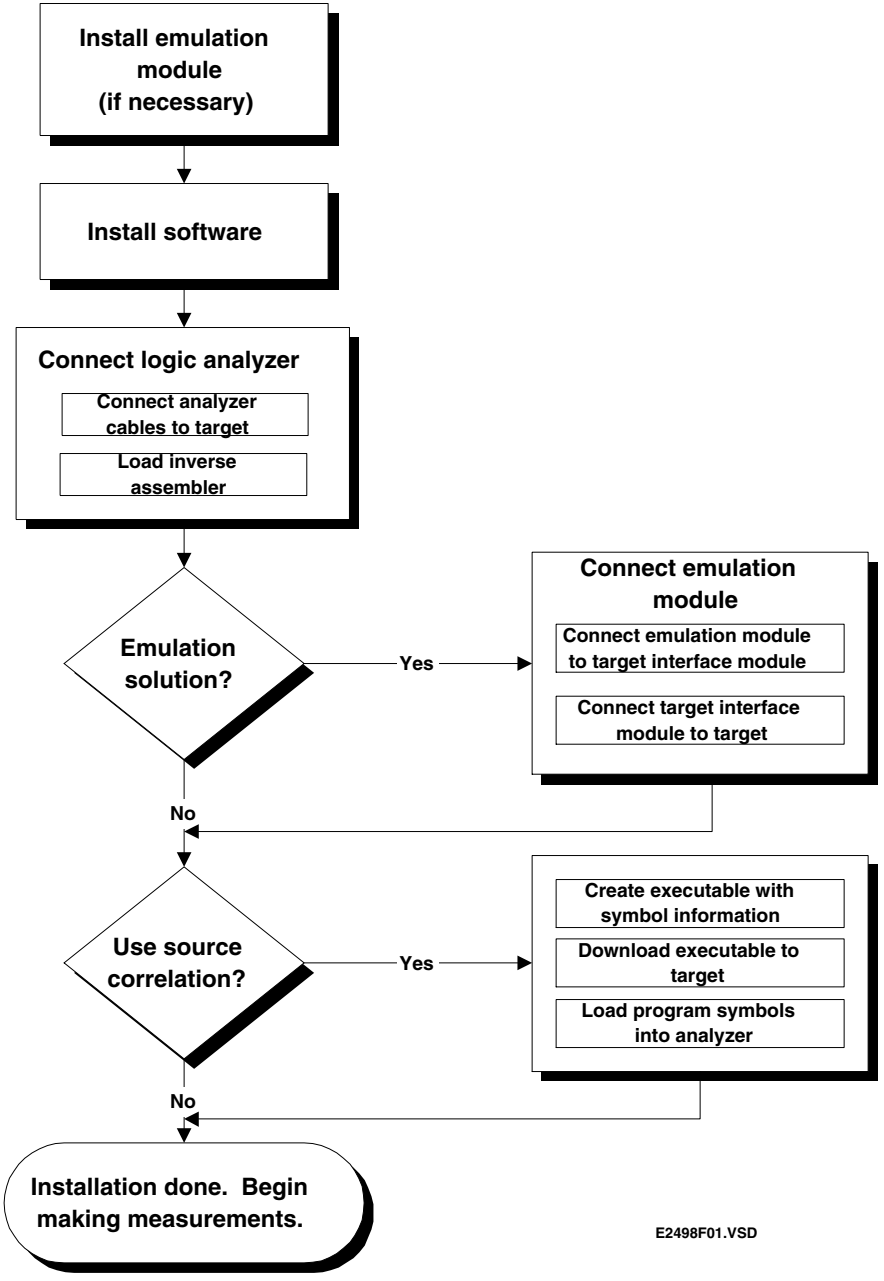
Part 1

Installation Guide

Overview of Installation and Setup

Follow these steps to connect your equipment:

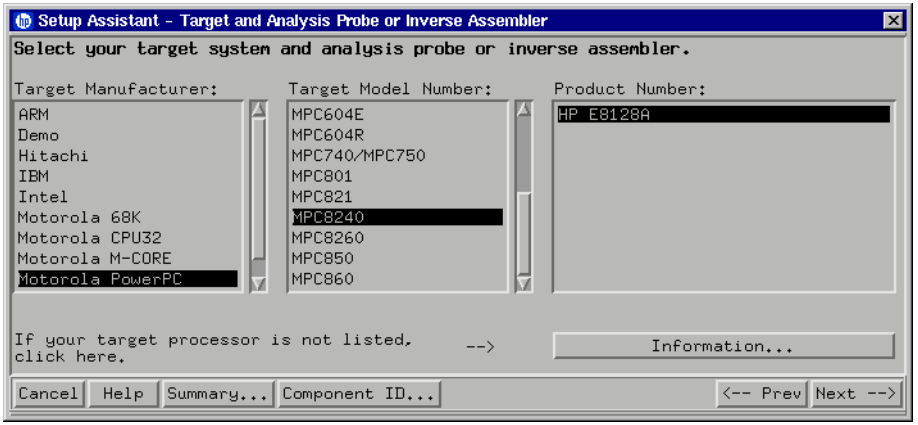
1. Check that you received all of the necessary equipment. See the chapter "Equipment and Requirements," on page 23.
2. If you need to install an emulation module in an HP 16600A/16700A-series logic analysis system, see "Installing the Emulation Module" on page 64.
3. Install the software. See "Installing Software" on page 70.
4. If you have an HP 16600A/16700A-series logic analysis system, use the Setup Assistant to help you connect the logic analyzer and configure the inverse assembler and emulation module. See "Using the Setup Assistant" on page 74.


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

Using the Setup Assistant

The Setup Assistant is an online tool for connecting and configuring your logic analysis system for microcontroller 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 an analysis probe, an emulation module, or other supported equipment. It will also guide you through connecting an analysis probe to the target system.



Start the Setup Assistant by clicking its icon in the system window.

> To connect to your target system using an analysis probe, choose **HP E8127A**. To connect using connectors built into your target system or a Motorola PMC board, choose **HP E8128A**.

If you ordered this product 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 “Installing Software” on page 70.

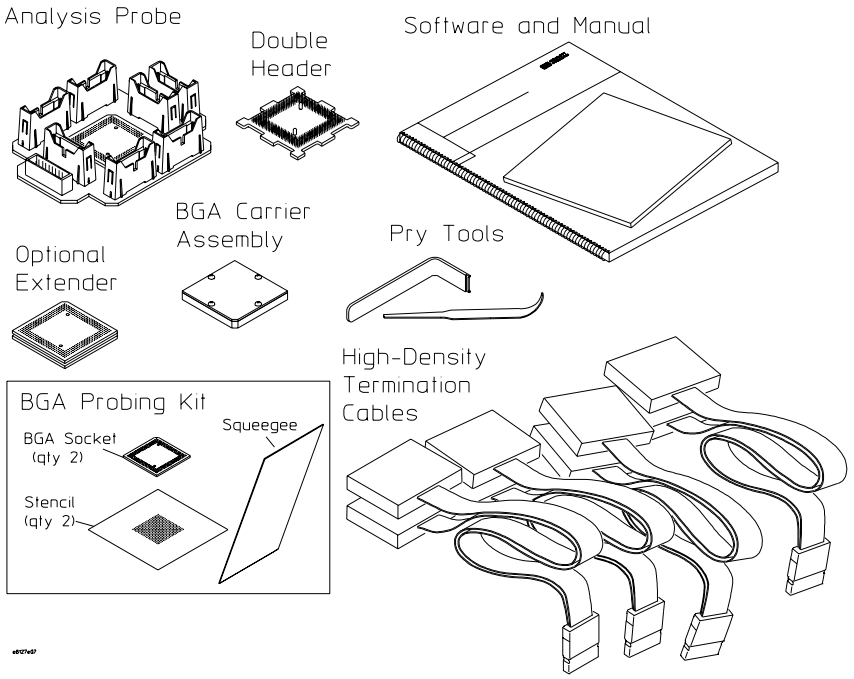
1

Equipment and Requirements

Equipment and Software Supplied

Listed below is the equipment and software supplied with the emulation solution for the MPC8240 (see page 4 for a list of which components are supplied with each ordering option).

Analysis Probe



The analysis probe includes:

- HP E8127A analysis probe.
- Four HP E5346A high-density termination cables.
- BGA probing kit (HP part number E8161-60001).
- BGA carrier assembly.
- Double header.
- Extender.
- One pry tool.

- Logic analyzer configuration files and the inverse assembler software on a CD-ROM (for HP 16600A/16700A-series logic analysis systems).
- This *User's Guide*.

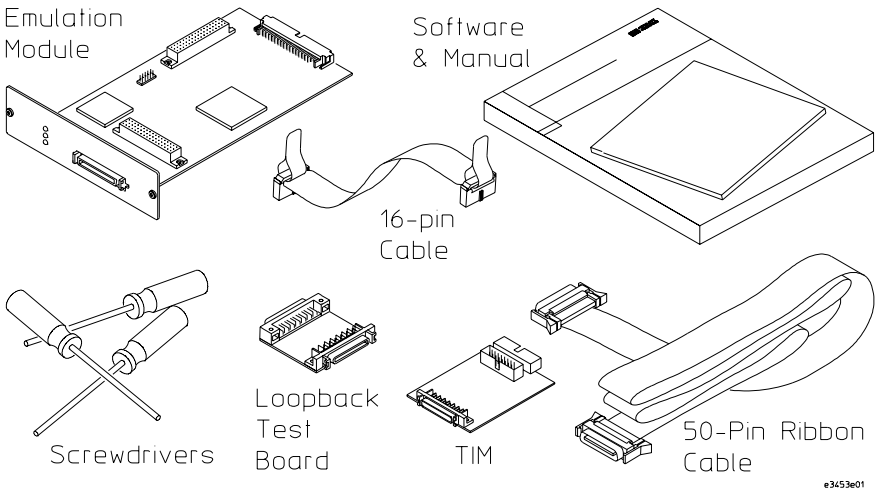
Inverse Assembler

The inverse assembler (HP E9611A Option 001 when ordered separately) includes:

- Logic analyzer configuration files and the inverse assembler software on a CD-ROM (for HP 16600A/16700A-series logic analysis systems).
- This *User's Guide*.

The inverse assembler, without the analysis probe, is identified as "E8128A" in the Setup Assistant.

Emulation Module



The emulation module includes:

- An HP 16610A emulation module.

  If you ordered an emulation module as part of your HP 16600A/16700A-series logic analysis system, it is already installed in the frame.

- 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 (TIM).
- A target interface module (TIM) circuit board for connecting the emulation module to a JTAG port in the target system.
- A 16-pin ribbon cable for connecting the target interface module (TIM) to a JTAG port connector in the target system.
- One Torx T-10 and one Torx T-15 screwdriver.
- An emulation module loopback test board (HP part number E3496-66502).

Source Correlation Tool Set

The source correlation tool set includes:

- An entitlement certificate for licensing the software.
- The HP 16600A/16700A-series logic analysis system software CD-ROM.

The HP B4620B Source Correlation Tool Set software is already installed on the HP 16600A/16700A-series logic analysis system's disk. All you need is the entitlement certificate for licensing the source correlation tool set software. The CD-ROM is included in case you need to re-install the software.

Additional Equipment and Software Required

Listed below is the additional equipment and software required by the emulation solutions for the MPC8240.

Analysis Probe

The analysis probe requires:

- A logic analyzer module to capture MPC8240 processor execution.

Inverse Assembler

The inverse assembler, used without an analysis probe, requires:

- Logic analyzer connector headers designed into the target system.
- The proper termination for the type of connector headers in the target system.
- A logic analyzer module to capture MPC8240 processor execution.

Refer to the Chapter 2, “Preparing the Target System,” on page 29 for more information on headers, terminations, and supported logic analyzers.

Emulation Module

The emulation module requires:

- An HP 16600A/16700A-series logic analysis system into which it can be installed.
- Interface software that gives you access to the emulation module’s functionality.

  You can use the HP 16600A/16700A-series logic analysis system’ s Emulation Control Interface. Or, you can use a third-party high-level source debugger to access and control the emulation module.

Source Correlation Tool Set

The source correlation tool set requires the HP 16600A/16700A-series logic analysis system.

Other Optional Equipment and Software

Debuggers

The emulation module works with several debuggers offered by other vendors. See Chapter 11, “Using a Debugger with the Emulation Module,” on page 191.

PCI Bus Analysis

The analysis probe preserves both 3.3V and 5V signal levels. The emulation solution may be used with PCI analysis probes and inverse assemblers from FuturePlus Systems and Corelis.

2

Preparing the Target System

Preparing for Logic Analysis (and Inverse Assembly)

The MPC8240 inverse assembler for the HP 16600/16700-series logic analysis system requires a minimum of four logic analyzer pods.

For inverse assembly of both instructions and data, 8 pods are required. If optional signals are used, such as PCI analysis, additional logic analyzer pods are required.

If you are not using an analysis probe or PMC board, you must design high-density connectors into your target system for logic analyzer probe pods.

This section describes these considerations in more detail.

- Design Considerations
- Supported Logic Analyzers
- Using High-Density Connectors
- Recommended Connector Layout and Signal Routing
- Alternative Connector Layout and Signal Routing

Design Considerations

There are several things to keep in mind when designing a MPC8240 target system.

Configuring for Debug Mode

To use the inverse assembler, the MPC8240 must be configured to run in debug mode. Debug mode enables the inverse assembler to reconstruct full 32-bit physical addresses. See “Enabling Debug Mode on the MPC8240” on page 57.

Cache Memory

Internal Instruction Cache. The microprocessor supplies no external information when the cache is enabled. This poses a problem for logic analysis, since there is no linear address and data cycles to reconstruct code flow. Therefore, cache-on trace reconstruction must be used. See “Using Cache-On Trace Reconstruction” on page 106.

Data Cache. The microprocessor provides no external information for accesses to the internal data cache. The inverse assembler does not support accesses to the internal data cache.

Electrical Requirements

Any probed signal line must be able to supply a minimum of 600 mV to the probe tip and handle a minimum loading of 90 kΩ shunted by 10 pF. The maximum input voltage for the logic analyzer is +/- 40 volts peak.

For all signals, the logic analyzers require a minimum combined setup/hold window. For HP 16600-series logic analysis systems, the combined setup/hold window must be at least 4.5 ns. For all other logic analyzers, the combined window must be at least 3.5 ns.

The signals labelled “Pull up Vdd (TTL)” in the tables beginning on page 50 should be pulled up to V_{dd} using 10 kΩ (approximate value) resistors.

Supported Logic Analyzers

A minimum of four logic analyzer pods are required for inverse assembly. If you choose to probe optional signals, additional logic analyzer pods are required.

The inverse assembler only works in HP 16600A/16700A-series logic analysis systems. The HP 16500 logic analysis system and HP 166x/167x portable logic analyzer families are not supported.

The inverse assembler requires the latest logic analysis system software (version A.01.41 or greater). The latest logic analysis system software version is on the CD-ROM shipped with this emulation solution or inverse assembler product.

Given these restrictions, the following logic analyzers can be used:

Logic Analyzer	Channel Count	State Speed	Timing Speed	Memory Depth
HP 16550A (one or more cards)	102/card	100 MHz	250 MHz	4K states
HP 16554A (one or more cards)	68/card	110 MHz	125 MHz	500K states
HP 16555A (one or more cards)	68/card	110 MHz	250 MHz	1M states
HP 16555D (one or more cards)	68/card	110 MHz	250 MHz	2M states
HP 16556A (one or more cards)	68/card	100 MHz	200 MHz	1M states
HP 16556D (one or more cards)	68/card	100 MHz	200 MHz	2M states
HP 16557D (one or more cards)	68/card	135 MHz	250 MHz	2M states
HP 16600A*	204	100 MHz	125 MHz	64K states
HP 16601A*	136	100 MHz	125 MHz	64K states
HP 16602A*	102	100 MHz	125 MHz	64K states
HP 16603A*	68	100 MHz	125 MHz	64K states
HP 16710A (one or more cards)	102/card	100 MHz	250 MHz	8K states
HP 16711A (one or more cards)	102/card	100 MHz	250 MHz	32K states
HP 16712A (one or more cards)	102/card	100 MHz	250 MHz	128K states
HP 16715A (one or more cards)	68/card	167 MHz	333/667 MHz	2M/4M samples
HP 16716A (one or more cards)	68/card	167 MHz	333/667 MHz, 2 GHz TimingZoom	512K/1M samples
HP 16717A (one or more cards)	68/card	333 MHz	333/667 MHz, 2 GHz TimingZoom	2M/4M samples

Notes:

* The target must have a combined setup/hold window of 4.5 ns or greater. This is due to the limitations of the HP 1660x analyzer's setup and hold specifications.

Designing A Target System for the Analysis Probe

Electrical Requirements

In addition to the requirements listed in “Electrical Requirements” on page 31, your target system should meet the following requirements:

Electrical requirements for PCI bus analysis

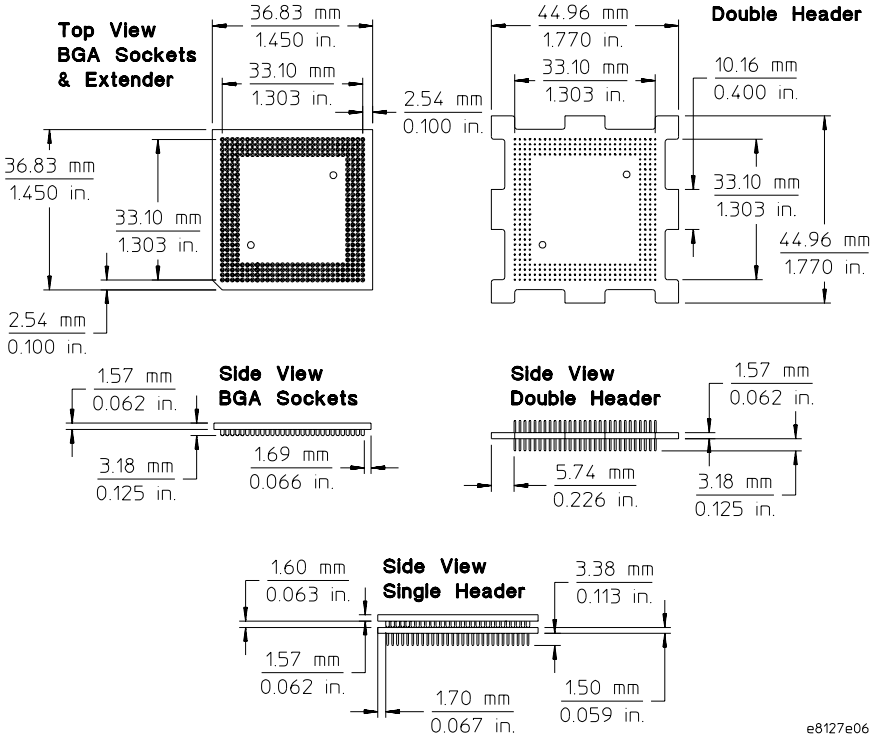
You do not need to condition the PCI bus signals; the analysis probe preserves both the 3.3V and 5V signal levels.

Mechanical Requirements

Keep-Out Area on the Target Board

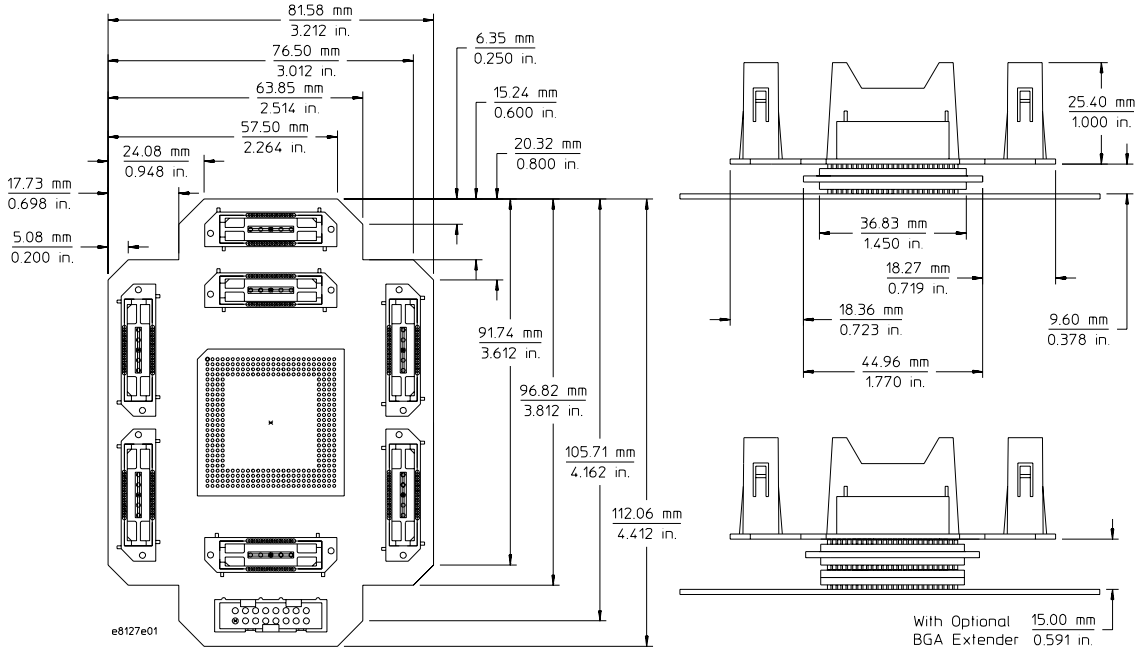
The BGA socket placed into the BGA footprint extends slightly beyond the outer row of pads. It is important to keep components from interfering with the socket in this restricted area.

The analysis probe requires a keep-out area of 44.96 mm by 44.96 mm where it overhangs the BGA socket. Components within the required keep-out area must be no higher than 3.18 mm. If the analysis probe interferes with components of the target system, or if a higher profile is required, additional BGA extenders (HP part number E8127-87607) can be used, but they add additional electrical intrusion.



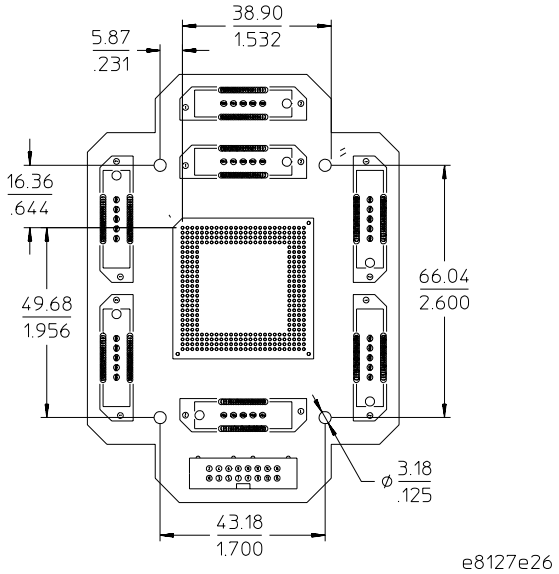
Clearance above the Target Board

Be careful to allow adequate space above the target board for the analysis probe, and for egress of the logic analyzer cables.



Mounting Holes

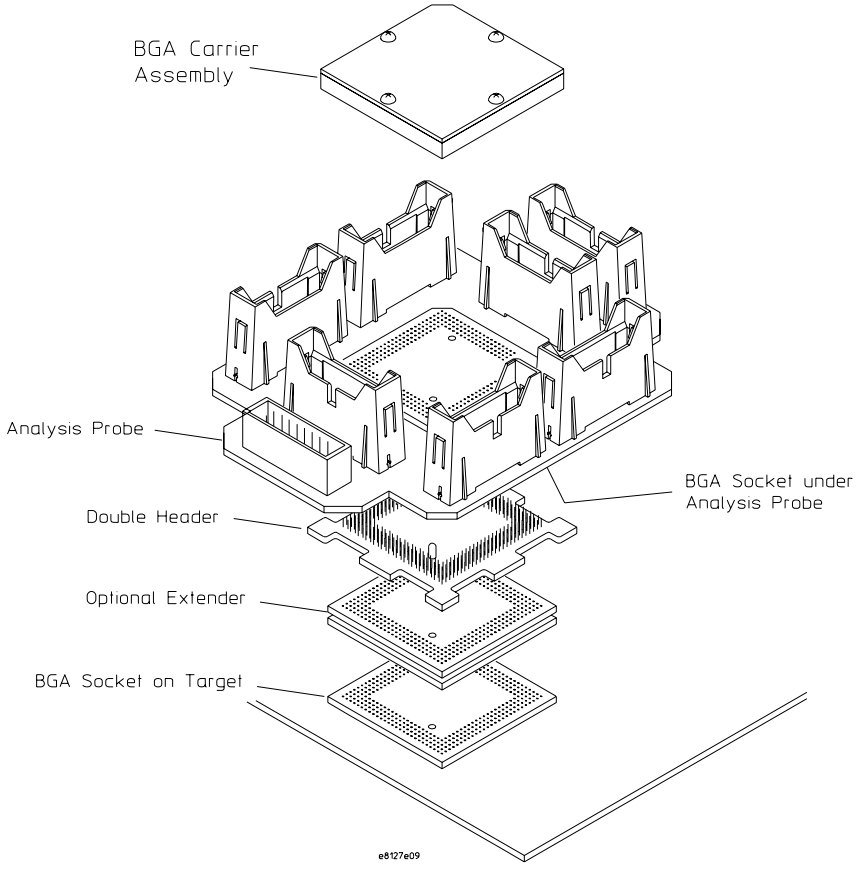
HP recommends drilling four non-plated holes according to the following dimensions. Mounting the analysis probe greatly alleviates stress on the fragile solder joints when the logic analyzer cables are connected.



Attaching the Analysis Probe to the Target System

If you are designing logic analyzer and debug port connectors into your target system, and won't be using an analysis probe, skip this section and refer to "Designing Logic Analyzer Connectors into Your Target System" on page 45 and "Preparing for Emulation" on page 58 instead.

Overview



Attaching the analysis probe to the target system consists of the following steps, which are described on the following pages:

- Solder the BGA socket onto the target system.
- Assemble the microprocessor into the BGA carrier.
- Test the target system with the BGA carrier assembly, without the analysis probe.
- Disconnect the BGA carrier assembly from the target system.
- Install the analysis probe onto the target system, and then install the BGA carrier assembly onto the analysis probe.

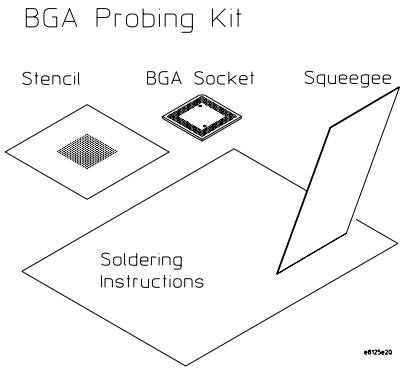
Protect Your Equipment

The analysis probe socket assembly pins are covered for shipment with a conductive foam wafer or conductive plastic pin protector. This is done to protect the delicate gold-plated pins from damage due to impact. When you're not using the analysis probe, protect the socket assembly pins from damage by covering them with the pin protector.

To solder the BGA socket onto the target system

The BGA probing kit, HP part number E8161-60001, requires a target system with an empty 352-pin BGA pad array. Install the BGA probing kit using the following instructions.

1. Ensure that your target system has a 352-pin BGA pad array with proper connections for your target microprocessor. This BGA pad array must be clean, unused, and have no solder on its pads.
2. Ensure that pin A1 of the BGA socket is properly aligned with pin A1 on the BGA pad array.
3. Following the soldering instructions in the process sheet that came with the BGA probing kit, install the socket onto the 352-pin BGA pad array, and solder it in place.



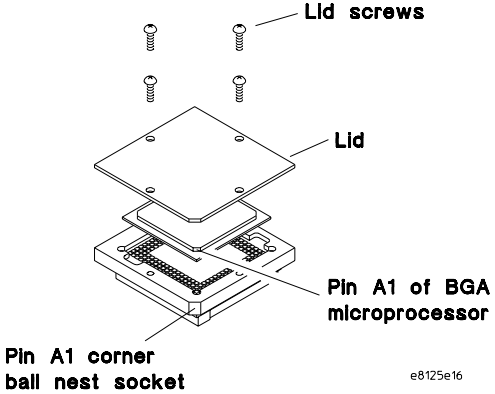
To assemble the microprocessor into the BGA carrier

The analysis probe has a BGA carrier for a 352-pin BGA microprocessor. Use the procedure below to install the BGA microprocessor into the BGA carrier.

1 Align pin A1 on the BGA microprocessor with the pin A1 corner of the BGA carrier (see below).

CAUTION: Serious damage to the target system or analysis probe can result from incorrect connection. Note the position of pin A1 on the BGA carrier and BGA microprocessor prior to making any connection.

2 Place the BGA microprocessor into the BGA carrier, and tighten the lid screws.

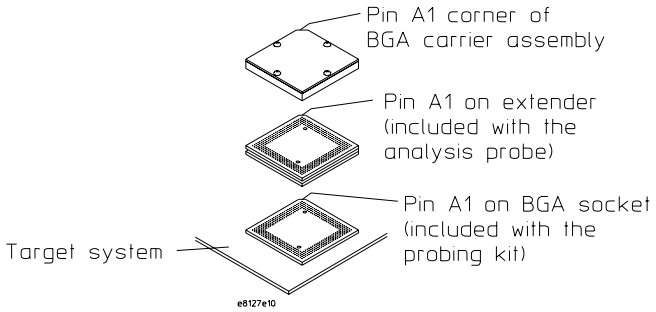


CAUTION: Multiple insertions of the BGA microprocessor into the BGA carrier may degrade the ball nest socket connections. Once the BGA microprocessor is inserted in the ball nest socket, tighten the four lid screws forcefully. Only remove the BGA microprocessor from the BGA carrier when necessary for silicon upgrades.

To test the target system with the BGA carrier assembly

Before installing the analysis probe onto the target system, ensure that the socket and extender have been installed successfully with the following steps.

1. Install the BGA carrier assembly into the extender.
2. Install the extender into the BGA socket on your target system.



3. Turn on your target system and check operation.

   The BGA socket, extender, and BGA carrier assembly add inductance and capacitance. Ensure that your target system operates properly before installing the analysis probe board assembly.

   Open connections or shorts may exist after soldering the BGA socket to the target board. If a previously functioning target board does not function after installing the socket, check continuity of the socket pins. Touch a dry-tip soldering iron to any open pin.

To remove the BGA carrier assembly (or analysis probe) from the socket

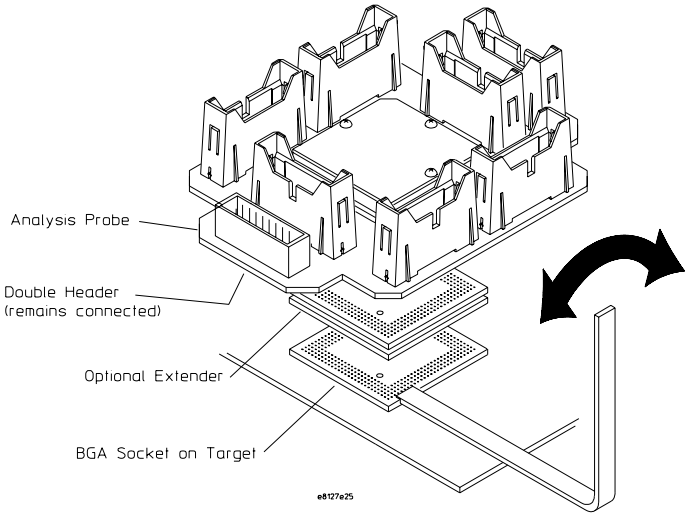
You must remove the BGA carrier assembly to attach the analysis probe. Use this procedure when disconnecting the BGA carrier assembly (or the analysis probe) from the BGA socket.

The extractor tool comes with an *Operating Guide* showing how to use the extractor tool to disconnect the BGA carrier assembly or the analysis probe from the BGA socket.

Follow these guidelines:

- Refer to the *Operating Guide* and use the extractor tool to lift the BGA carrier assembly (or the analysis probe) from the socket. Keep all connector pins straight during removal.
- Do not plug anything other than the analysis probe or optional extender into the BGA socket on the target board.
- Separate the BGA carrier assembly (or the analysis probe) from the socket on the target board, or from the optional extender.

CAUTION: Do not remove the double header from the analysis probe board. Removing the double header could damage the analysis probe.



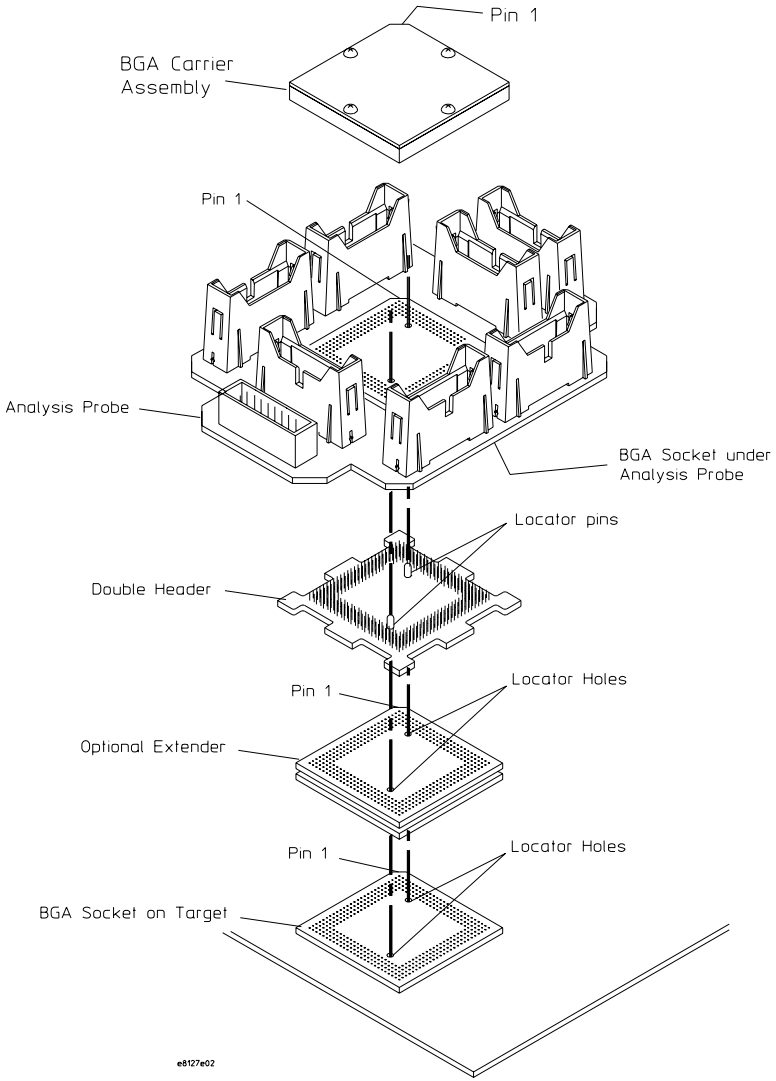
To attach the analysis probe (with BGA carrier) to the target system

A BGA socket is on the bottom of analysis probe. It connects to the double header which in turn connects to the extender or socket on the target system.

1. Install the double header into the BGA socket on the bottom of the analysis probe.
2. Install the analysis probe into the extender or socket on the target system. Ensure that pin A1 is properly aligned (see the following figure).

CAUTION: Target System Damage. Serious damage to the target system or analysis probe can result from incorrect connection. Note the position of pin A1 on the target system, double header, analysis probe, and BGA carrier assembly prior to making any connection.

If the analysis probe interferes with components of the target system, or if a higher profile is required, additional BGA extenders (HP part number E8127-87607) can be used.


Pin 1
BGA Carrier Assembly
Pin 1
Analysis Probe
BGA Socket under Analysis Probe
Locator pins
Double Header
Pin 1
Locator Holes
Optional Extender
Pin 1
Locator Holes
BGA Socket on Target
e8127e02

Designing Logic Analyzer Connectors into Your Target System

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

If you are using an analysis probe, you do not need to include connectors on your target system.

Using High-Density Connectors

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

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

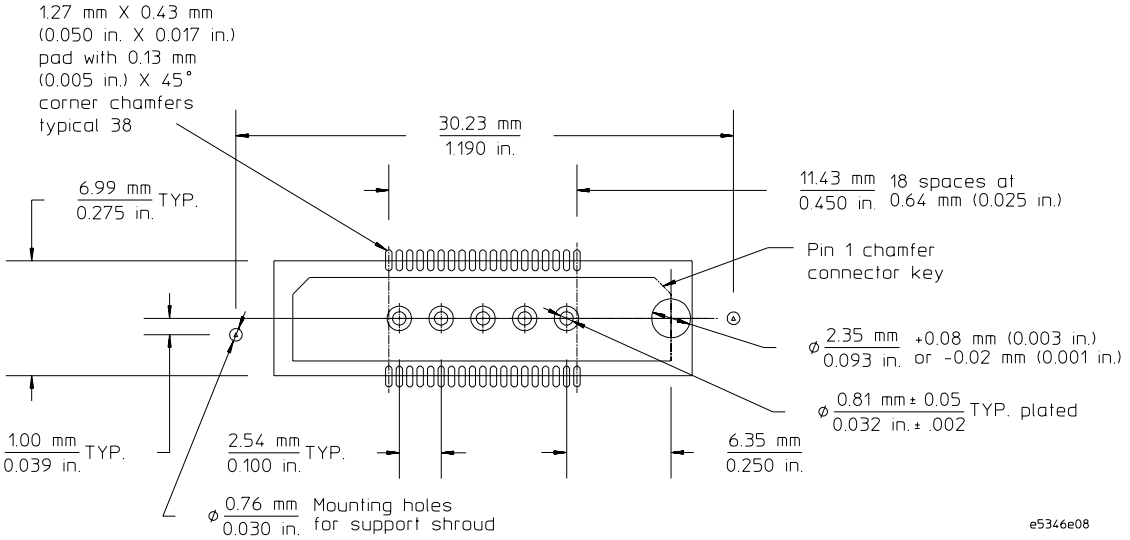
See Also

More information on this connector is available in the document *HP E5346A High-Density Termination Adapter*, HP part number 5965-5475E. This document is available in Portable Document Format (PDF) from the web site:

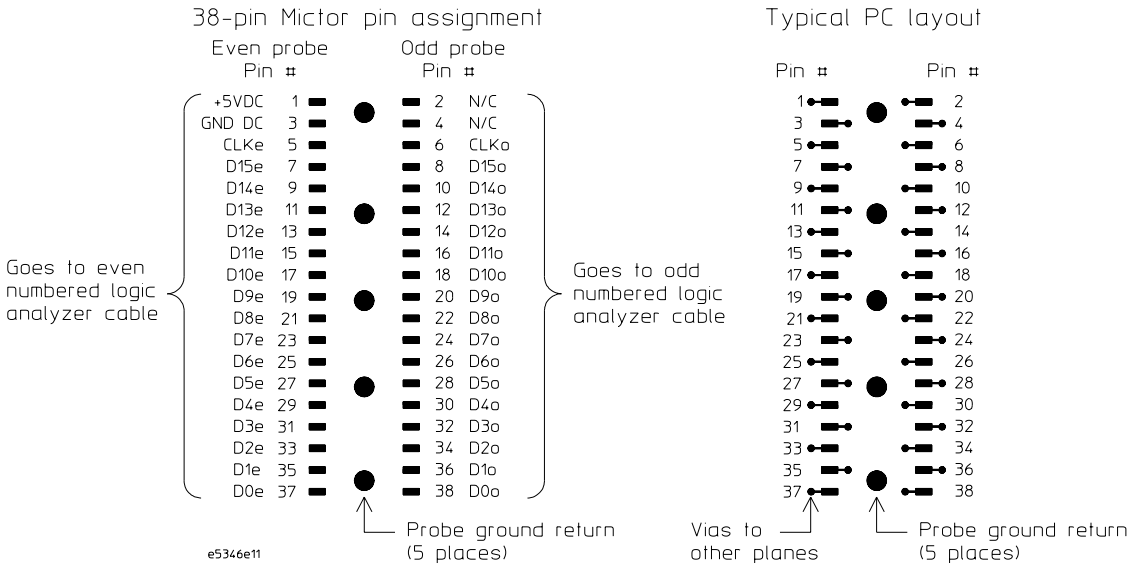
http://www.tmo.hp.com/tmo/datasheets/English/HPE5346A.html

High-Density Connector Mechanical Specifications

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



The high-density connector pin assignment and recommended circuit board routing are shown below.



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

Recommended Connector Layout and Signal Routing

The advantages of the recommended configuration are:

- It is optimized for minimum trace lengths and electrical loading.

The disadvantages are:

- It requires four high-density connectors for main memory disassembly in an 8-pod logic analyzer. This is required since pods 7 and 8 are split between two Mictor connectors to minimize trace lengths.

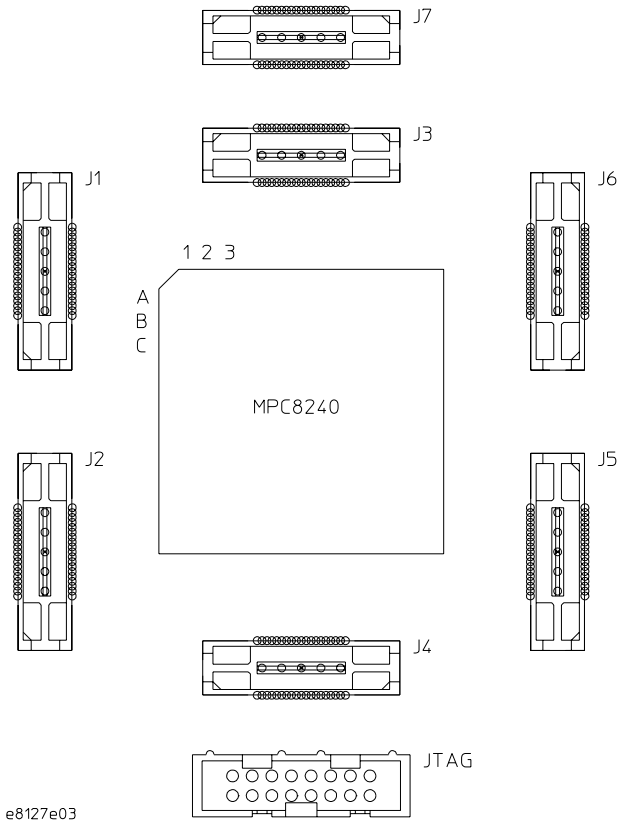
Recommended Configuration Connection Notes

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

- '#' or $\overline{\text{overscore}}$ denotes an active low signal.
- J1-J2: Required for inverse assembly.
- J3 odd, J4 odd: Required for inverse assembly with 32-bit data.
- J3 even, J4 even: Required for inverse assembly with 64-bit data.
- J7: Miscellaneous signals
- J5-J6: (Optional) Required for PCI bus.

Recommended Connector Layout

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



Recommended Signal Routing

Mictor Connector J1					
J1 Pin	**BGA Pin**	**MPC8240 Signal**	**J1 Pin**	**BGA Pin**	**MPC8240 Signal**
1		nc	2		nc
3		nc	4		nc
5	A16	$\overline{\text{MIV}}$	6	D1	SDRAM_CLK[0]
7	Y2	RTC	8	B15	debug_addr[1]/CKO
9	H2	CKE	10	F2	debug_addr[0]/$\overline{\text{QACK}}$
11	J2	debug_addr[15]/MTP[0]	12	P2	SDBA0
13	F1	debug_addr[14]/MTP[1]	14	P1	SDMA[12]/SDBA1
15	AF19	debug_addr[14]/FTP[0]	16	N1	SDMA[11]
17	AF17	debug_addr[12]/FTP[1]	18	R1	SDMA[10]
19	AD26	debug_addr[11]/FTP[2]	20	R2	SDMA[9]
21	A22	debug_addr[10]/PLL_CFG[0]	22	T1	SDMA[8]
23	B19	debug_addr[9]/PLL_CFG[1]	24	T2	SDMA[7]
25	A21	debug_addr[8]/PLL_CFG[2]	26	U4	SDMA[6]
27	B18	debug_addr[7]/PLL_CFG[3]	28	U2	SDMA[5]
29	B17	debug_addr[6]/PLL_CFG[4]	30	U1	SDMA[4]
31	W26	debug_addr[5]/$\overline{\text{GNT}}$[4]	32	V1	SDMA[3]
33	Y26	debug_addr[4]/$\overline{\text{REQ}}$[4]	34	V3	SDMA[2]
35	AF26	debug_addr[3]/PCI_CLK[4]	36	W1	SDMA[1]
37	C25	debug_addr[2]/FTP[3]	38	W2	SDMA[0]
Even Cable			**Odd Cable**		
Logic Analyzer Pod 2			**Logic Analyzer Pod 1**		

* The Motorola naming convention for the SDMA and debug_addr signals is now little-endian.

Mictor Connector J2					
J2 Pin	**BGA Pin**	**MPC8240 Signal**	**J2 Pin**	**BGA Pin**	**MPC8240 Signal**
1		nc	2		nc
3		nc	4		nc
5	AD2	$\overline{\text{SDCAS}}$	6	AD1	$\overline{\text{SDRAS}}$
7	AF3	PAR/AR[0]	8	H1	$\overline{\text{FOE}}$
9	AE3	PAR/AR[1]	10	AA1	$\overline{\text{WE}}$
11	G4	PAR/AR[2]	12	Y1	$\overline{\text{AS}}$
13	E2	PAR/AR[3]	14	Y4	RAS / $\overline{\text{CS}}$[0]
15	AE4	PAR/AR[4]	16	AA3	RAS / $\overline{\text{CS}}$[1]
17	AF4	PAR/AR[5]	18	AA4	RAS / $\overline{\text{CS}}$[2]
19	D2	PAR/AR[6]	20	AC4	RAS / $\overline{\text{CS}}$[3]
21	C2	PAR/AR[7]	22	M2	RAS / $\overline{\text{CS}}$[4]
23	AB1	CAS / $\overline{\text{DQM}}$[0]	24	L2	RAS / $\overline{\text{CS}}$[5]
25	AB2	CAS / $\overline{\text{DQM}}$[1]	26	M1	RAS / $\overline{\text{CS}}$[6]
27	K3	CAS / $\overline{\text{DQM}}$[2]	28	L1	RAS / $\overline{\text{CS}}$[7]
29	K2	CAS / $\overline{\text{DQM}}$[3]	30	N4	$\overline{\text{RCS0}}$
31	AC1	CAS / $\overline{\text{DQM}}$[4]	32	N2	$\overline{\text{RCS1}}$
33	AC2	CAS / $\overline{\text{DQM}}$[5]	34	AF2	MAA[0]
35	K1	CAS / $\overline{\text{DQM}}$[6]	36	AF1	MAA[1]
37	J1	CAS / $\overline{\text{DQM}}$[7]	38	AE1	MAA[2]
Even Cable			**Odd Cable**		
Logic Analyzer Pod 4			**Logic Analyzer Pod 3**		

Mictor Connector J3					
J3 Pin	**BGA Pin**	**MPC8240 Signal**	**J3 Pin**	**BGA Pin**	**MPC8240 Signal**
1		nc	2		nc
3		nc	4		nc
5			6		
7	B1	DL[16]	8	E4	DH[16]
9	A1	DL[17]	10	A2	DH[17]
11	A3	DL[18]	12	B3	DH[18]
13	A4	DL[19]	14	D4	DH[19]
15	A5	DL[20]	16	B4	DH[20]
17	A6	DL[21]	18	B5	DH[21]
19	A7	DL[22]	20	D6	DH[22]
21	D7	DL[23]	22	C6	DH[23]
23	A8	DL[24]	24	B7	DH[24]
25	B8	DL[25]	26	C9	DH[25]
27	A10	DL[26]	28	A9	DH[26]
29	D10	DL[27]	30	B10	DH[27]
31	A12	DL[28]	32	A11	DH[28]
33	B11	DL[29]	34	A13	DH[29]
35	B12	DL[30]	36	B13	DH[30]
37	A14	DL[31] (LSB)	38	A15	DH[31]
Even Cable			**Odd Cable**		
Logic Analyzer Pod 6			**Logic Analyzer Pod 5**		

Mictor Connector J4					
J4 Pin	**BGA Pin**	**MPC8240 Signal**	**J4 Pin**	**BGA Pin**	**MPC8240 Signal**
1		nc	2		nc
3		nc	4		nc
5			6		
7	AD17	DL[0]	8	AC17	DH[0] (MSB)
9	AE17	DL[1]	10	AF16	DH[1]
11	AE15	DL[2]	12	AE16	DH[2]
13	AF15	DL[3]	14	AE14	DH[3]
15	AC14	DL[4]	16	AF14	DH[4]
17	AE13	DL[5]	18	AC13	DH[5]
19	AF13	DL[6]	20	AE12	DH[6]
21	AF12	DL[7]	22	AE11	DH[7]
23	AF11	DL[8]	24	AE10	DH[8]
25	AF10	DL[9]	26	AE9	DH[9]
27	AF9	DL[10]	28	AE8	DH[10]
29	AD8	DL[11]	30	AC7	DH[11]
31	AF8	DL[12]	32	AE7	DH[12]
33	AF7	DL[13]	34	AE6	DH[13]
35	AF6	DL[14]	36	DH[14]	DH[14]
37	AE5	DL[15]	38	DH[15]	DH[15]
Even Cable			**Odd Cable**		
Logic Analyzer Pod 8			**Logic Analyzer Pod 7**		

Mictor Connector J5					
J5 Pin	**BGA Pin**	**MPC8240 Signal**	**J5 Pin**	**BGA Pin**	**MPC8240 Signal**
1		nc	2		nc
3		nc	4		nc
5		nc	6	AC25	PCI_CLK[0]
7		nc	8		nc
9	AD18	PMAA[0]	10		Pull up Vdd (TTL)
11	AF18	PMAA[1]	12		Pull up Vdd (TTL)
13	AE19	PMAA[2]	14		Pull up Vdd (TTL)
15	AA25	$\overline{\text{REQ}}$[1]	16	AC26	$\overline{\text{INTA}}$
17	W23	$\overline{\text{GNT}}$[1]	18		Pull up Vdd (TTL)
19	K26	$\overline{\text{TRDY}}$	20	P25	C/$\overline{\text{BE}}$[3]
21	J24	$\overline{\text{FRAME}}$	22	K23	C/$\overline{\text{BE}}$[2]
23	K25	$\overline{\text{IRDY}}$	24	F23	C/$\overline{\text{BE}}$[1]
25		GND	26	A25	C/$\overline{\text{BE}}$[0]
27		Pull up Vdd (TTL)	28	H26	$\overline{\text{DEVSEL}}$
29	AB26	$\overline{\text{REQ}}$[0]	30	H25	$\overline{\text{STOP}}$
31	V26	$\overline{\text{GNT}}$[0]	32	J26	$\overline{\text{LOCK}}$
33	P26	IDSEL	34	G26	$\overline{\text{PERR}}$
35		Pull up Vdd (TTL)	36	F26	$\overline{\text{SERR}}$
37		Pull up Vdd (TTL)	38	G25	PAR
Even Cable			**Odd Cable**		

Mictor Connector J6					
J6 Pin	**BGA Pin**	**MPC8240 Signal**	**J6 Pin**	**BGA Pin**	**MPC8240 Signal**
1		nc	2		nc
3		nc	4		nc
5		nc	6		nc
7	V25	AD[31]	8	F24	AD[15]
9	U25	AD[30]	10	E26	AD[14]
11	U26	AD[29]	12	E25	AD[13]
13	U24	AD[28]	14	E23	AD[12]
15	U23	AD[27]	16	D26	AD[11]
17	T25	AD[26]	18	D25	AD[10]
19	T26	AD[25]	20	C26	AD[9]
21	R25	AD[24]	22	A26	AD[8]
23	R26	AD[23]	24	B26	AD[7]
25	N26	AD[22]	26	A24	AD[6]
27	N25	AD[21]	28	B24	AD[5]
29	N23	AD[20]	30	D19	AD[4]
31	M26	AD[19]	32	B23	AD[3]
33	M25	AD[18]	34	B22	AD[2]
35	L25	AD[17]	36	D22	AD[1]
37	L25	AD[16]	38	C22	AD[0]
Even Cable			**Odd Cable**		

Mictor Connector J7					
J7 Pin	**BGA Pin**	**MPC8240 Signal**	**J7 Pin**	**BGA Pin**	**MPC8240 Signal**
1		nc	2		nc
3		nc	4		nc
5		*	6		*
7			8	B16	$\overline{\text{SRESET}}$
9			10	B20	$\overline{\text{SUSPEND}}$
11			12	B14	TBEN
13			14	AF22	TCK
15			16	AF23	TDI
17			18	AC21	TDO
19			20	AE22	TMS
21			22	AE23	$\overline{\text{TRST}}$
23			24	C19	IRQ_0 / S_INT
25			26	B21	IRQ_1 / S_CLK
27			28	AC22	IRQ_2 / S_RST
29	A20	$\overline{\text{HRST\_CTRL}}$	30	AE24	IRQ_3 / $\overline{\text{S\_FRAME}}$
31	A19	$\overline{\text{HRST\_CPU}}$	32	A23	IRQ_4 / LINT
33	A17	$\overline{\text{MCP}}$	34	AE20	SDA*
35	D16	NMI	36	AF21	SCL*
37	A18	$\overline{\text{SMI}}$	38	D14	$\overline{\text{CHKSTOP\_IN}}$
Even Cable			**Odd Cable**		

Notes:
* You may place the SDA and SCL on pins 5 and 6, which are clock lines.

Enabling Debug Mode on the MPC8240

Full address information does not normally appear on the MPC8240 bus. To reconstruct physical addresses, the inverse assembler uses information from the 16 debug address pins, which are enabled by setting the processor to run in debug mode.

To enable debug mode in software

Set the WP_DEBUG_ bit in the Watchpoint Control Register (WP_CONTROL bit 28) to 0.

To enable debug mode in hardware

Pull down $\overline{\text{GNT}}$[4] (pin W26 on the BGA), which is sampled at reset.

Preparing for Emulation

When using the MPC8240 emulator, you need to consider how the emulator connects to the target system.

Debug Port Connection

A 2x8 0.1 inch center BERG-style connector is required to connect the emulator to the MPC8240 JTAG interface. The header should be placed as close to the microprocessor as possible to ensure signal integrity. TDO, TDI, TCK, TMS, and $\overline{\text{TRST}}$ should have signal traces less than three inches between the JTAG connector and the microprocessor. If these signals are connected to other nodes, it is required that they connect in a daisy chain between the JTAG debug connector and the MPC8240. These signals are sensitive to cross-talk and cannot be routed next to active signals such as clocks.

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

The emulator adds about 40pF to all target system signals routed to the debug connector. This added capacitance may reduce the rise time of the $\overline{\text{HRESET}}$ and $\overline{\text{SRESET}}$ signals beyond the processor specifications. If so, the target may need to increase the pull-up current on these signal lines.

When the emulator performs a 'reset', $\overline{\text{HRESET}}$ is held low for approximately 300ms.

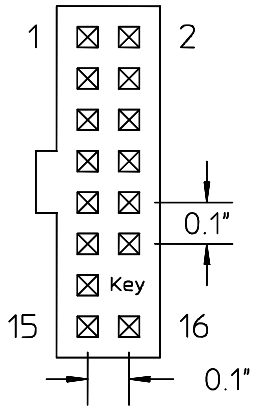
Reset Signals for the MPC8240

The $\overline{\text{SRESET}}$, $\overline{\text{HRESET}}$ signals from the JTAG connector may be logically ORed with their respective signals on the target system. The MPC8240 has two hard resets, $\overline{\text{HRST\_CPU}}$ and $\overline{\text{HRST\_CTRL}}$. $\overline{\text{HRESET}}$ from the debug connector must be routed to the $\overline{\text{HRST\_CPU}}$ reset logic and optionally routed to the $\overline{\text{HRST\_CTRL}}$ reset logic depending on system requirements. The analysis probe can be configured to connect $\overline{\text{HRST\_CPU}}$ and $\overline{\text{HRST\_CTRL}}$ together (page 96).

The emulator drives $\overline{\text{SRESET}}$ and $\overline{\text{HRESET}}$ with open-drain drivers using 2.7 KΩ pullups to VDD. The target system designer may take advantage of these open-drain drivers by wire-ORing $\overline{\text{SRESET}}$ and/or $\overline{\text{HRESET}}$ to open-drain drivers on the target system. It is not necessary to use a wire-OR configuration, but reset status messages can only be generated by the emulator when using the wire-ORed configuration.

The $\overline{\text{TRST}}$ signal from the JTAG connector must be logically ORed with $\overline{\text{TRST}}$ on the target system. $\overline{\text{TRST}}$ is actively driven by the emulator and cannot be wire-ORed.

JTAG Connector Pinout and Electrical Information



Pin	Signal	Resistor
1	TDO	
2	NC	
3	TDI	1 KΩ pull-down
4	$\overline{\text{TRST}}$	10 KΩ pull-up
5	NC	
6	+3.3V	1 KΩ series
7	TCK	10 KΩ pull-up
8	NC	
9	TMS	10 KΩ pull-up
10	NC	
11	$\overline{\text{SRESET}}$	10 KΩ pull-up
12	NC	
13	$\overline{\text{HRESET}}$	10 KΩ pull-up
14	NC	Key (no pin)
15	$\overline{\text{CHKSTP}}$	10 KΩ pull-up
16	GND	

3

Setting Up the Logic Analysis System

Power-ON/Power-OFF Sequence

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

To power-ON the HP 16600A/16700A-series logic analysis systems

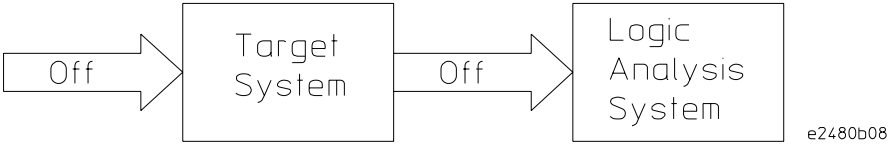
Ensure the target system is powered off.

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 logic analyzer is connected to the target system, and everything is configured, turn on your target system.

To power-OFF

Turn off power to your system in the following order:

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



Installing Logic Analyzer Modules

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

Refer to the HP 16600A/16700A-series logic analysis system's *Installation Guide*.

Installing the Emulation Module

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

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

To install in a HP 16700A-series logic analysis system

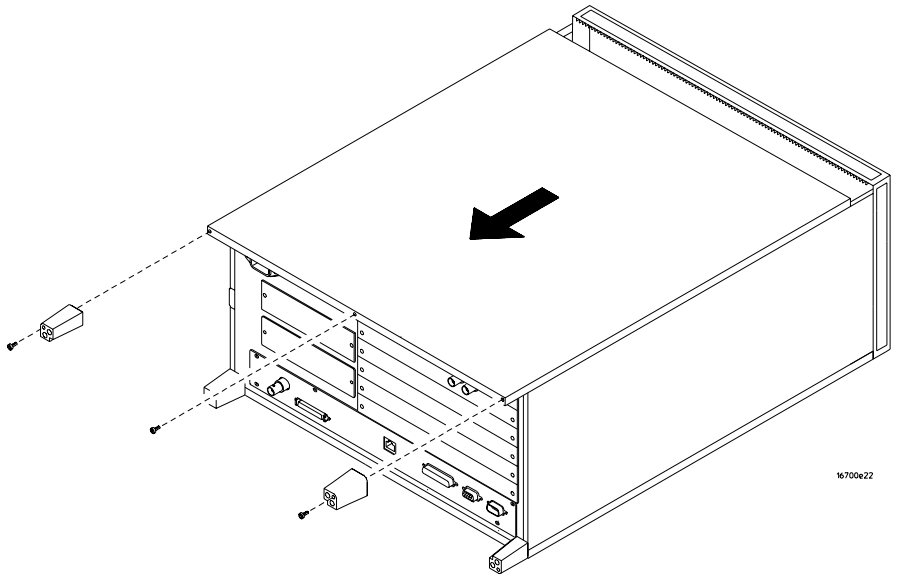
Or, to install in an HP 16701A expansion frame:

You will need T-10 and T-15 Torx screw drivers.

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

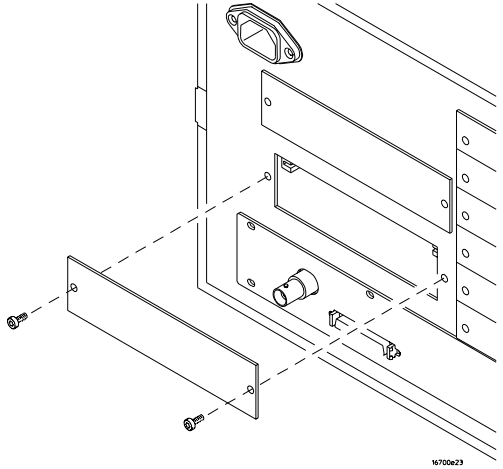
   Remove any other cables (including mouse or video monitor cables).

2. Turn the logic analysis system frame upside-down.
3. Remove the bottom cover.

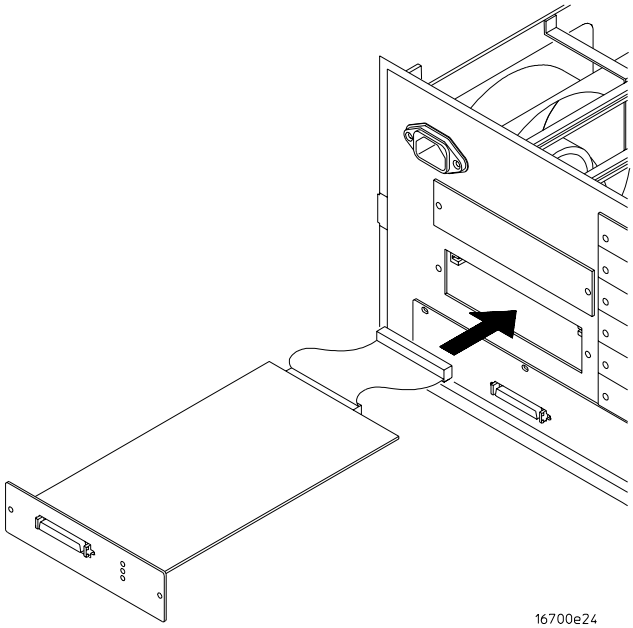


4 Remove the slot cover.

You may use either slot.

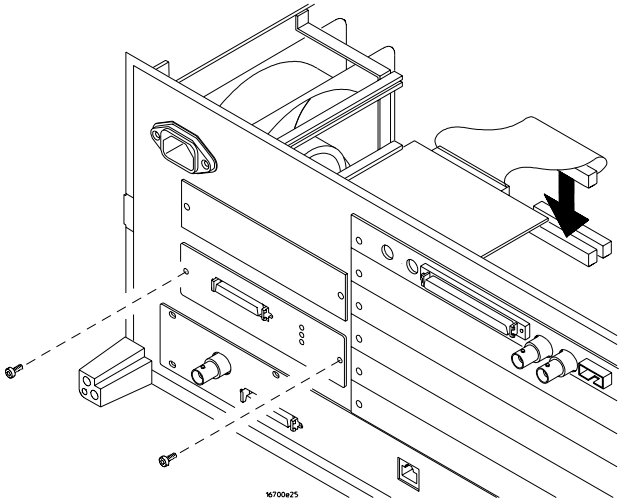


5 Install the emulation module.



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 "To update emulation module firmware" on page 99 for information on giving the emulation module a "personality" for your target processor.

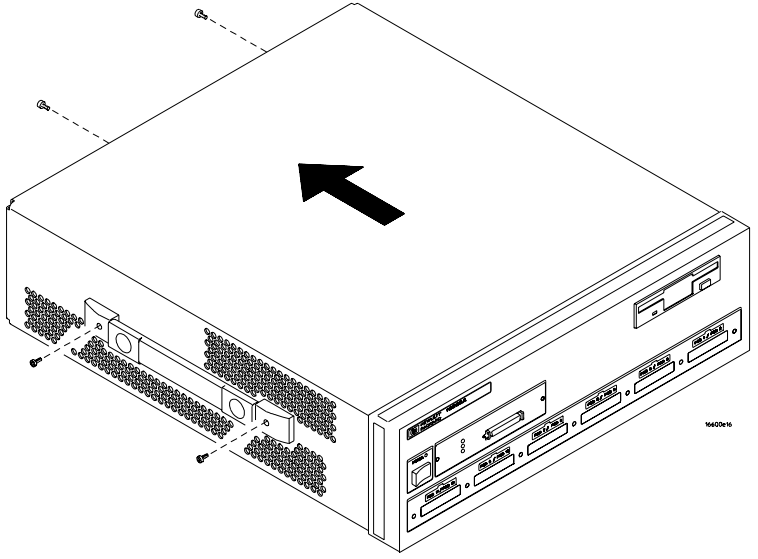
To install in a HP 16600A-series logic analysis system

You will need T-8, T-10, and T-15 Torx screw drivers.

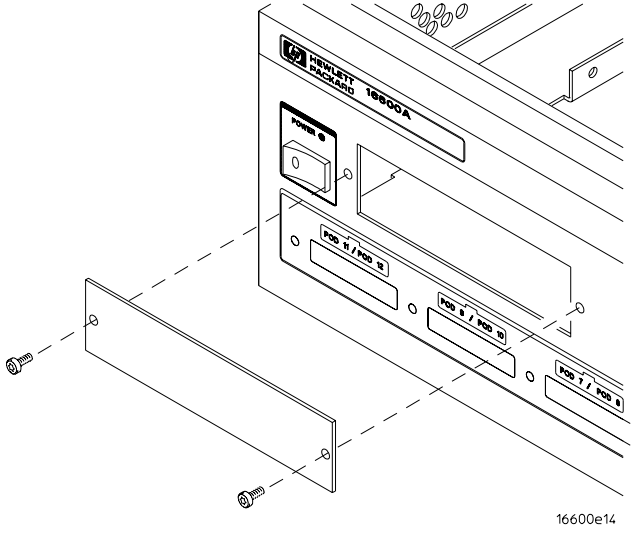
1 Turn off the logic analysis system and REMOVE THE POWER CORD.

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

2 Slide the cover back.

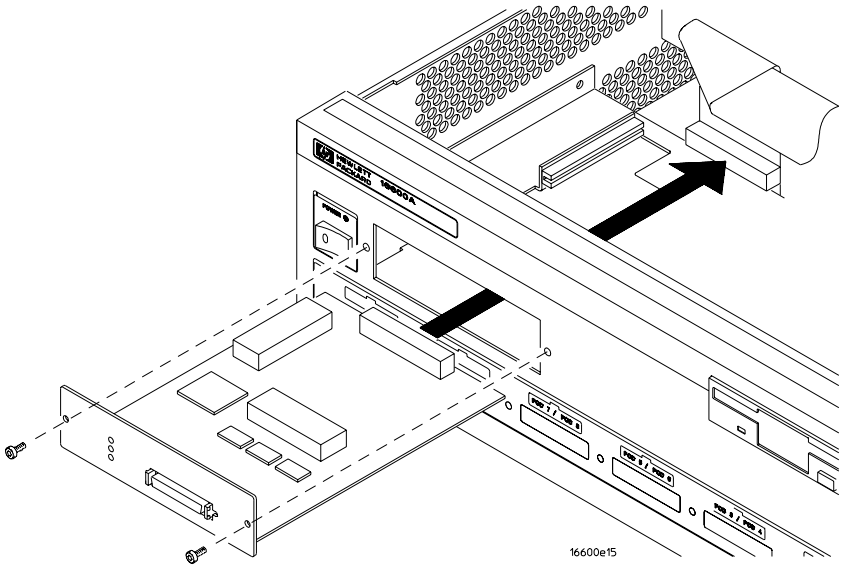


3 Remove the slot cover.



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 "To update emulation module firmware" on page 99 for information on giving the emulation module a "personality" for your target processor.

To test the emulation module

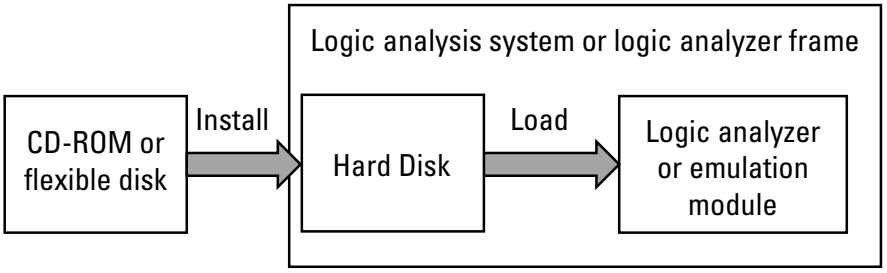
If this is the first time that you have used the emulation module, you should run the built-in performance verification tests before you connect to a target system. Refer to "Troubleshooting the Emulation Module" on page 235 for information on performance verification.

Installing Software

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

Installing and loading

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



What needs to be installed

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

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

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

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

To install software from CD-ROM

Installing a processor support package from a CD-ROM will take just a few minutes. If the processor support package requires an update to the HP 16600A/16700A-series logic analysis system's operating system, installation may take approximately 15 minutes.

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

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

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

2. Insert the CD-ROM in the drive.
3. Click the **System Admin** icon.
4. Click the **Software Install** tab.
5. Click **Install...** .

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

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

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

8. Click on the "MPC82XX" package.

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

9. Click **Install**.

   The Continue dialog box will appear.

10. Select **Continue**.

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

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

software installation windows.

The configuration files are stored in /hplogic/configs/hp/*processor*.

The inverse assemblers are stored in /hplogic/ia.

See Also

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

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

4

Probing the Target System

Using the Setup Assistant

The Setup Assistant is an on-line 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 target system to a logic analyzer, an emulation module, or other supported equipment.

Start the Setup Assistant by clicking its icon in the system window.

Connecting the Logic Analyzer to the Target System

Disconnect power from the logic analyzer and your target system before you make or break connections. (If you have an emulation probe instead of an emulation module, also disconnect its power.)

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 J7 contains additional signals you might want to monitor.

- HP 16550A logic analyzer (two cards)—page 78
- HP 16554/55/56/57 logic analyzers (two cards)—page 81
- HP 16600A logic analysis system—page 83
- HP 16601A logic analysis system—page 85
- HP 16710/11/12A logic analyzers (two cards)—page 88
- HP 16715/16/17A logic analyzers (two cards)—page 90

32-bit data

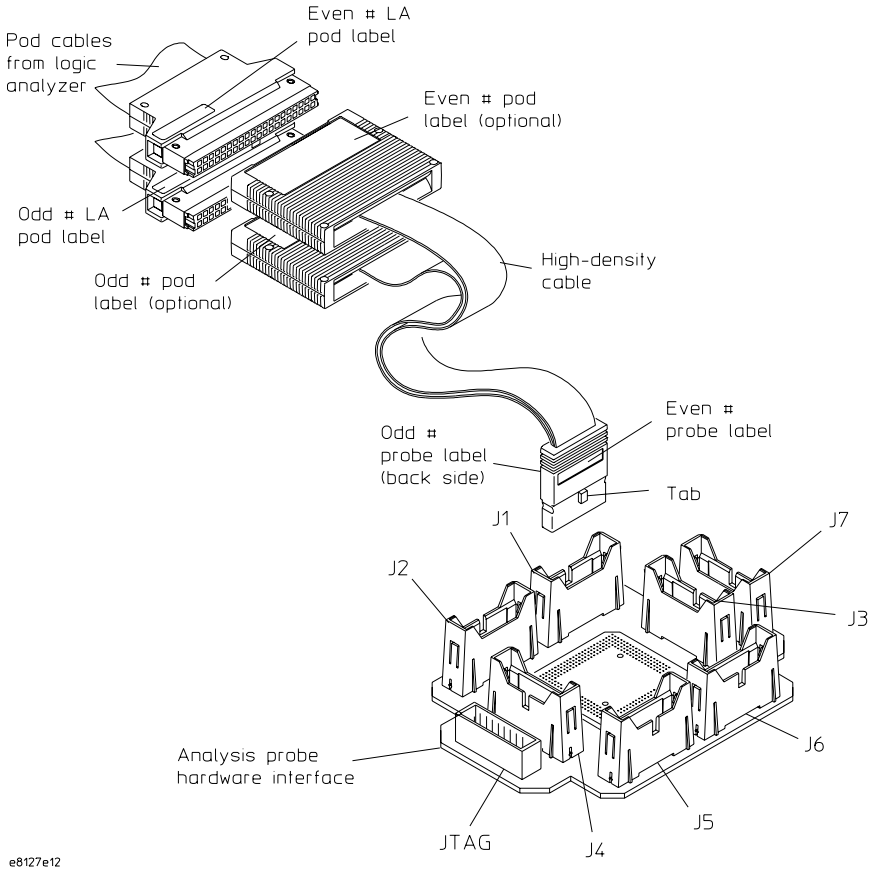
This type of analysis will allow you to trace the 32 bits of DATA only, not DATA_B. The connections and configuration files are the same as for 64-bit data.

No data

This type of analysis allows you to trace program flow only using J1 and J2. Opcodes in the inverse assembly listing will be taken from an 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.

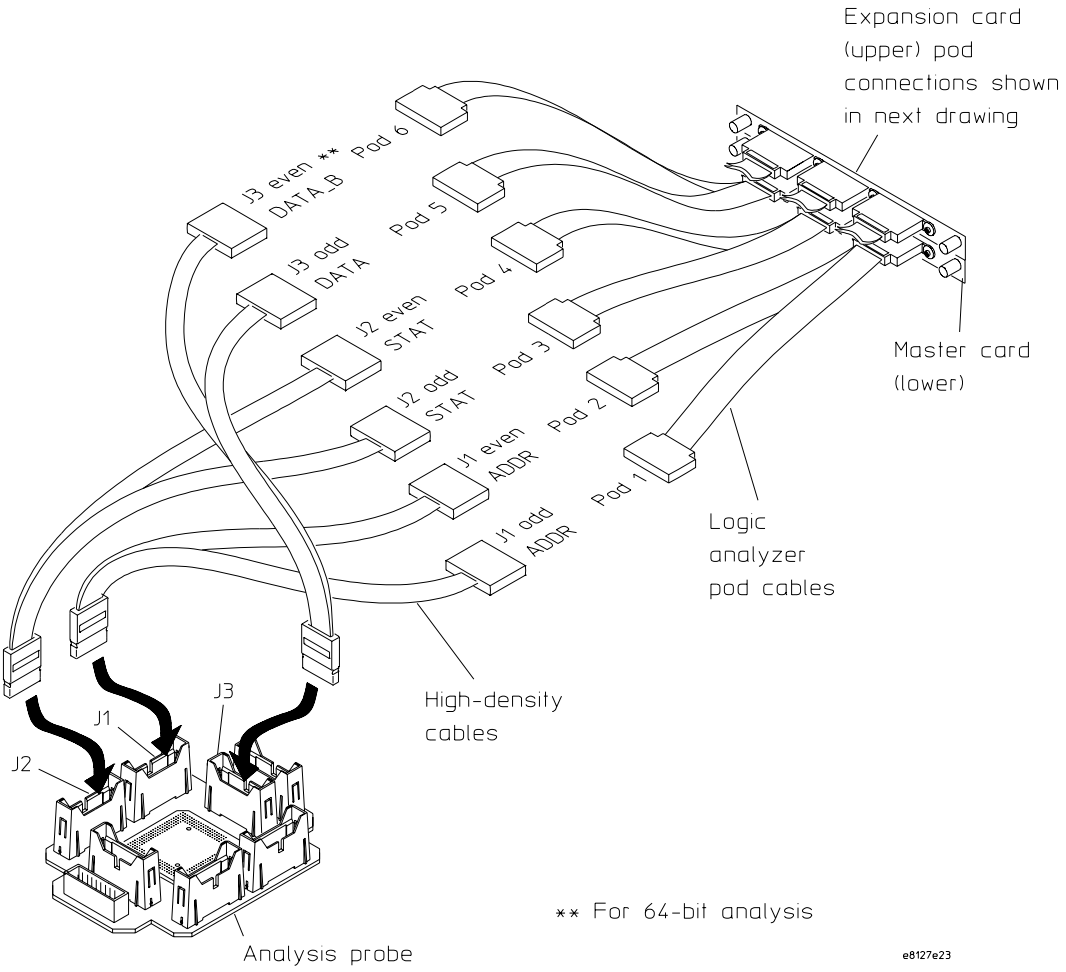
To connect the high-density termination cables to the analysis probe

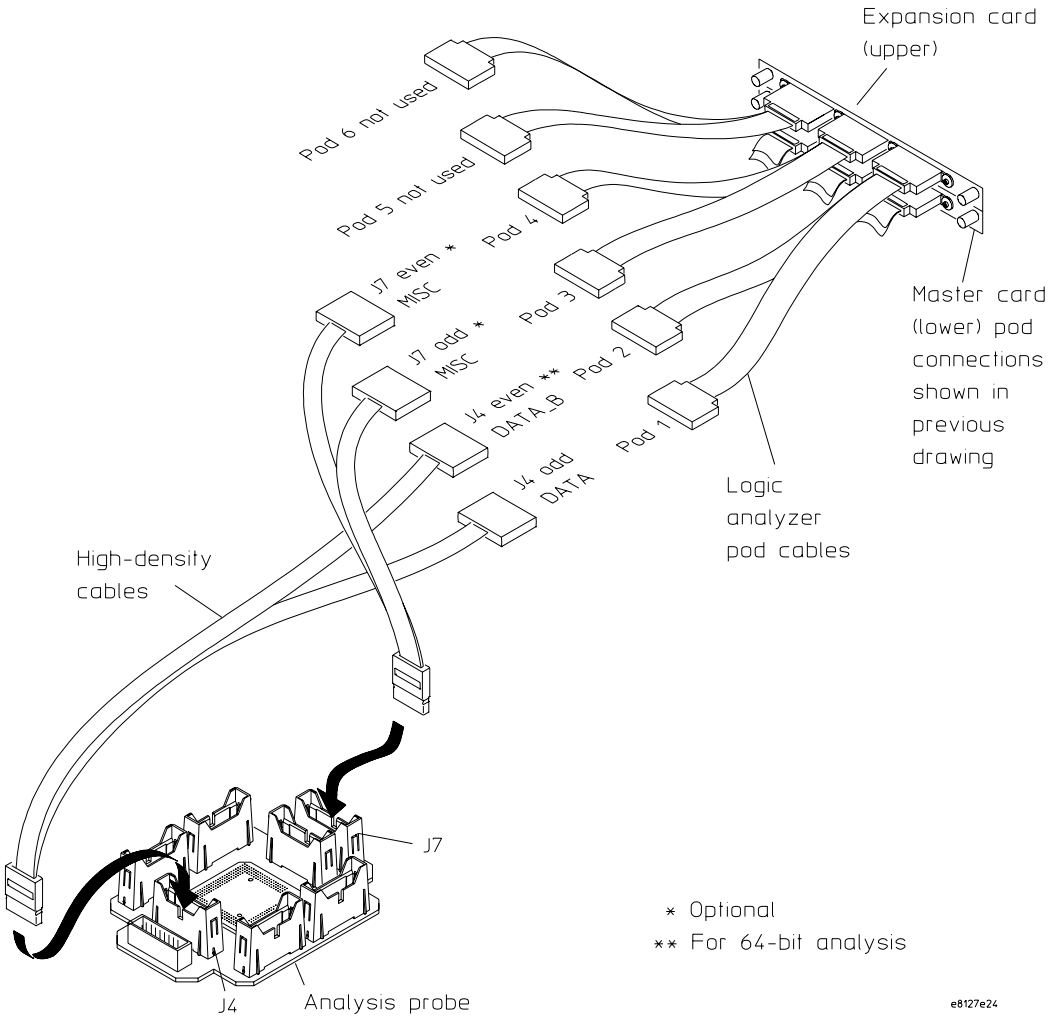
Four HP E5346A high-density termination cables, and labels to identify them, are included with the analysis probe. Connect the cables to the connectors on the analysis probe as shown in the illustration below. Attach the labels to the cables after connecting the cables to the logic analyzer.



To connect a two-card HP 16550A logic analyzer for 64-bit or 32-bit data analysis

Use the following figures to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system. Find the labels that were shipped with the high-density cables and use them to help identify the connections.

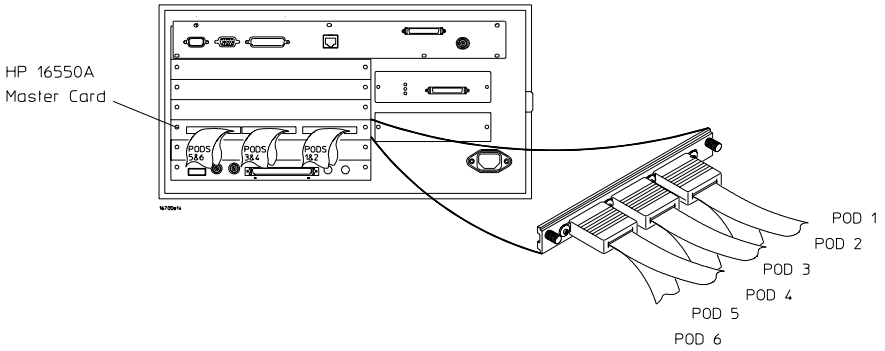




Configuration File

Use configuration file c8240F_2.

To connect an HP 16550A logic analyzer for no-data analysis



Use this table to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system.

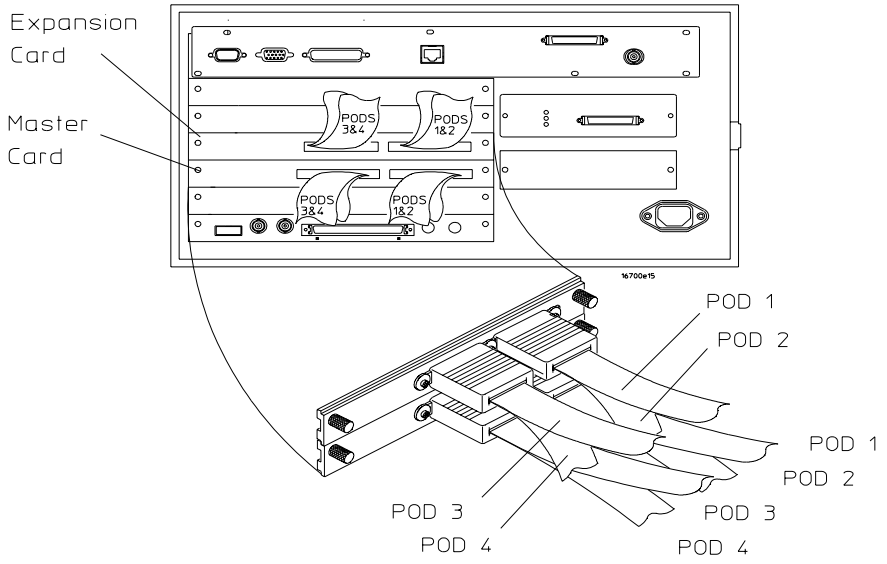
	Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
Master Card	J7, Even[1]	J7, Odd[1]	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional MISC signals

Configuration File

Use configuration file c8240F_1.

To connect a two-card HP 16554/55/56/57 logic analyzer for 64-bit or 32-bit data analysis



Use this table to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system.

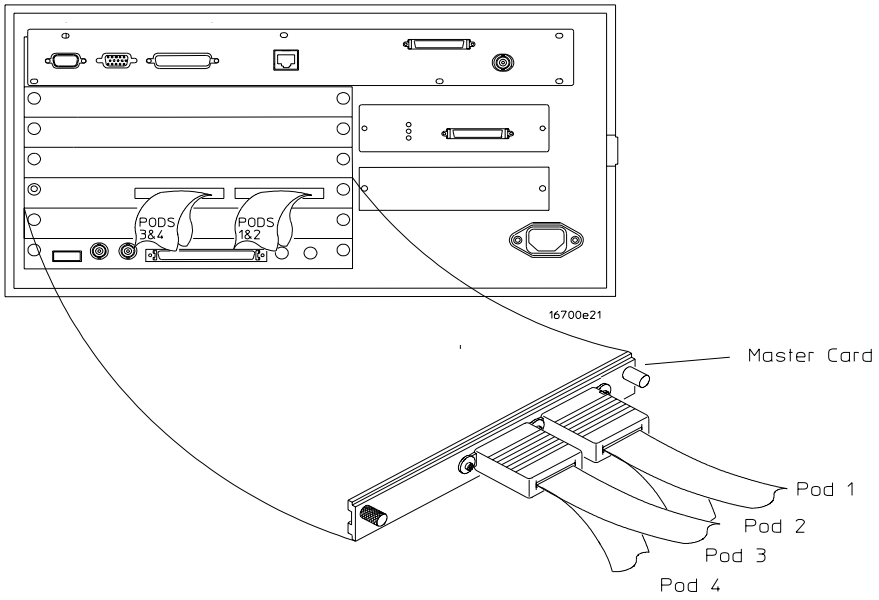
	Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	J4, Even[1]	J4, Odd	J3, Even[1]	J3, Odd
Master Card	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional DATA_B signals for 64-bit analysis

Configuration File

Use configuration file c8240M_2.

To connect an HP 16554/55/56/57 logic analyzer for no-data analysis



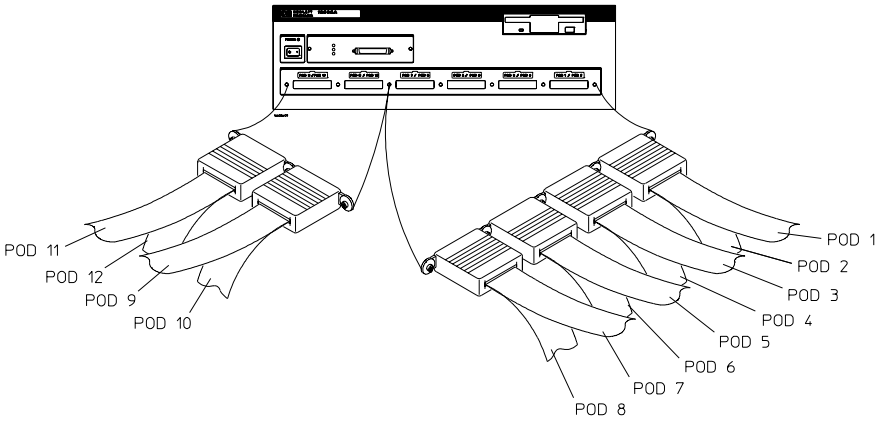
Use this table to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system.

	Pod 4	Pod 3	Pod 2	Pod 1
Master Card	J2, Even	J2, Odd	J1, Even	J1, Odd

Configuration File

Use configuration file c8240M_1.

To connect the HP 16600A logic analyzer for 64-bit or 32-bit data analysis



Use this table to connect cables from the HP 16600A logic analyzer to the connectors on the analysis probe or on the target system.

Pod 12	Pod 11	Pod 10	Pod 9	Pod 8	Pod 7
unused	unused	J7, Even[1]	J7, Odd[1]	J4, Even[2]	J4, Odd

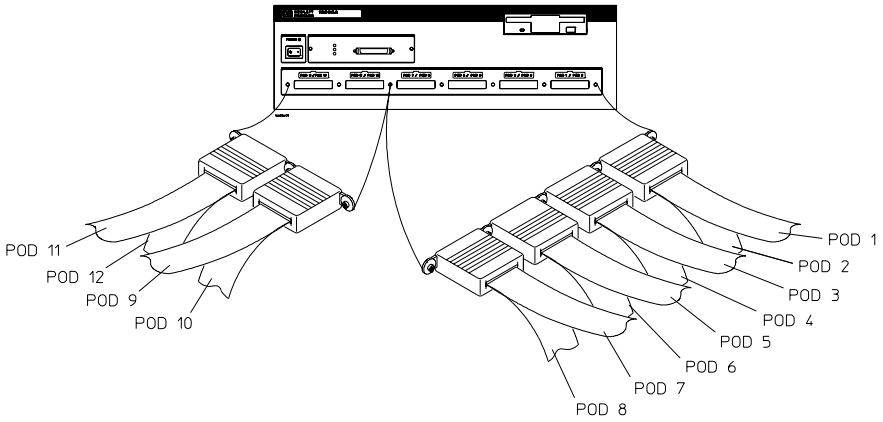
Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
J3, Even[2]	J3, Odd	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional MISC signals
2 Optional DATA_B signals for 64-bit analysis

Configuration File

Use configuration file c8240F_2.

To connect the HP 16600A logic analyzer for no-data analysis



Use this table to connect cables from the HP 16600A logic analyzer to the connectors on the analysis probe or on the target system.

Pod 12	Pod 11	Pod 10	Pod 9	Pod 8	Pod 7
unused	unused	unused	unused	unused	unused

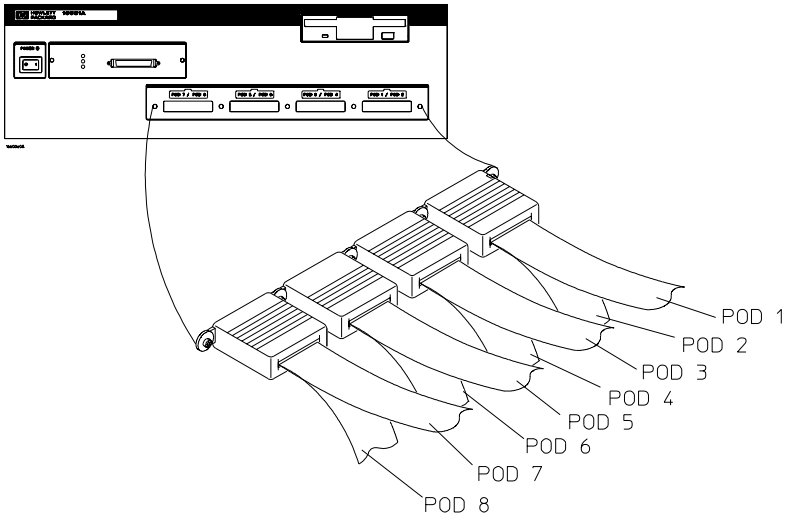
Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
J7, Even[1]	J7, Odd[1]	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional MISC signals

Configuration File

Use configuration file c8240F_1.

To connect the HP 16601A logic analyzer for 64-bit or 32-bit data analysis



Use this table to connect cables from the HP 16601A logic analyzer to the connectors on the analysis probe or on the target system.

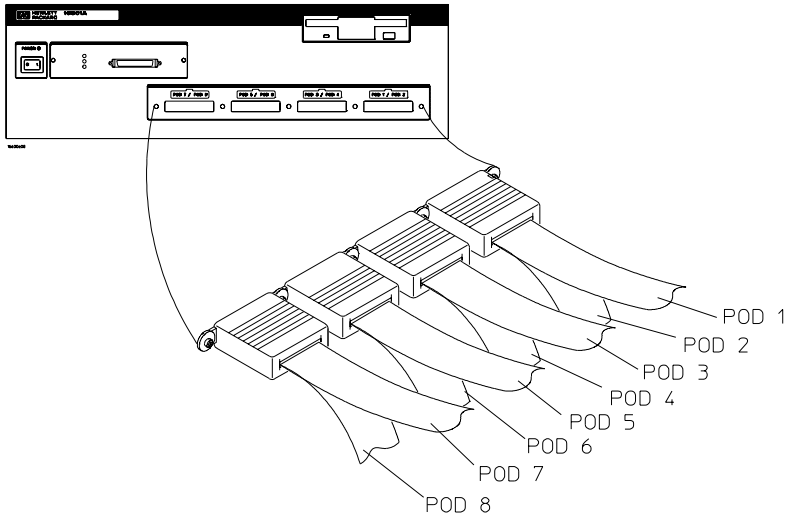
Pod 8	Pod 7	Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
J4, Even[1]	J4, Odd	J3, Even[1]	J3, Odd	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1Optional DATA_B signals for 64-bit analysis

Configuration File

Use configuration file c8240F_2.

To connect the HP 16601A logic analyzer for no-data analysis



Use this table to connect cables from the HP 16601A logic analyzer to the connectors on the analysis probe or on the target system.

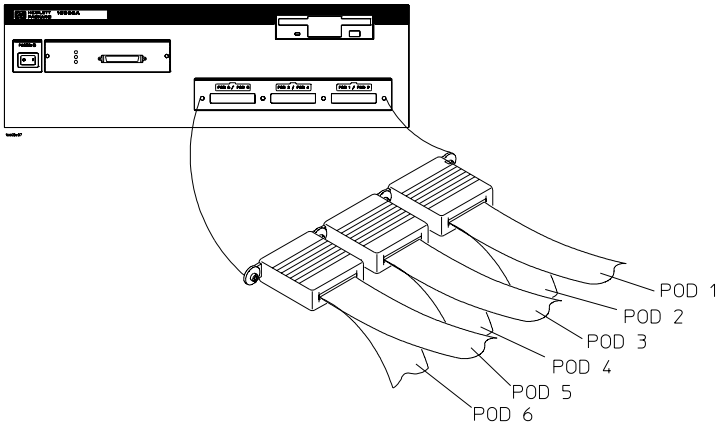
Pod 8	Pod 7	Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
unused	unused	J7, Even[1]	J7, Odd[1]	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional MISC signals

Configuration File

Use configuration file c8240F_1.

To connect the HP 16602/3A logic analyzer for no-data analysis



Use this table to connect cables from the HP 16602/3A logic analyzer to the connectors on the analysis probe or on the target system.

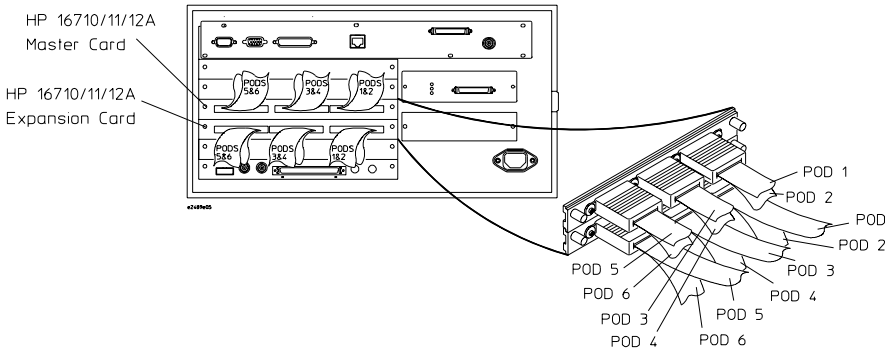
Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
J7, Even[1]	J7, Odd[1]	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional MISC signals (HP 16602A only)

Configuration File

Use configuration file c8240F_1.

To connect a two-card HP 16710/11/12A or logic analyzer for 64-bit or 32-bit data analysis



Use this table to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system.

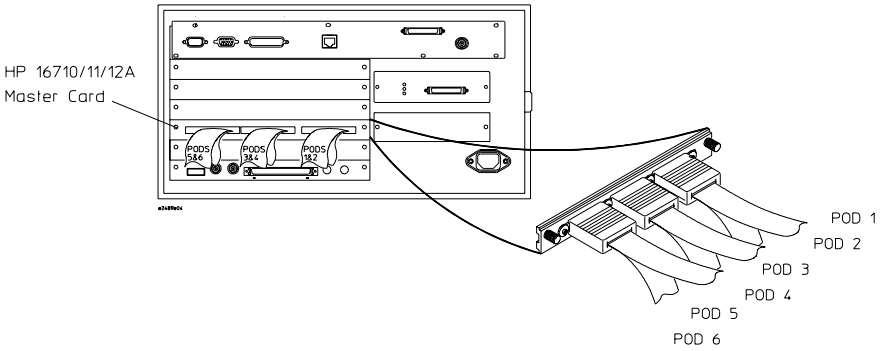
	Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	unused	unused	J7, Even[1]	J7, Odd[1]	J4, Even[2]	J4, Odd
Master Card	J3, Even[2]	J3, Odd	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional MISC signals
2 Optional DATA_B signals for 64-bit analysis

Configuration File

Use configuration file c8240F_2.

To connect an HP 16710/11/12A logic analyzer for no-data analysis



Use this table to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system.

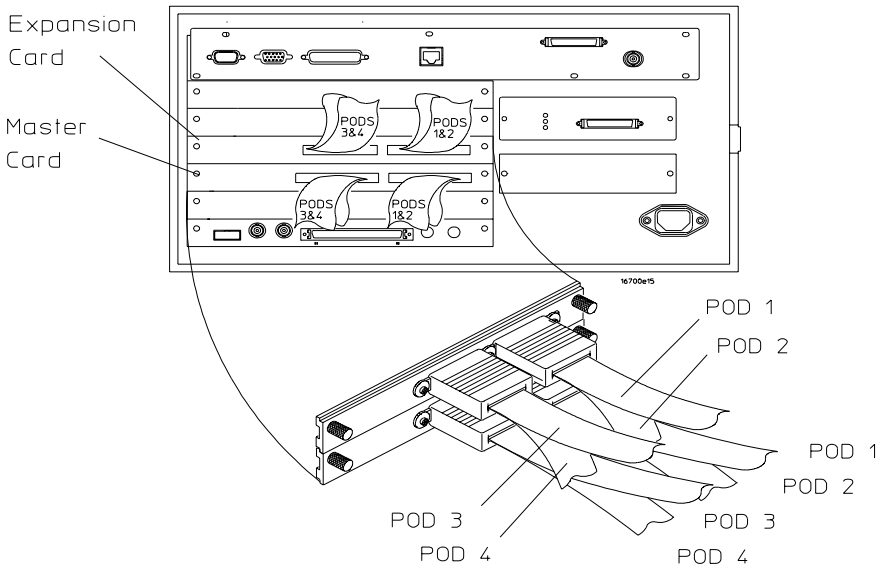
	Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
Master Card	J7, Even[1]	J7, Odd[1]	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional MISC signals

Configuration File

Use configuration file c8240F_1.

To connect a two-card HP 16715/16/17A logic analyzer for 64-bit or 32-bit data analysis



Use this table to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system.

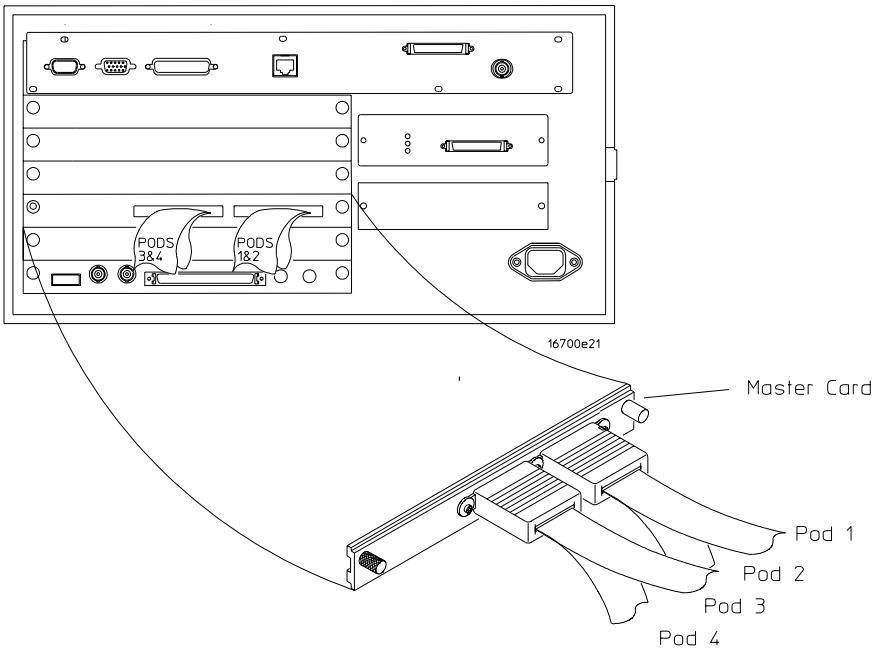
	Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	J4, Even[1]	J4, Odd	J3, Even[1]	J3, Odd
Master Card	J2, Even	J2, Odd	J1, Even	J1, Odd

Notes:
1 Optional DATA_B signals for 64-bit analysis

Configuration File

Use configuration file c8240L_2.

To connect an HP 16715/16/17A logic analyzer for no-data analysis



Use this table to connect cables from the logic analyzer to the connectors on the analysis probe or on the target system.

	Pod 4	Pod 3	Pod 2	Pod 1
Master Card	J2, Even	J2, Odd	J1, Even	J1, Odd

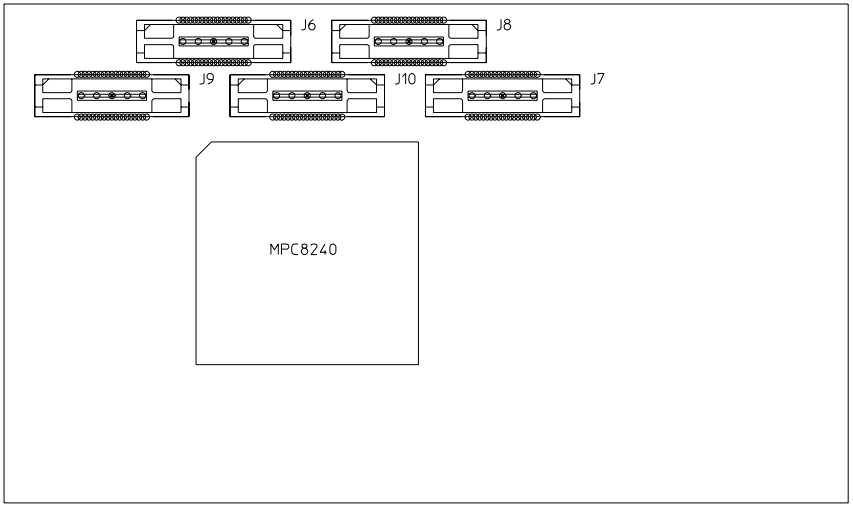
Configuration File

Use configuration file c8240L_1.

Connecting the Logic Analyzer to a Motorola PMC8240 Board

Disconnect power from the logic analyzer and your target system before you make or break connections. (If you have an emulation probe instead of an emulation module, also disconnect its power.)

This section shows the connections between the logic analyzer pod cables and the connectors on the PMC8240 board. The 32-bit and no-data analysis options are not available for this board.



e8127e04

To connect a two-card HP 16710/11/12A or HP 16550A logic analyzer to a PMC8240 board

	Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	unused	unused	unused	unused	J6, Even	J6, Odd
Master Card	J9, Even	J9, Odd	J10, Even	J10, Odd	J7, Even	J7, Odd

Configuration File

Use configuration file cPMC8240F_2.

To connect an HP 16600A or HP 16601A logic analyzer to a PMC8240 board

Pod 8	Pod 7	Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
J6, Even	J6, Odd	J9, Even	J9, Odd	J10, Even	J10, Odd	J7, Even	J7, Odd

Configuration File

Use configuration file cPMC8240F_2.

To connect a two-card HP 16554/55/56/57 logic analyzer to a PMC8240 board

	Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	J6, Even	J6, Odd[1]	J9, Even	J9, Odd
Master Card	J10, Even	J10, Odd	J7, Even	J7, Odd

Configuration File

Use configuration file cPMC8240M_2.

To connect a two-card HP 16715/16/17A logic analyzer to a PMC8240 board

	Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	J6, Even	J6, Odd[1]	J9, Even	J9, Odd
Master Card	J10, Even	J10, Odd	J7, Even	J7, Odd

Configuration File

Use configuration file cPMC8240L_2.

Connecting the Emulation Module to the Target System

This section shows you how to connect the emulation module to the JTAG connector on the analysis probe or on the target system. After you have connected the emulation module to your target system, you may need to update the firmware in the emulation module.

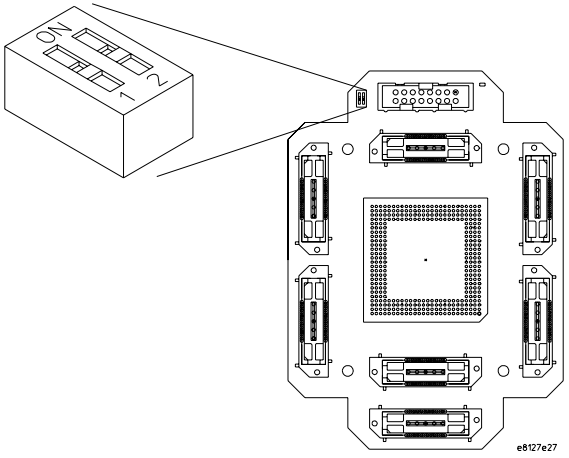
See Also

For information on designing a JTAG port on your target board, see "Preparing for Emulation" on page 58.

For a list of the parts supplied with the emulation module, see "Emulation Module" on page 25.

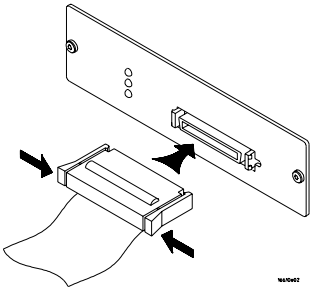
To set the analysis probe DIP switches

Switch	Description	OFF (default)	ON
1	JTAG Control	Microprocessor's JTAG port is isolated to the analysis probe's JTAG port. (default)	Microprocessor's JTAG port is connected to both the analysis probe and the target system's JTAG chain.
2	$\overline{\text{HRST}}$ Connection	$\overline{\text{HRST\_CPU}}$ is not connected to $\overline{\text{HRST\_CTRL}}$.	$\overline{\text{HRST\_CPU}}$ is connected to $\overline{\text{HRST\_CTRL}}$. The $\overline{\text{HRESET}}$ signal on the JTAG port will control both of these signals together. (default)



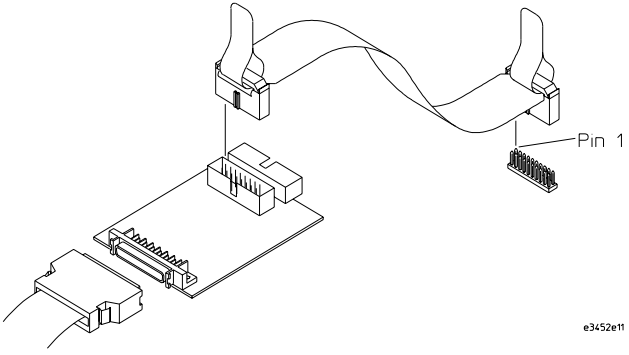
To connect to a JTAG/COP port on the target system or analysis probe

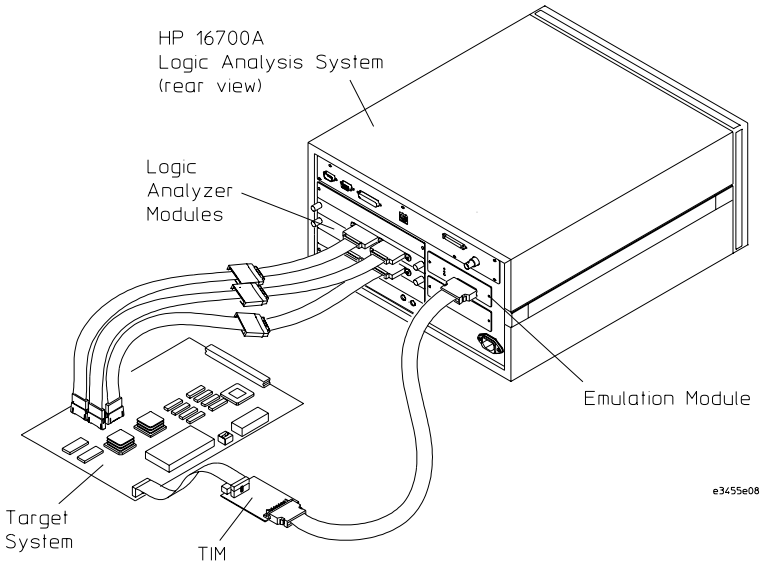
1. Turn off the target system and disconnect it from all power sources.
2. Turn off power to the logic analysis system.
3. Plug one end of the 50-pin cable into the emulation module.



4. Plug the other end of the 50-pin cable into the target interface module.
5. Plug one end of the 16-pin cable into the target interface module.
6. Plug the other end of the 16-pin cable into the JTAG port connector on the target system or analysis probe.

CAUTION: Be careful to orient the connector as shown. *The red wire does not correspond to Pin 1.* If the connector is rotated, your target system or the emulator may be damaged.





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

8 If you are connecting to a JTAG port on the target sytem, and an analysis probe is connected to the target system, check that the analysis probe has power.

The LED next to the JTAG port should be lighted. If the LED is not lighted, connect at least one logic analyzer cable to the analysis probe to provide power.

See Also For information on designing a target system for use with the emulation module, see “Preparing for Emulation” on page 58.

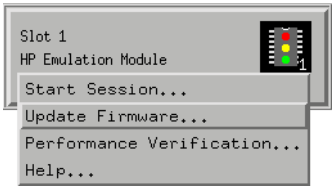
To update emulation module firmware

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 target interface module (TIM).
- The emulation module was not shipped already installed in the logic analysis system.
- You have an updated version of the firmware from HP.

To update the firmware:

1. End any Emulation Control Interface 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. In the Update Firmware window, select the firmware version to load into the emulation module.
6. 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

For instructions on how to install the firmware files on the hard disk, see "Installing Software" on page 70.

To display the emulation module firmware version information

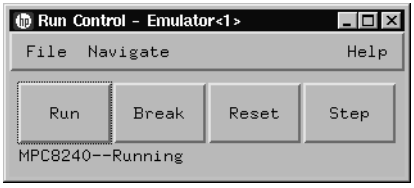
- In the Update Firmware window, click Display Current Version.

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

To verify communication with the 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 TIM match your target processor, the Run Control window should be displayed:



Part 2

Using the Logic Analyzer

5

Configuring the Logic Analyzer

Loading Configuration Files

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, how signals are routed to the connectors in the target system, and the type of analysis (64-bit data, 32-bit data, or no-data).

The MPC8240 inverse assembler decodes captured data into software addresses (SW_ADDR label) and assembly language mnemonics.

To load configuration files (and the inverse assembler)

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/mpc82xx/mpc8240 exists.

> If the above directory does not exist, you need to install the MPC82XX Processor Support Package. See "To install software from CD-ROM" on page 71.

2 Using File Manager, select the configuration file you want to load from the /hplogic/configs/hp/mpc82xx/mpc8240 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 MPC8240 analysis by loading the appropriate MPC8240 configuration file. Loading configuration files also automatically loads the inverse assembler.

3 Close File Manager.

Logic Analyzer Configuration Files

Analyzer Model	64/32-bit Configuration File	No-data Configuration File	PMC8240 Configuration File
HP 16550A	c8240F_2	c8240F_1	cPMC8240F_2
HP 16554A	c8240M_2	c8240M_1	cPMC8240M_2
HP 16555A/D	c8240M_2	c8240M_1	cPMC8240M_2
HP 16556A/D	c8240M_2	c8240M_1	cPMC8240M_2
HP 16557D	c8240M_2	c8240M_1	cPMC8240M_2
HP 16600A	c8240F_2	c8240F_1	cPMC8240F_2
HP 16601A	c8240F_2	c8240F_1	cPMC8240F_2
HP 16602A	na	c8240F_1	na
HP 16603A	na	c8240F_1	na
HP 16710/11/12A	c8240F_2	c8240F_1	cPMC8240F_2
HP 16715/16/17A	c8240L_2	c8240L_1	cPMC8240L_2

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.

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

Example program exception routine:

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

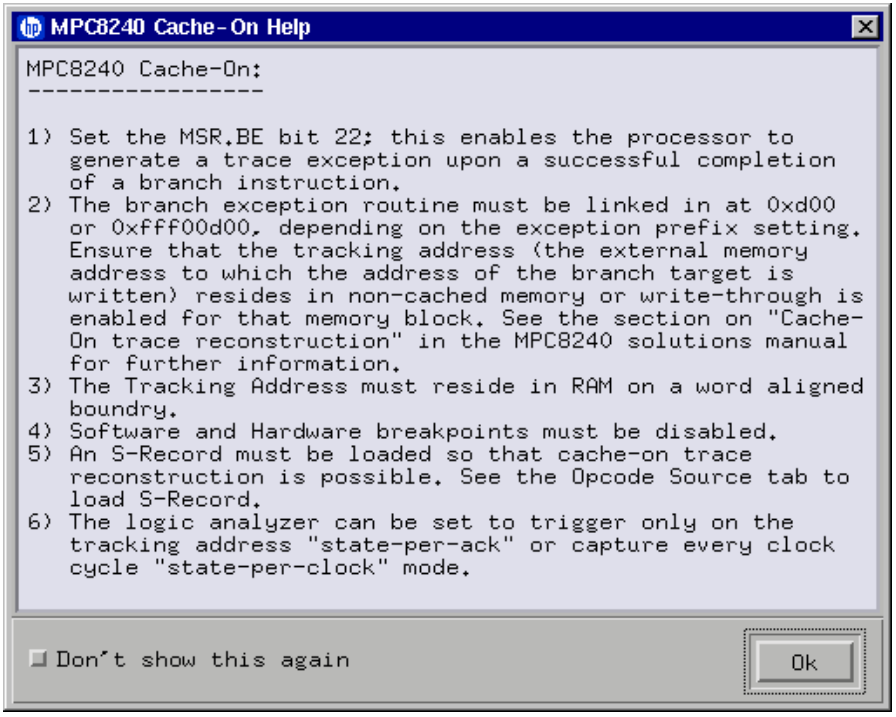
To enable cache-on trace reconstruction:

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

1. Set the cache-on mode
2. Set data bus connected
3. Provide the tracking address in the Opcode Source tab
4. Load an S-Record executable file

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

Cache-on help dialog



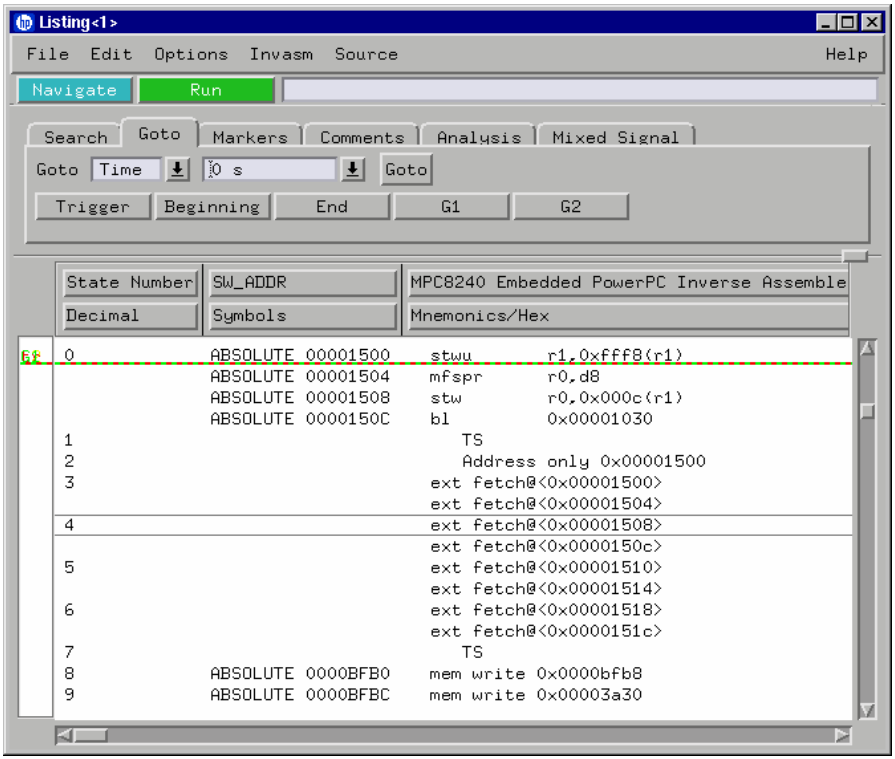
The Don't shot this again button can be selected to prevent this dialog from appearing until the inverse assembler is reloaded.

Enabling branch exception disassembly

The following trace shows cache-on execution using branch trace exception disassembly. See page 106 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 0x00001000



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 Inverse Assembler Preferences

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

Trigger Tool

The Preferences dialog also contains the Trigger Tool, which you will use when setting logic analyzer triggers for memory addresses. See “To trigger on an access to a memory address” on page 141.

To set the inverse assembler preferences

To open the Preferences dialog:

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

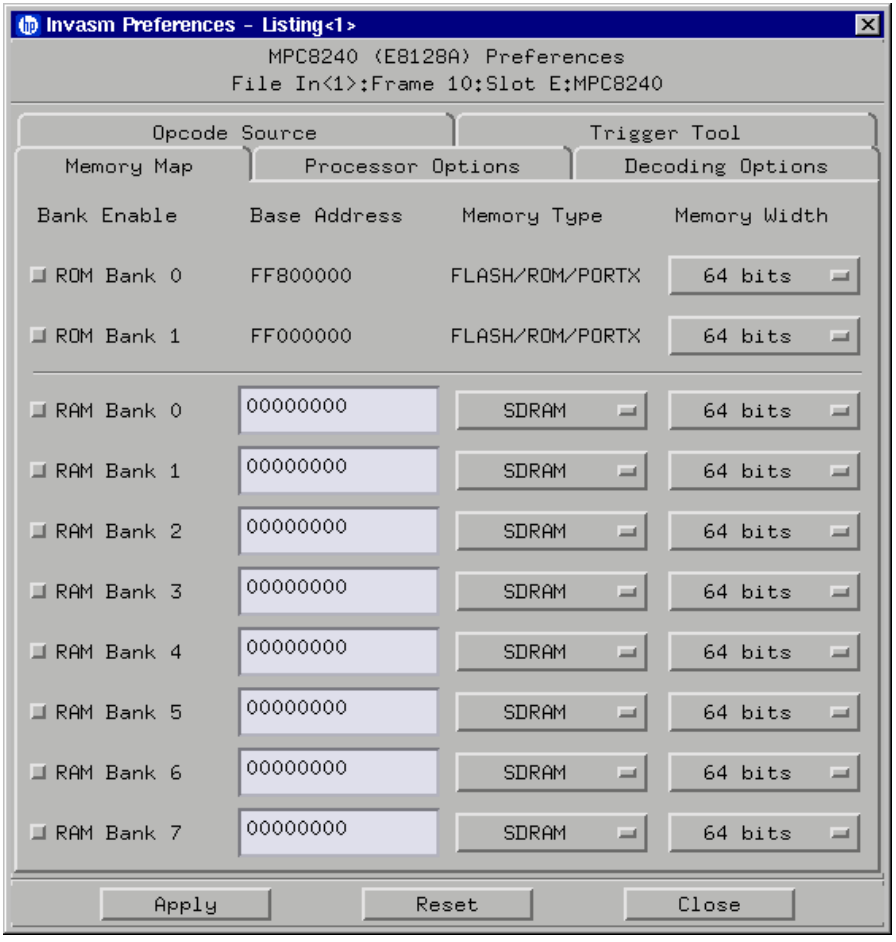
To set the Memory Map preferences

The MPC8240 does not provide all of the signals required to reconstruct physical addresses. It is therefore necessary to reconstruct the address from various pieces of information:

- Memory map configuration (set in the Preferences dialog)
- The 8 RAS_ and 8 CAS_ signals
- The 16 debug address signals (present when the MPC8240 debug mode is enabled)
- 11 SDMA signals

NOTE: You can enable debug mode by setting WP_CONTROL (WP_DEBUG_) to zero or pulling down the GNT_[4] pin.

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



Click Apply when you have finished configuring the memory map.

Bank Enable/Disable. Enable the banks for all of the chip selects that are being used in the system; otherwise, the inverse assembler may not function correctly. These enables are required because chip selects are active low and unconnected logic analyzer channels float low; without the enables, the inverse assembler cannot tell the difference between active and unconnected chip selects.

You can find out which banks are enabled by examining the Memory Bank Enable Register (MER). Each bit in this register corresponds to a memory bank.

Memory Bank Enable Register		
Register	**Address**	**Size**
Memory Bank Enable Register	0xA0	1 byte

bit 7	6	5	4	3	2	1	bit 0
bank 7	bank 6	bank 5	bank 4	bank 3	bank 2	bank 1	bank 0
1 = bank is enabled		0 = bank is disabled					

Examples: reading the Bank Enable register using the emulator's built-in command line interface:

Map A, Big Endian

```
m -a4 -d4 80000CF8=A0000080
m -a1 80000CFC
```

In this case, the gateway register is 0x80000CFC the and address register is 0x80000CF8. The first command sets the configuration register address to **A0** (the Memory Bank Enable Register). The second command displays the value of this 1-byte register.

Map B, Big Endian

```
m -a4 -d4 FEC00000=A0000080
m -a1 FEE00000
```

In this case, the gateway register is 0xFEE00000 and the address register is 0xFEC00000.

Base Address. Specifies the starting address of the memory bank.

You can find out the base address by examining the Memory Starting Address register and the Extended Memory Starting Address register.

Memory Starting Address Registers		
Register	**Address**	**Size**
Memory Starting Address Register for banks 0-3	0x80	4 bytes
Memory Starting Address Register for banks 4-7	0x84	4 bytes
Extended Memory Starting Address Register for banks 0-3	0x88	4 bytes
Extended Memory Starting Address Register for banks 4-7	0x8C	4 bytes

These values are combined into a base address as follows

bit	31	30	29	28	27 20	19 0
	0	0	extended starting address		starting address	0x00000

Examples: reading a Starting Address register using software

Map A, Big Endian

First set:

```
r0 = 0x80000080
r1 = 0x80000CF8
```

Next, run this code:

```
stw r0,0(r1)
lwz r2,4(r1)
```

This will set the address (0x800000CF8) to 0x**80**000080. r2 will be set to the value of the configuration register at 0x**80** (the value might be something like 0x00102030).

Map B, Big Endian

First, set:

```
r0 = 0x80000080
r1 = 0xFEC00000
r2 = 0xFEE00000
```

Next, run this code:

```
stw r0,0(r1)
lwz r3,0(r2)
```

This sets the address (0xFEC00000) to 0x**80**000080. r3 will be set to the value of the configuration register at 0x**80**.

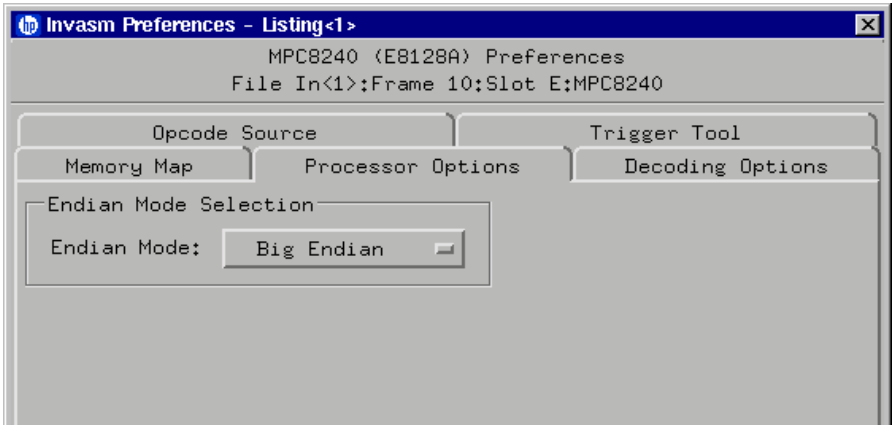
Memory Type. Depending upon the type of memory that is accessed, there are different signals that assert when a valid address is on the bus. For this reason, the inverse assembler must know what type of memory is being accessed.

You can find out the memory type for the RAM banks by examining bit 17 of the Memory Control Configuration register at 0xF0. “0” indicates SDRAM and “1” indicates FPM/EDO DRAM.

The MPC8240 does not support mixed DRAM/SDRAM configurations, so all of the memory banks will be configured to the same memory type.

To set the Processor Options preferences

The Processor Options tab of the Preferences dialog lets you tell the inverse assembler which mode the MPC8240 is operating in.



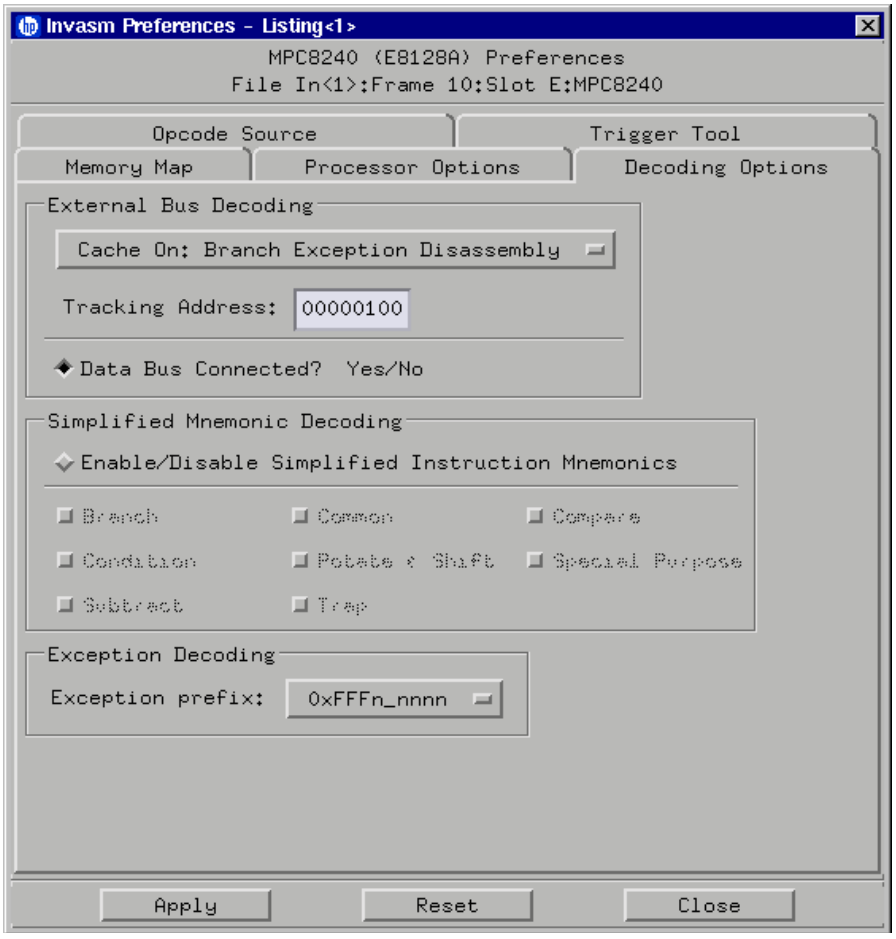
Endian Mode Selection

Big Endian Mode. Big endian mode is the native mode of the processor.

Little 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.

To set the Decoding Options preferences



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.

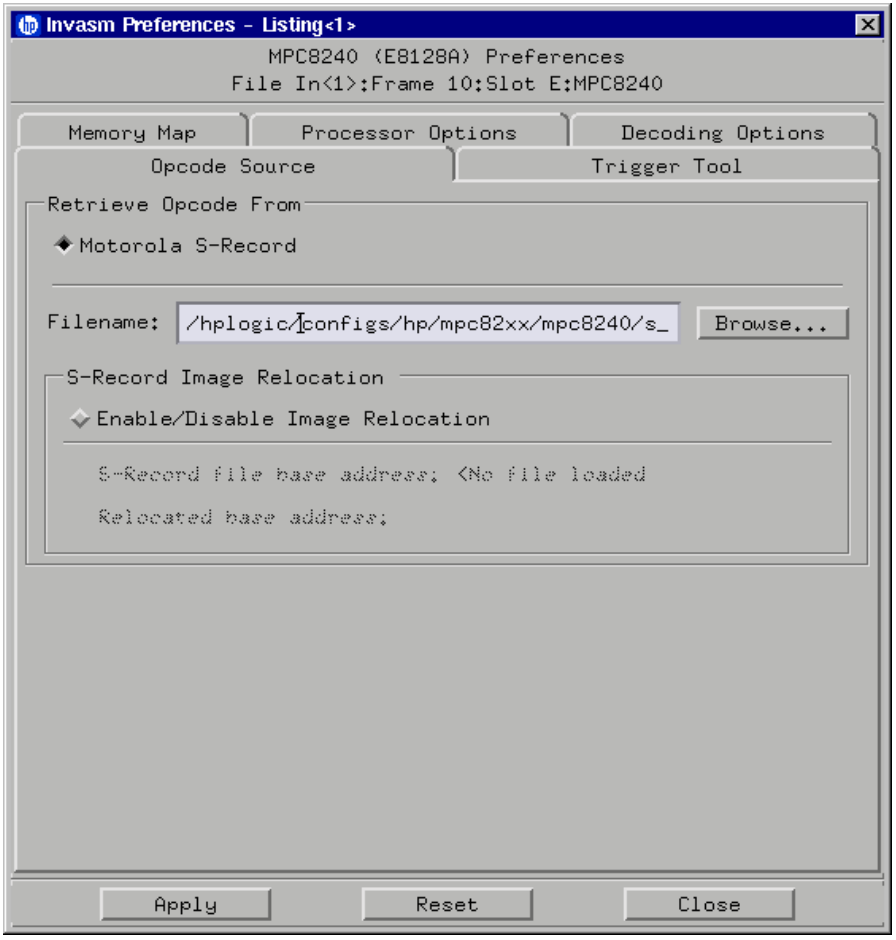
Data Bus Connected. Read and write states are always indicated regardless of whether the data bus is connected. However, when the data bus is connected, read/write data will also be displayed. See “Inverse Assembler Modes of Operation” on page 110

Simplified Mnemonic Decoding. PowerPC assemblers support a number of simplified mnemonics for some popular assembly language instructions, as described in the *PowerPC Programming Environments Guide* (Appendix E). 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.

Displaying the simplified mnemonics may help you to get a better idea of what a particular instruction is really doing. For example, an "or r1,r1,r1" instruction is simplified to a "nop."

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 IP bit (bit 25) of the MSR register. This can be done by examining the initialization code or by using an emulator to view the MSR register.

To set the Opcode Source preferences



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

S-Record Image Relocation. The Image Relocation portion of the dialog box allows you to relocate the SREC file to some other location in memory. This is useful when the loaded file is moved to some other location in memory.

For example, the starting address in the SREC file is 1000. However, memory starting at 1000 is relocated to 5000. In order for the inverse assembler to retrieve the correct data, the entire SREC file must be relocated to 5000. Enter the relocated base address; all the resulting offsets will be calculated by the inverse assembler.

To enable/disable the instruction cache on the MPC8240

When the instruction cache is enabled, many PowerPC instructions are executed from the cache and do not appear on the external bus. Cache-on trace reconstruction is used to trace execution in cache.

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
```

Loading Symbol Information

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

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

Also, you can download symbols from certain object file formats into HP logic analyzers.

To view predefined symbols for the MPC8240

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

User-defined symbols are saved with logic analyzer configurations. The logic analyzer configuration files included with the MPC8240 inverse assembler contain predefined symbols for logic analyzer labels.

To display the predefined symbols for the MPC8240:

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

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

Predefined Symbols

Label	Value (hex)	Symbol
MAASRC*	8	mem read
	9	touch load
	A	inst fetch
	B	reserved read
	C	PCI mem read
	D	DMA0 mem read
	E	DMA1 mem read
	F	I2O mem read
	0	mem write
	1	reserved write
	2	reserved write
	3	reserved write
	4	PCI mem write
	5	DMA0 mem write
	6	DMA1 mem write
	7	I2O mem write
RAS_	7F	CS0
	BF	CS1
	DF	CS2
	EF	CS3
	F7	CS4
	FB	CS5
	FD	CS6
	FE	CS7
	FF	0XFF
RCS0_	0	CS0
	1	1
RCS1_	0	CS1
	1	1
WE_	1	read
	0	write

* MAASRC is composed of the MAA and $\overline{WE}$ signals.

To load object file symbols

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

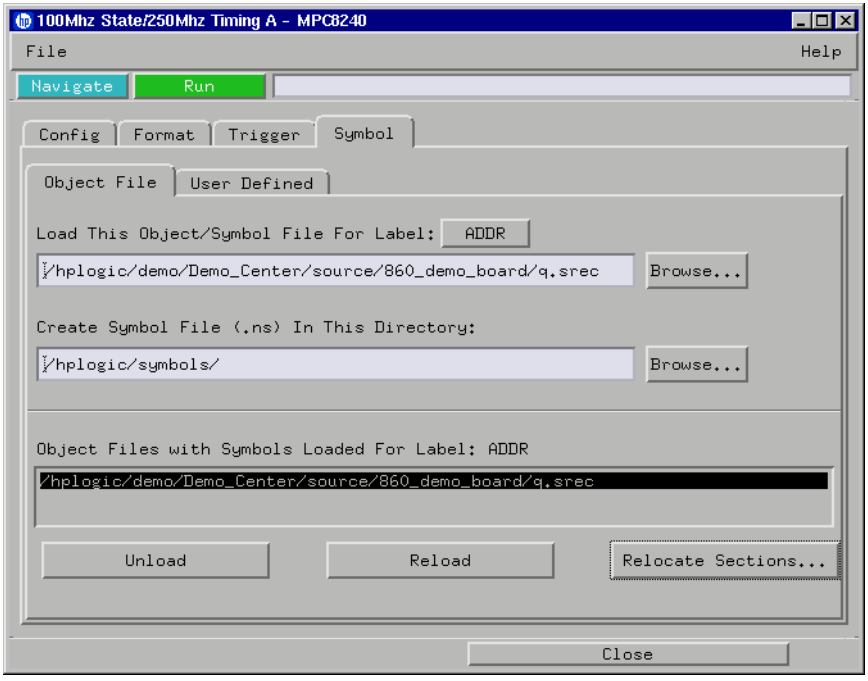
If your compiler generates object files in a format that the logic analyzer doesn't understand, you can use a General-Purpose ASCII (GPA) symbol file (see Chapter , "General-Purpose ASCII (GPA) Symbol File Format," on page 271).

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

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

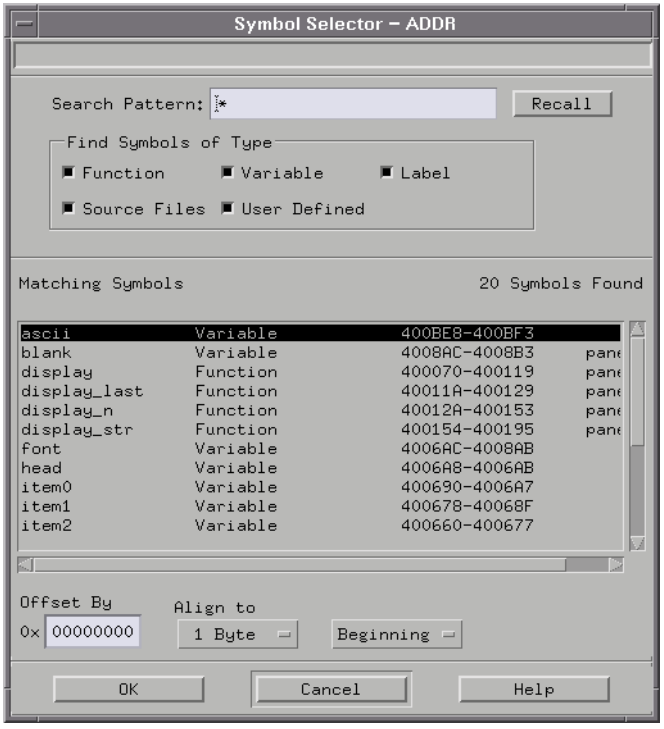
   Make sure the label is ADDR.

   From this dialog you can select object files and load their symbol information.



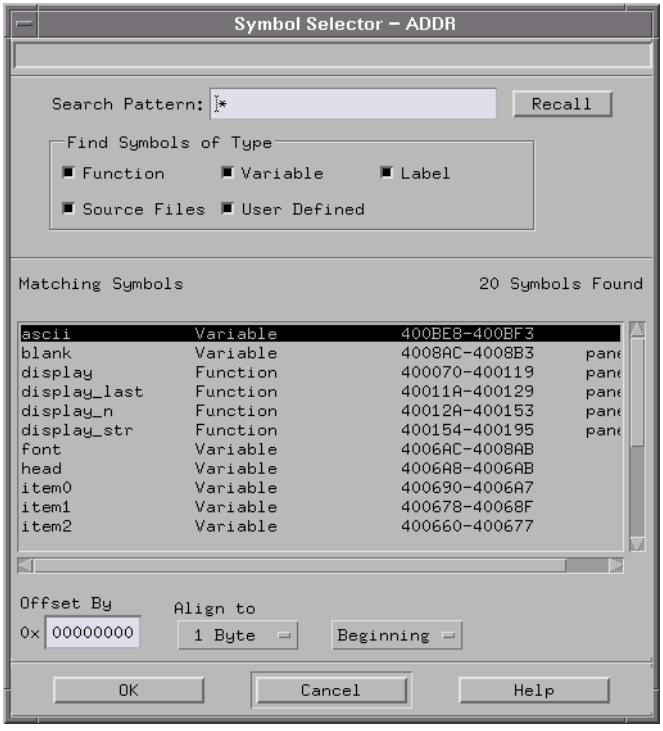
When you load object file symbols into a logic analyzer, a database of symbol/line number to address assignments is generated from the object file.

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



To compensate for relocated code

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



Entering the appropriate address offset will cause the logic analyzer to reference the correct symbol information for the relocatable or offset code.

Setting Up Labels for Groups of Signals

Predefined Label Descriptions

The logic analyzer configuration files automatically set up labels for most MPC8240 signals. The following tables show some of the predefined labels for the most commonly used signals..

ADDR Label

ADDR	Kahlua Signal Name
Bit 0	SDMA[0]
Bit 1	SDMA[1]
Bit 2	SDMA[2]
Bit 3	SDMA[3]
Bit 4	SDMA[4]
Bit 5	SDMA[5]
Bit 6	SDMA[6]
Bit 7	SDMA[7]
Bit 8	SDMA[8]
Bit 9	SDMA[9]
Bit 10	SDMA[10]
Bit 11	SDMA[11]
Bit 12	SDMA[12] / SDBA1
Bit 13	SDBA0
Bit 14	debug_address[0]
Bit 15	debug_address[1]
Bit 16	debug_address[2]
Bit 17	debug_address[3]
Bit 18	debug_address[4]
Bit 19	debug_address[5]
Bit 20	debug_address[6]
Bit 21	debug_address[7]
Bit 22	debug_address[8]
Bit 23	debug_address[9]
Bit 24	debug_address[10]
Bit 25	debug_address[11]
Bit 26	debug_address[12]
Bit 27	debug_address[13]
Bit 28	debug_address[14]
Bit 29	debug_address[15]

DATA Label

DATA	Kahlua Signal Name
Bit 0	DH[31]
...	...
Bit 31	DH[0] (MSB)

DATA_B Label

DATA_B	Kahlua Signal Name
Bit 0	DL[31]
...	...
Bit 31	DL[0] (LSB)

STAT Label

STAT	Kahlua Signal Name
Bit 0	MAA[2]
Bit 1	MAA[1]
Bit 2	MAA[0]
Bit 3	$\overline{\text{RCS1}}$
Bit 4	$\overline{\text{RCS0}}$
Bit 5	RAS / $\overline{\text{CS}}$[7]
Bit 6	RAS / $\overline{\text{CS}}$[6]
Bit 7	RAS / $\overline{\text{CS}}$[5]
Bit 8	RAS / $\overline{\text{CS}}$[4]
Bit 9	RAS / $\overline{\text{CS}}$[3]
Bit 10	RAS / $\overline{\text{CS}}$[2]
Bit 11	RAS / $\overline{\text{CS}}$[1]
Bit 12	RAS / $\overline{\text{CS}}$[0]
Bit 13	$\overline{\text{AS}}$
Bit 14	$\overline{\text{WE}}$
Bit 15	$\overline{\text{FOE}}$
Bit 16	CAS / $\overline{\text{DQM}}$[7]
Bit 17	CAS / $\overline{\text{DQM}}$[6]
Bit 18	CAS / $\overline{\text{DQM}}$[5]
Bit 19	CAS / $\overline{\text{DQM}}$[4]
Bit 20	CAS / $\overline{\text{DQM}}$[3]
Bit 21	CAS / $\overline{\text{DQM}}$[2]
Bit 22	CAS / $\overline{\text{DQM}}$[1]
Bit 23	CAS / $\overline{\text{DQM}}$[0]
Bit 24	SDRAM_CLK[0]*

STAT	Kahlua Signal Name
Bit 25	$\overline{\text{MIV}}$
Bit 26	$\overline{\text{SDRAS}}$
Bit 27	$\overline{\text{SDCAS}}$

Clock & Qualifier

CLK	Kahlua Signal Name
J	SDRAM_CLK[0] (SYSCLK)
K	$\overline{\text{MIV}}$

To define additional labels

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

Changing the Analysis Mode

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

- State.
- Timing.

Inverse assembly is available in the state analysis mode.

To change to state analysis

In state mode, the logic analyzer uses the CLKIN signal from the MPC8240 target system to capture data synchronously. This mode allows inverse assembly of MPC8240 instructions and is the default mode set up by the configuration files.

To configure the logic analyzer for state mode:

- Load the appropriate logic analyzer configuration file (see "Loading Configuration Files" on page 104).

  The state configuration files set up the rising edge of the J clock (J↑) as the master clock signal.

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

To change to timing analysis

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

Inverse assembly is not available in the timing analysis mode.

To configure the logic analyzer for timing analysis:

1. Load the appropriate logic analyzer configuration file (see "Loading Configuration Files" on page 104).
2. Open the logic analyzer's Setup window.
3. Select the Config tab.
4. Change the type option from State to Timing.

6

Capturing MPC8240 Execution

The normal steps in using the logic analyzer are:

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

With the MPC8240 inverse assembler, the logic analyzer is configured, and labels are created (formatted) for the logic analysis channels when configuration files are loaded (see "Loading Configuration Files" on page 104).

You can load program object file symbols into the logic analyzer when configuring it (see "Loading Symbol Information" on page 126).

This chapter describes setting up logic analyzer triggers when using the inverse assembler and the HP B4620B Source Correlation Tool Set.

See Chapter 7, "Displaying Captured MPC8240 Execution," beginning on page 147 for information on displaying captured data.

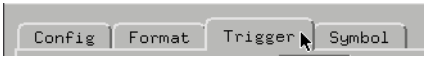
Note: The screens you see may be different from what you see in this manual, depending on the version of your logic analyzer system software.

Setting Up Logic Analyzer Triggers

To set up logic analyzer triggers

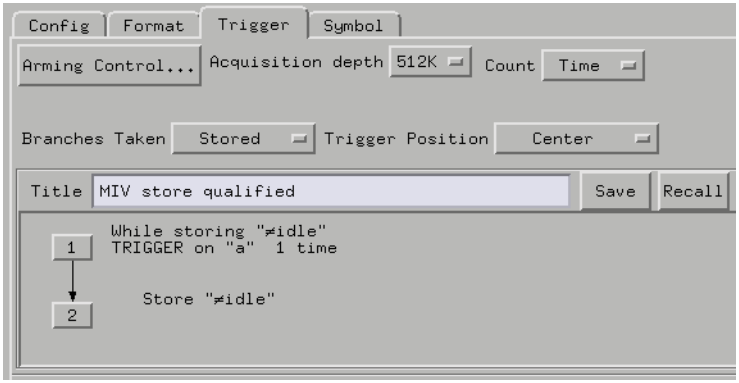
1 Open the logic analyzer's Setup window.

2 Select the Trigger tab.



3 Define the patterns, ranges, and other resources that will be used in the logic analysis measurement.

4 Set up the trigger sequence.



5 Run the measurement.

See Also

The HP 16600A/16700A-series logic analysis system's on-line help for more information on setting up logic analyzer triggers.

To trigger on an access to a memory address

The address bus is comprised of a group of signals.

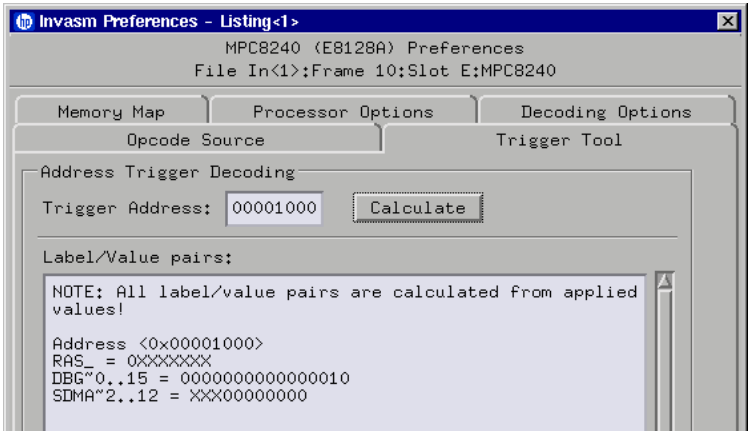
1 In the Listing window, open the Invasm Preferences dialog.

2 Click the **Memory Map** tab and check that you have set up the memory banks correctly. Click **Apply**.

For more information on the memory map, see "To set the Memory Map preferences" on page 113.

3 Click the **Trigger Tool** tab.

4 Enter the trigger address in the Trigger Address field, then click **Calculate**.

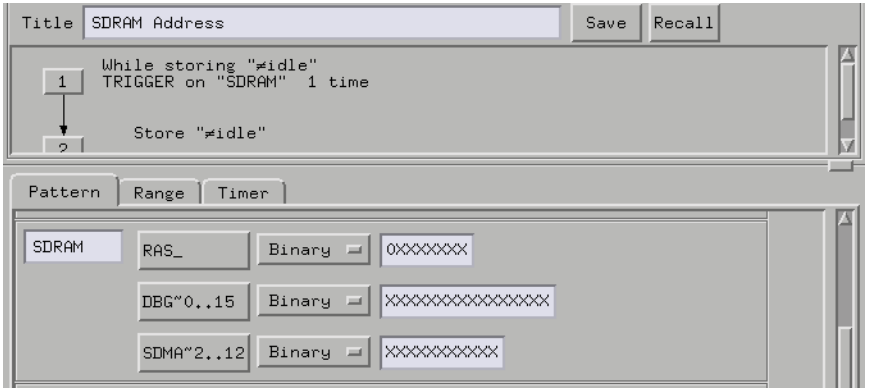


For DRAM or SDRAM addresses:

If the Trigger Tool shows a RAS_ value, the address is in DRAM or SDRAM.

5 In the logic analyzer Setup window, click the **Trigger** tab, then click **Recall** and select the **SDRAM Address** trigger.

6 Click the **Pattern** tab and scroll down to the “SDRAM” pattern.



7 Enter the values from the Trigger Tool.

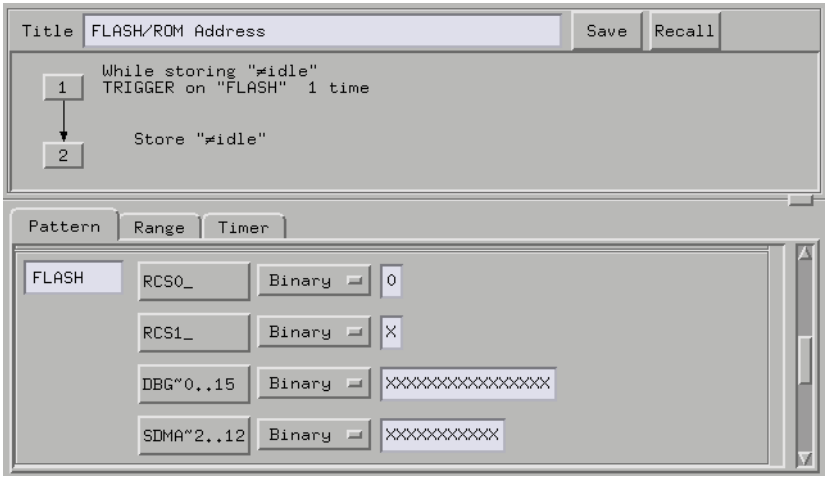
If you are working at a display which is connected directly to the logic analysis system, you can cut text by holding down the left mouse button dragging the cursor over the text. You can paste text by clicking the middle button.

For FLASH/ROM addresses:

If the Trigger Tool shows a RCS0_ or RCS1_ value, the address is in FLASH/ROM.

5 In the logic analyzer Setup window, click the **Trigger** tab, then click **Recall** and select the **FLASH/ROM Address** trigger.

6 Click the **Pattern** tab and scroll down to the “FLASH” pattern.



7 Enter the values from the Trigger Tool.

If the Trigger Tool does not show a value for one of the chip selects, enter ‘X’ (“don’t care”). For example, if the Trigger tool shows “RCS0_ = 0”, enter ‘0’ in the RCS0_ field, and enter ‘X’ for RCS1_.

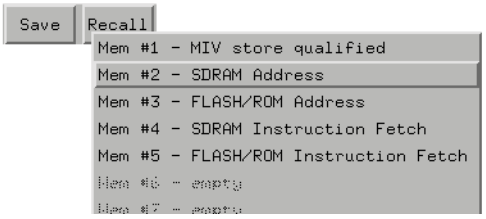
If you are working at a display which is connected directly to the logic analysis system, you can cut text by holding down the left mouse button dragging the cursor over the text. You can paste text by clicking the middle button.

For both kinds of addresses:

8 **Run** the logic analyzer measurement.

Using the Saved Trigger Specifications

Logic analyzer configuration files for the MPC8240 contain saved trigger specifications. You can recall these trigger specifications from the Recall button of the Trigger tab.



MIV store qualified

This is the default trigger specification. Any captured state will trigger the logic analyzer and any non-idle ($\overline{\text{MIV}}$=0) state after that is stored. It gives you an easy way to return to the default.

SDRAM Address

Any access to the address defined by the “SDRAM” pattern will trigger the logic analyzer and any non-idle ($\overline{\text{MIV}}$=0) state after that is stored.

The inverse assembler Preferences dialog has a Trigger Tool tab that will show you the values to use for a particular SDRAM address (see “To trigger on an access to a memory address” on page 141).

FLASH/ROM Address

Any access to the address defined by the “FLASH” pattern will trigger the logic analyzer and any non-idle state after that is stored.

The inverse assembler Preferences dialog has a Trigger Tool tab that will show you the values to use for a particular address.

SDRAM Instruction Fetch

Any instruction fetch from address defined by the “SDRAM” pattern will trigger the logic analyzer and any non-idle state after that is stored.

The inverse assembler Preferences dialog has a Trigger Tool tab that will show you the values to use for a particular address.

FLASH/ROM Instruction Fetch

Any instruction fetch from address defined by the “FLASH” pattern will trigger the logic analyzer and any non-idle state after that is stored.

The inverse assembler Preferences dialog has a Trigger Tool tab that will show you the values to use for a particular address.

Triggering on Source Code

When setting up trigger specifications to capture MPC8240 execution:

- 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).

To avoid capturing library code execution

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

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

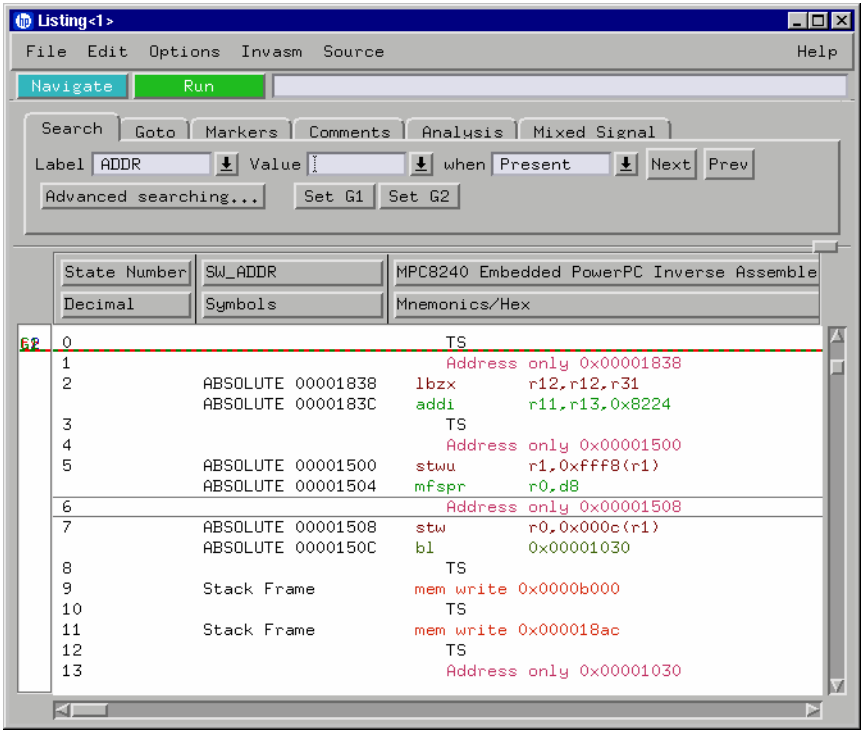
7

Displaying Captured MPC8240 Execution

To display the captured state data

1 Open the Listing display window.

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



The inverse assembler is already loaded when state configuration files are loaded, but it can also be loaded into a Listing display using the Invasm menu. The name of the inverse assembler file is I8620E, and it is located in the /

hplogic/ia directory.

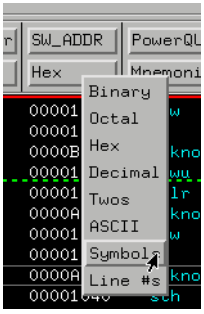
See Also

"To use the inverse assembler filters" on page 150 for information on displaying or hiding certain types of microprocessor bus cycles.

The HP 16600A/16700A-series logic analysis system on-line help for information on using the Listing display.

To display symbols

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



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

See Also

"To load object file symbols" on page 128.

To interpret the inverse assembled data

Registers

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.

Numbers.

Most numerical data is displayed in hexadecimal, for example, "lwz r28, 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."

Instructions.

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 MPC8240 are decoded as "illegal". Any instruction that does not decode to a valid opcode is shown as "unknown".

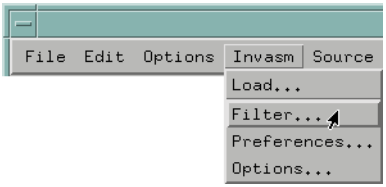
A “*” is used to mark unused prefetches. A “?” is used to mark *possibly* unused prefetches.

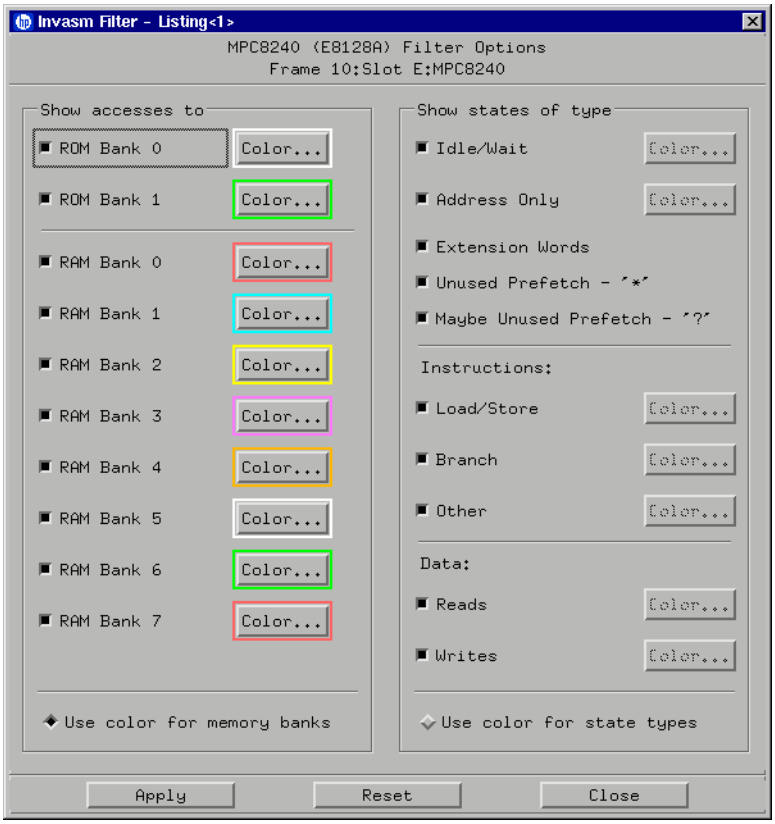
An instruction word of 00000000 is decoded as "illegal." Otherwise, if an opcode is invalid, it is shown as "Undefined Opcode."

When an exception occurs, the exception type is displayed in parentheses after the instruction.

To use the inverse assembler filters

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





The inverse assembler filtering options allow you to display or hide certain types of microprocessor bus cycles or memory bank accesses.

Because the filter options do not affect the data that is stored by the logic analyzer (they only affect whether that data is displayed), they let you display the same data in different ways.

Filtering allows faster analysis in two ways:

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

You can also use color to distinguish between cycle types or memory bank accesses (when they are displayed). Color can be used for distinguishing between memory bank accesses or cycle types, but not both at the same time.

You can display or hide the following types of cycles:

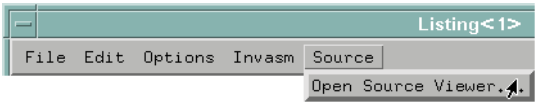
- Idle/Wait States.
- Address Only States.
- Unused or possibly unused prefetches.
- Extension Words.
- Branch Instructions.
- Load/Store Instructions.
- Other Instructions.
- Data Reads.
- Data Writes.

See Also

The address ranges for memory banks 0-11 are specified in the Preferences menu (see “To set the inverse assembler preferences” on page 112).

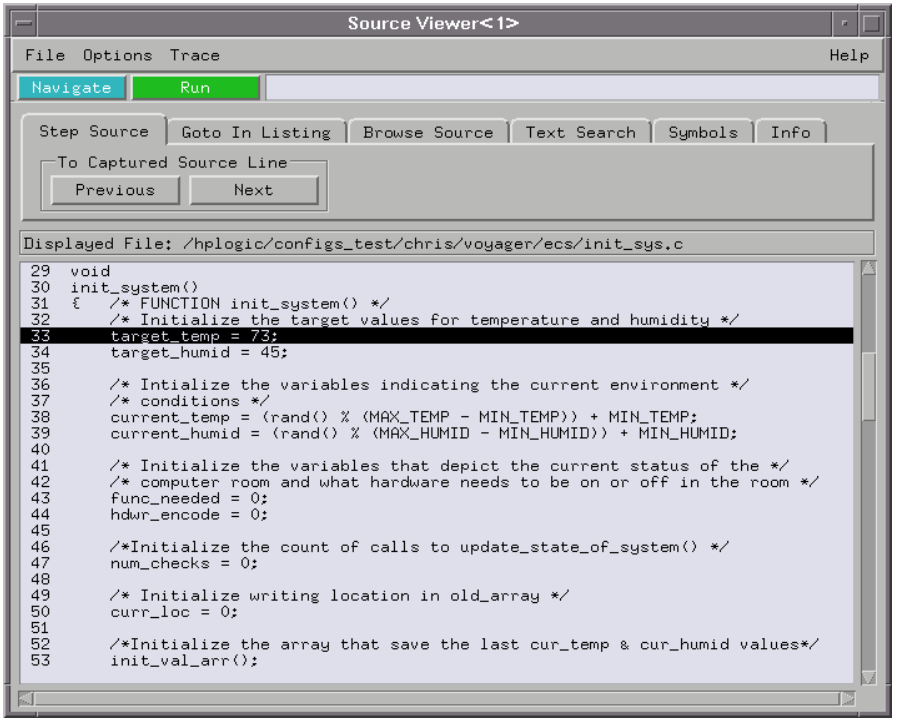
To view the source code associated with captured data

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



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

The source correlation tool set correlates the logic analyzer's address label with a line of high-level source code whose address, symbol name, file name, and line numbers are described in a symbol file downloaded to the logic analyzer (see "To load object file symbols" on page 128).



Inverse Assembler Generated SW_ADDR (Software Address) Label

In the HP 16600A/16700A-series logic analysis system, the MPC8240 inverse assembler generates a “SW_ADDR” label. The SW_ADDR label is displayed as another column in the Listing tool. This label is also known as the software address generated by the inverse assembler.

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

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

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

begin after the first assembly instruction of the loop has been executed. A "Goto this line in listing" command would not find the source line.

Access to Source Code Files

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

Source File Search Path. Verify that the correct file search paths for the source code have been entered into the source correlation tool set. The 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.

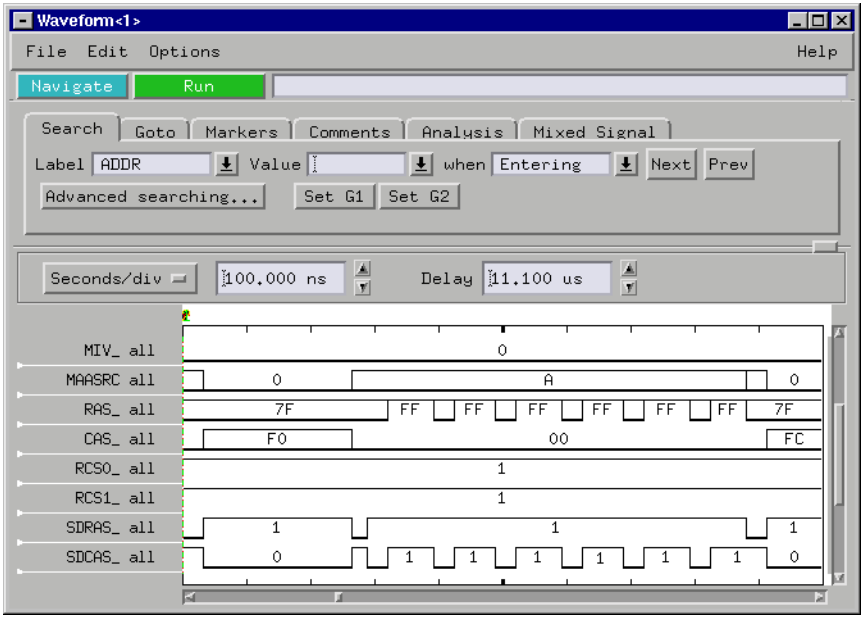
See Also

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

To display captured timing analysis mode data

- Open the Waveform display for your logic analyzer.

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



8

Troubleshooting the Logic Analyzer

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

If you still have difficulty using the analyzer after trying the suggestions in this chapter, please contact your local 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, and probes. Otherwise, you may damage circuitry in the logic analyzer or target system.

Solving Logic Analyzer Problems

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

Intermittent data errors

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

- ❏ Remove and reseat all cables and probes, ensuring that there are no bent pins or poor 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 162 for information on other sources of intermittent data errors.

Unwanted triggers

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

- ❏ Set the trigger at an address which is not the target of a conditional branch to avoid this problem. For example, set the trigger in the middle of a subroutine, rather than at the first instruction of a subroutine.

  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.

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.

Solving Probing Problems

This section lists probing problems that you might encounter when using a logic analyzer. If the solutions suggested here do not correct the problem, you may have a damaged logic analyzer. Contact your local Hewlett-Packard Sales Office if you need further assistance.

Target system will not boot up

If the target system will not boot up after connecting the target system, the microprocessor (if socketed) or the probe 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 logic analyzer and target system.

1. Power up the logic analyzer.
2. Power up the target system.

If you power up the target system before you power up the logic analyzer, interface circuitry may latch up and prevent proper target system operation.

❏ Verify that the logic analyzer cables are in the proper target system connector headers and are firmly inserted.

Erratic trace measurements

There are several general problems that can cause erratic variations in trace lists and inverse assembly failures.

- ❏ 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 connected.

  See "Capacitive loading" on page 162. While logic analyzer probe loading is slight, pin protectors, extenders, and adapters may increase it to unacceptable levels. If the target system design has close timing margins, such loading may cause incorrect processor functioning and give erratic trace results.

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

  Ensure that you have ambient temperature conditions and 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 logic analyzer, or system lockup in the microprocessor. All logic analyzer probes 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.

Solving 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 logic analyzer or in your target system. If you follow the suggestions in this section to ensure that you are using the logic analyzer and inverse assembler correctly, you can proceed with confidence in debugging your target system.

No inverse assembly or incorrect inverse assembly

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

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

  There is not always a one-to-one correspondence between analyzer pod numbers and probe cable numbers. Probes must supply address (ADDR), data (DATA), and status (STAT) information to the analyzer in a predefined order. The cable connections for each probe are often altered to support that need. Thus, one probe might require that you connect cable 2 to analyzer pod 2, while another will require you to connect cable 5 to analyzer pod 2. See "Connecting the Logic Analyzer to the Target System" on page 75 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 inverse assemblers also require other data labels. See "Configuring the Logic Analyzer" on page 103 for more information.

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

In most cases, if the microprocessor caches and memory managers remain enabled you should still get inverse assembly; however, it will consist of cache line fills.

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

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 "Configuring the Logic Analyzer" on page 103 for details.

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

Logic Analyzer Messages

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

". . . Inverse Assembler Not Found"

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

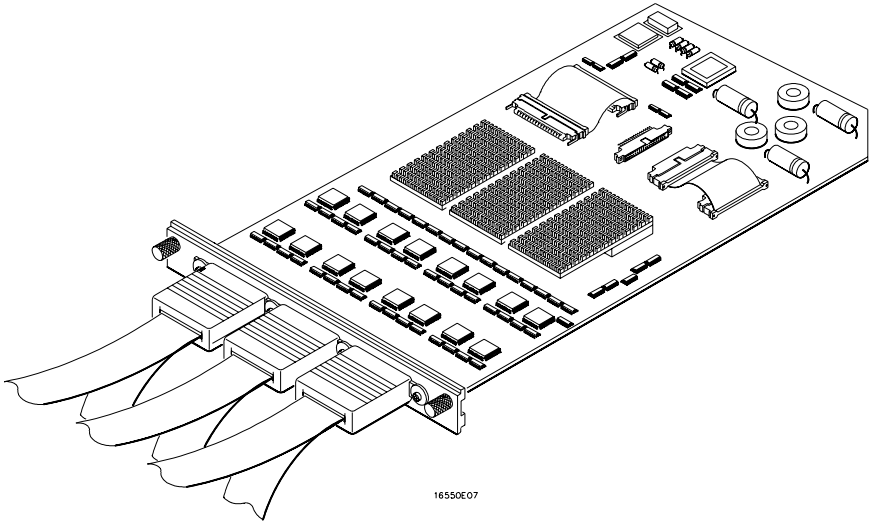
Ensure that the inverse assembler file is not renamed or deleted, and that it is located in the correct directory:

- For HP 16600A/16700A-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.

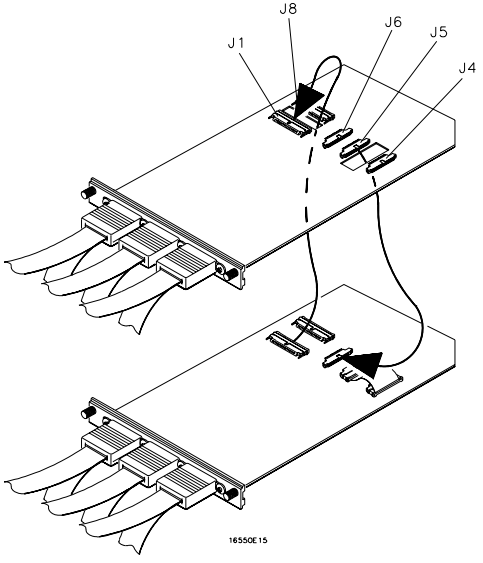
"Measurement Initialization Error"

This error occurs when you have installed the cables incorrectly for one or two HP 16550A logic analyzer cards. The following diagrams show the correct cable connections for one-card and two-card installations. Ensure that your cable connections match the silk-screened labels 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 configuration files.

See Also

See "Configuring the Logic Analyzer" on page 103 for a description of how to load 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 HP 16600A/16700A-series logic analysis system frame or in the HP 16701A expansion frame. Ensure that the cards are firmly seated.
- ❏ This error might occur if the target system is not running properly. Ensure that the target system is on and operating properly.
- ❏ If the error message persists, check that the logic analyzer pods are connected to the proper connectors on the target system. See "Connecting the Logic Analyzer to the Target System" on page 75

to determine the proper connections.

"Time from Arm Greater Than 41.93 ms"

The 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, if the trigger condition is set to look for an opcode fetch at an address not corresponding to a word boundary, the trigger will never be found.

Part 3

Using the Emulation Module

9

Using the Emulation Control Interface

The Emulation Control Interface in your HP 16600A/16700A-series logic analysis system allows you to control an emulation module.

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 or Intel Hex format) to the target system RAM or ROM.
- View emulator status and errors.
- Write and play back emulator command script files.

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.

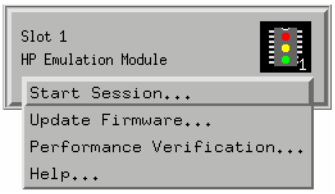
See Also

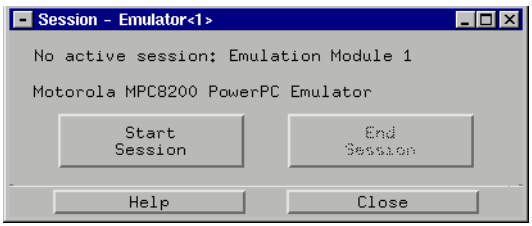
All of the Emulation Control Interface windows provide on-line help with a **Help** button or a **Help->On this window** menu selection. Refer to the on-line help for complete details about how to use a particular window.

To start from the main System window

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

2 Select **Start Session...**.



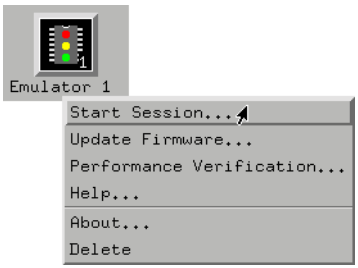


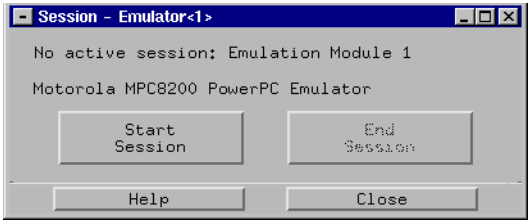
To start from the Workspace window

1 Open the Workspace window.

2 Drag the Emulator icon onto the workspace.

3 Right-click on the Emulator icon, then select Start Session....




Session - Emulator<1>
No active session: Emulation Module 1
Motorola MPC8200 PowerPC Emulator
Start Session
End Session
Help
Close

10

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.

Entering Emulation Module Commands

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 on-line 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.

To use the Emulation Control Interface

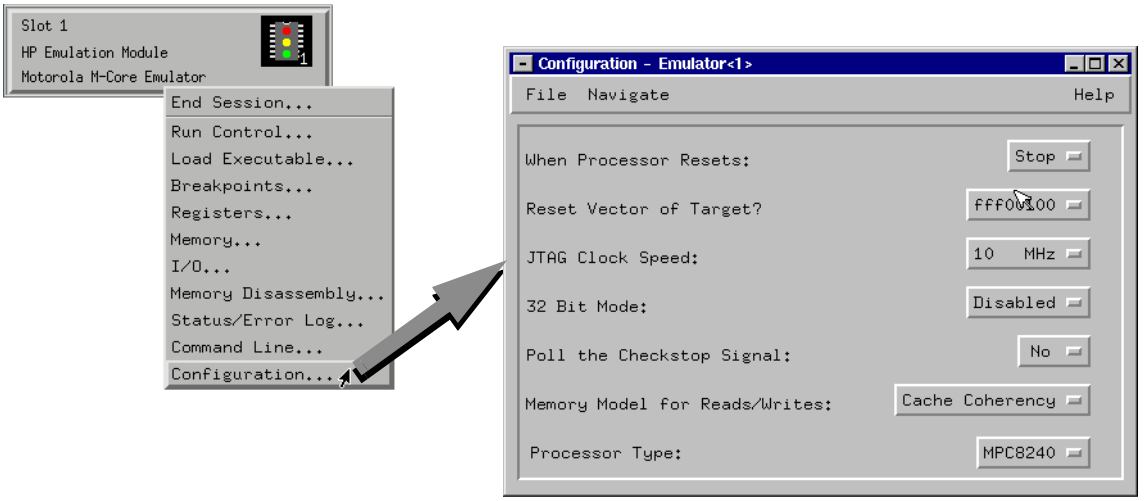
The easiest way to configure the emulation module is to use the Emulation Control Interface.

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

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 emulator with the Emulation Control Interface or a debugger interface, you can configure the emulator using the built-in "terminal interface" commands.

1 Connect a telnet session to the emulator 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 239.

Example

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

```
M>help cf

  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 ---
    proc     - Set type of processor
    reset    - Run or stop after reset
    vector   - Reset vector address
    rrt      - Set restriction to real time runs
    breakin  - BNC break in control
    trigout  - Trigger out control
    speed    - Set JTAG Clock Divisor

M>
```

To configure using a debugger

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

See Also

Information about specific debuggers in Chapter 11, “Using a Debugger with the Emulation Module,” on page 191.

Your debugger manual.

To configure the processor type

Processor type configuration		
Value	**Emulation module configured for**	**Built-in command**
MPC8260	MPC8260 PowerQUICC II (ver. 1.0)	cf proc = MPC8260
MPC8240	MPC8240 (ver. 1.0)	cf proc = MPC8240

The cfsave -s command will store the processor type configuration in the emulation module's flash memory. The cfsave -r command will restore this configuration.

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**
run	Issuing the rst command will hard reset the processor, reset the JTAG interface, and allow the processor to run. (Default)	cf reset=run
stop	Issuing the rst command will hard reset the processor, reset the JTAG interface, and cause the processor to stop at the reset exception vector 0xfff00100 or 0x00000100. The address used is determined by the config item 'vector'.	cf reset=stop

To configure reset vector address

The reset vector address configuration item specifies which reset vector the target hardware is using. This value does not set the reset vector. This is done

by the hardware.

Vector address configuration		
Value	**Emulation module configured for**	**Built-in command**
fff00100	Default	cf vector=fff00100
00000100		cf vector=00000100

To configure restriction to real-time runs

This option enables or disables restriction to real-time runs implemented for all commands other than "rst", "b", "s" and "r".

Real-time runs configuration		
Value	**Emulation module configured for**	**Built-in command**
no	If the processor is running user code, a request for a register or memory display will put the processor in monitor mode, read the requested register(s), then restore the processor to running user code. (Default)	cf rrt=no
yes	If the processor is running user code, a request for a memory or register display will return: !ERROR 647! Restricted to Real Time	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 that feature. For this reason, the Emulation Control Interface does not provide a way to set rrt=yes.

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.

The base JTAG clock frequency is 10MHz. This configuration item selects the division factor for the clock.

Processor clock speed configuration		
Value	**Processor clock (CLKIN) 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

To configure 32 bit mode

32 bit mode configuration		
Value	**Emulation module configured for**	**Built-in command**
Disabled	(default)	cf 32bitmode=off
Enabled	The maximum memory access size is forced to 32 bits.	cf 32bitmode=on

Do not enable this option if it is not supported by your target board.

To configure polling of the checkstop signal

Select whether to poll the checkstop signal (pin 15 on the JTAG connector)to detect a checkstop condition.

Checkstop polling configuration		
Value	**Emulation module configured for**	**Built-in command**
No	The checkstop signal will not be polled. Use this for target systems which are not using this signal as a checkstop. (default)	cf checkstop=off
Yes	Poll the checkstop signal to detect a checkstop condition.	cf checkstop=on

To configure the memory model for reads and writes

The emulator decodes the contents of the instruction cache before displaying it. Only valid instructions can be decoded. Invalid instructions will be displayed as-is, in coded form, and thus might not match the contents of memory.

All memory writes of 1 Kbyte or greater cause the icache to be invalidated and the dcache is flushed and invalidated for the current address range.

Memory model configuration		
Value	**Emulation module configured for**	**Built-in command**
Cache Coherency	Use cache coherency memory model for reads/writes. (default) All memory reads and writes less than 1K bytes will maintain cache coherency and the cache will not be invalidated or flushed. This model assumes instructions and data are in separate address spaces and will not occur within the same cache set and way.	cf memmodel=cache
Physical Memory	Use only physical memory for reads/writes regardless of the state of cache.	cf memmodel=phys

Break

If the following error occurs, the processor is in an unstable state and should be reset via a hard reset or a power-on reset. The 'rst' command may not be able to bring the processor into a known state.

```
!ERROR  145! Unable to soft stop - freezing the processor
clocks
```

Software Breakpoints

The number of software breakpoints is unlimited but they cannot be used with ROM or FLASH memory. Software breakpoints cannot be used to debug exceptions. Do not modify the IABR register while software breakpoints are enabled.

It is recommended that only -t (temporary) or -p (permanent) breakpoints be used by a debugger. If neither option is given, the breakpoint acts much like a temporary breakpoint except that when the breakpoint is hit, it is disabled but remains in the breakpoint list. A temporary breakpoint is removed from the breakpoint list when hit.

NOTE: For software breakpoints, the break conditions set are dependent on the value of the MSR(IP) bit value. When software breakpoints are enabled, the value of this bit determines the break condition. If user code or debuggers change this value after software breakpoints have been enabled, software breakpoints will not work. The work around for this is to set a hardware breakpoint to an instruction after the code that changes the MSR(IP), then after hitting the hardware breakpoint, disable hardware breakpoints and enable software breakpoints.

Hardware Breakpoints

One hardware breakpoint is available. Since using software breakpoints requires the use of the hardware breakpoint, both types of breakpoints cannot be used at the same time. Enable hardware breakponts using 'bc -e hwbp'. Set a hardware breakpoint using 'bp -h <-p | -t> <address>. Use either the -p or -t option when setting the hardware breakpoints.

Testing the Emulation Module 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

See Chapter 13, “Troubleshooting the Emulation Module,” on page 235 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 by running a 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.

11

Using a Debugger with the Emulation Module

Several companies sell source debuggers which work with the HP emulator.

Benefits of Using a Debugger

A debugger lets you:

- control (start and stop) the execution of your microprocessor
- step through your code at the source-code level
- set breakpoints
- single-step through source code
- examine variables
- modify source code variables
- download executable code to your target system

Compatibility with Other Logic Analysis System Tools

You can use your logic analysis system to trace and analyze target system execution while you use your debugger.

If the computer running the debugger is also running X Windows server software, you can display logic analyzer windows next to your debugger windows.

Minimum Requirements

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

- A debugger that is compatible with the emulation module.

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

- A LAN connection between the PC or workstation that is running the debugger and the emulation probe or the HP 16600A/16700A-series logic analysis system (which contains the emulation module).

  Emulation probes or emulation modules communicate with debuggers over the LAN.

- To have the logic analysis system user interface displayed on your PC or workstation screen along with the debugger, your computer needs to be running X Windows server software.

  Most UNIX workstations run X Windows server software, but on a PC you

may need to install X Windows server software.

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 section assume that your PC or workstation is already connected to the LAN and that you have already installed the debugger software according its documentation.

To use your debugger with the emulation module:

1. Install the emulation module (see "Installing the Emulation Module" on page 64).
2. Connect the emulation module to your target system (see "Connecting the Emulation Module to the Target System" on page 95).
3. If you are using the debugger with an emulation module in a logic analysis system, you must connect the logic analysis system to the LAN (see "To connect the logic analysis system to the LAN" on page 195).
4. If you want to display logic analysis system windows next to your debugger windows, export the logic analysis system's display to your PC or workstation (see "To view logic analysis system windows next to the debugger" on page 197).
5. Configure the emulation module (see "Configuring the Emulation Module" on page 177).

   If you use the logic analysis system's Emulation Control Interface to configure the emulation module, remember to end the Emulation Control Interface session before you start the debugger.
6. Begin using your 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

See the logic analysis system's installation guide or on-line help for information on setting up a logic analysis system on the LAN.

Debuggers require information about a logic analysis system's LAN connection so they can communicate with an emulation module. They need (write the information here for future reference):

- IP Address of Logic Analysis System _____
- Hostname of Logic Analysis System _____
- Gateway Address _____
- Port Number of Emulation Module _____

Default emulation module port numbers

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 in an HP 16600A/700A-series logic analysis system)
6476	Slot 2 (Second emulation module in an HP 16700A-series system)
6480	Slot 3 (Third emulation module in an expansion frame)
6484	Slot 4 (Fourth emulation module in an expansion frame)
*Port numbers for telnet connections are different than for debugger connections because telnet uses a different service than debuggers, and a telnet port is already set up in order to display the logic analysis system interface remotely.	

To change the port number of an emulation module

Some debuggers do not provide a way to specify an emulation module port

number. In this case:

- The debugger will always connect to port 6470 (the default port number of an emulation probe, or the port number of the emulation module in slot 1 of an HP 16600A/16700A-series logic analysis system).
- If the port number of the emulation module is not 6470, you must change it.

To view or change the port number of an emulation module:

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 LAN 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

For information on physically connecting the logic analysis system to the LAN and configuring its LAN parameters, see the installation guide or on-line help for your logic analysis system.

To view logic analysis system windows next to the debugger

1. Make sure the computer running the debugger is also running X Windows server software and has telnet software.
2. Give the logic analysis system permission to display on the X Windows server.
3. Connect to the logic analysis system, log in, and start a session, displaying on the X Windows server.

Example, UNIX

On a UNIX workstation:

1. 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, PC

On a Windows 95 PC with Reflection X server software from Walker Richer & Quinn, Inc.:

1. On the PC, start the X Windows server software and connect to the logic analysis system.

   To start Reflection X, click the Reflection X Client Startup icon. Enter the following values in the Reflection X Client Startup dialog:

   a. In the Host field, enter the hostname 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.

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

Using the Green Hills Debugger

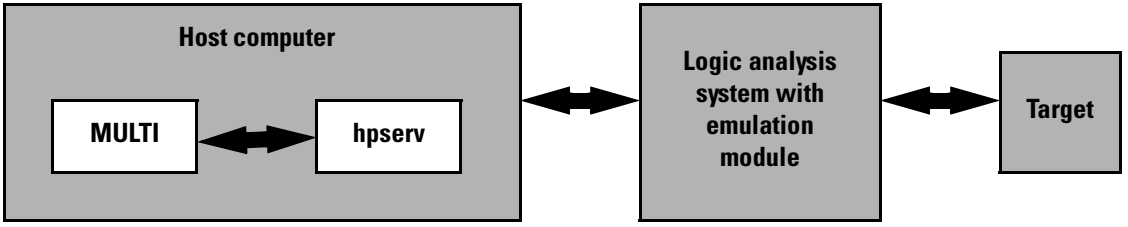
Compatibility

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

This information in this section is intended to be used along 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 an MPC8200 processor.

Go to the system window of the logic analyzer interface and verify that the Emulation Module icon is described as a "Motorola MPC82XX Emulator". If it is not, follow the instructions on page 99 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 ppc603.lnk file to user.lnk.

You may need to edit the user.lnk file to place the program at a location where target system memory is available.

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 "To connect the logic analysis system to the LAN" on page 195 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 JTAG 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 one of the following methods to explore the configuration options and configure the emulation module and target system:

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

See "To configure using an initialization script" on page 202 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:

NOTE: You can significantly increase download performance by disabling the caches. Disable caches by writing the appropriate bits to the HID0 register (see "Preparing for Emulation" on page 58).

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

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

To configure 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 provides some sample initialization sequence files in the "hpdemo" directory. Look at the files that have the ".rc" suffix.

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=runrom
```

```
_INIT_SP=0x10000
```

Save the file in the MULTI startup 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, right-click on the source code line where the breakpoint is to be located.
- To clear a breakpoint, right-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 with 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 terminal interface 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 4 through 7 in "To get started" on page 199 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"

This message usually means that there is no target interface module (TIM) connected to the emulation module or that the emulation module does not have firmware for the MPC8000 family.

1. Verify that the emulation module is connected to the target.
2. Next, check that your emulation module is programmed with firmware for the Motorola MPC8000:

   See "To display the emulation module firmware version information" on page 100.

"command socket connection failed: WSAECONNREFUSED: connection refused"

This message usually means the emulation module is not at port #6470 on the logic analysis system.

See Also

Green Hills MULTI Software Development Environment User's Guide.

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

The Green Hills web site: **http://www.ghs.com**

See "Configuring the Emulation Module" on page 177 for more information on configuration options and the "cf" command.

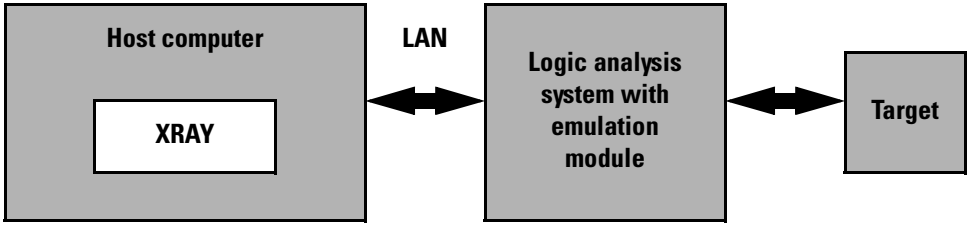
Using the Microtec Research Debugger

Compatibility

Version 4.1 of the XRAY HP Probe debugger from Microtec Research, a Mentor Graphics Company, is another debugger that connects to the HP emulation module.

This information in this section is intended to be used along with the XRAY documentation provided by Microtec Research.

Overview



To get started

1 Edit the gtw.brd file.

The file gtw.brd includes example hostnames, port numbers and initialization information for HP emulators that might be on the network for XRAY to connect to. The gtw.brd file is in the “etc” directory under the Microtec tools directory.

a Modify gtw.brd to identify the emulation module.

Modify the file to include the emulation module that you want XRAY to communicate with.

See “To connect the logic analysis system to the LAN” on page 195 for information on which port number to use for your emulation module.

b Add commands to initialize the target system.

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

XRAY provides a way for target initialization to occur through the gtw.brd file. The initialization sequences (contained in "{}" pairs) included in the gtw.brd commands specify the commands that will be sent to the HP emulation module to initialize it and prepare it for code download.

- The gtwinit command sequence defined in the gtw.brd file is sent to the HP emulation module when XRAY is establishing connection with the module.
- The gtwreset command sequence is sent to the emulation module when the XRAY "Reset" command is invoked.

The example gtw.brd file provided by Microtec Research contains initialization sequences which can be referenced. If the configuration for your target board is very involved, you can use the "gtwinit" definition in gtw.brd to merely reset the processor and break and use an include file to do the many configuration steps. Please refer to "To configure the emulation module using an INCLUDE file" on page 208 for more information on using an include file.

If you are unsure of the configuration needed for your emulation module, you can telnet to the emulation module or use the Configuration window in the logic analysis system's Emulation Control Interface to explore the configuration options. If you use this interface to actually configure your emulation module while connected to XRAY, configuration will be complete and you can proceed to the next step.

NOTE: You must start up XRAY from scratch after gtw.brd is modified for the changes you have made in gtw.brd to be recognized by XRAY.

2 Start XRAY.

After modifying gtw.brd, bring up the XRAY debugger. When XRAY comes up, the Managers dialog will be highlighted. (If the dialog is not present, the Managers dialog can be brought up from the Output Logging Window by

selecting **Managers→Connection Manager**).

Using the Managers dialog, set up the connection to your HP emulation module by selecting the Connect tab, clicking on your emulation module name in the lower Available Connections table and click on the **connect** button. You should see your emulation module name appear in the Active Connections table in the top half of the dialog. At this point, you are connected to the emulation module and the initialization commands specified in the gtw.brd file have been sent to your emulation module. If you look in the Output Logging Window, you can verify that the connection and initialization did in fact take place.

3 Download the application code.

NOTE: You can significantly increase download performance by disabling the caches. Disable caches by writing the appropriate bits to the HID0 register (see "Preparing for Emulation" on page 58).

In the Managers dialog, select the Debug tab, then select **Execution→Load File to Target** or **Control→Load File to Target**. This will open the "Load File To Target" dialog. (Alternatively, you may select the Files tab and select **Load→Load File to Target**.)

Use the Load File To Target dialog to choose the file you would like to download. When the file you want is listed in the center window, you may double click on it to start the load.

When the load is complete, you will see the file you loaded appear in the Active Files window of the File tab and in the Active Processes window of the Debug tab. You are now ready to debug your application code.

To configure the emulation module using an INCLUDE file

You can use an include file to configure the emulation module and set up your target system after bringing up the XRAY debugger. If a complex configuration is needed for your emulation module and target (such as multi-commands sent to the emulation module) this will save time and reduce errors.

1 Save the configuration commands in a text file, one command per line. Microtec Research provides an example include file in its tools

directory under the xhippchp directory in the file “mo8xxads.inc” (for the MPC8xx family).

2 To run the include file, select **Include Commands from File** under the Debug menu in the Code window and double click on the include filename you want to execute.

To perform common debugger tasks

- To display registers, select **Register** under the Windows menu in the Code window.
- To set a breakpoint, double-click on the source code line where the breakpoint is to be located.
- To clear a breakpoint, double-click on the line where the breakpoint is set.
- To step through code, select one of the step icons at the top of the Code window.
- To run from current PC, click on the first icon in the Code window.
- To toggle the display between source code and source code interlaced with assembly code, click on the **Dsm** button at the bottom of the code display window.
- To load program symbols, reset the PC, reset the stack pointer, and run from start, click **restart**.

To send commands to the emulation module

"Terminal interface" commands may be sent directly to the emulation module from XRAY. There are two ways to do this:

- Using an include file (as explained in the "Using an INCLUDE file to configure the emulation module and target" section)

  OR

- Using the XRAY "cf" command.

  This command takes a string as a parameter and sends it to the emulation module. For example, if you want to send the emulation module command cf rst=soft, you can type

  ```
  cf "rst=soft"
  ```

  in the XRAY Debugger command line.

  Note that the command must be surrounded by double quotes.

To view commands sent by XRAY

XRAY communicates with the emulation module using the emulation module's "terminal interface" commands. XRAY automatically generates and sends the commands required for normal operation. The communication between XRAY and the emulation module can be logged to a file after a connection has been established between XRAY and the emulation module and viewed later. To enable logging, enter the command:

```
PROBEMESSAGE ON,msgfile
```

This will create the "msgfile" and log a summary of the messages that occur between XRAY and the emulation module to it. The logging can be turned off with the following command:

```
PROBEMESSAGE OFF
```

To disconnect from the emulation module and target

In the Managers window, select the Connect tab. Click on the emulation module name that you want to disconnect. Under the Control menu, select "Disconnect from Board" (or you can "Reconnect to Board" if you have lost connection to the emulation module).

Error conditions

"!ERROR 800! Invalid command: bcast"

This message usually means that there is no target interface module (TIM) connected to the emulation module or the emulation module does not have firmware for the MPC8000 family.

1. Verify that the emulation module is connected to the target.
2. Next, check that your emulation module is programmed with firmware for the Motorola MPC8000:

   See "To display the emulation module firmware version information" on page 100. If the emulation module is not programmed with the proper firmware, see "To update emulation module firmware" on page 99.

"command socket connection failed: WSAECONNREFUSED: connection refused"

This message usually means the emulation module is not at port #6470 on the logic analysis system.

See Also

The Microtec Research web site: **http://www.mentorg.com/microtec**

The *XRAY Debugger Reference Manual* by Microtec Research.

See "Configuring the Emulation Module" on page 177 for more information on configuration options and the "cf" command.

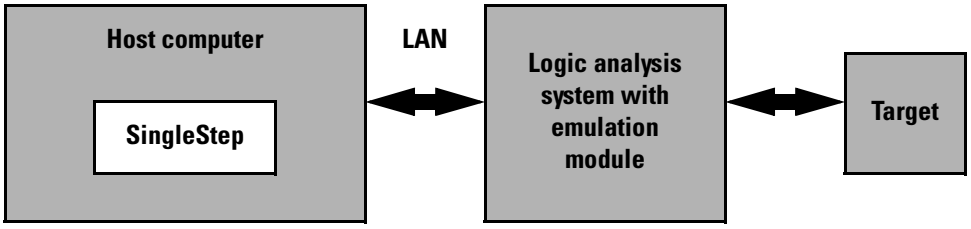
Using the Software Development Systems Debugger

Compatibility

Version 7.3 of the SingleStep debugger from Software Development Systems, Inc. is another debugger that connects to the HP emulation module.

The information in this section is intended to be used along 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 reset target option is selected, the target is reset.
- Hardware breakpoints are disabled.
- Software breakpoints are enabled.
- All breakpoints are cleared.
- main() _exit breakpoints are set, if that option is selected.

To get started

1. 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. See "To change the port number of an emulation module" on page 195 for more information on port numbers.

2. 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 Target Configuration tab in the "Debug" dialog box.

   An alternate way of performing target initialization is by using 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).

   The "Debug" dialog method and the sstep.ini method are mutually exclusive. Use one or the other, but not both.

   Initialization of the target will not actually occur until the "Debug" dialog is successfully exited.

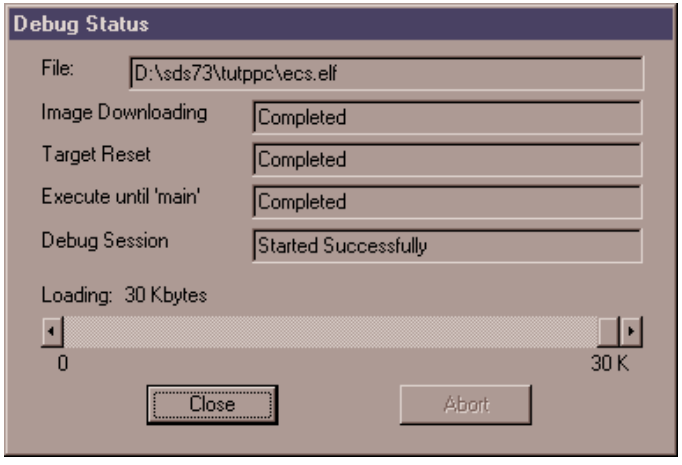
3. Set up the Loading and Execution options in the Options tab of the Debug dialog.
4. Download the application and run:

NOTE: You can significantly increase download performance by disabling the caches. Disable caches by writing the appropriate bits to the HID0 register (see "Preparing for Emulation" on page 58).

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 "load image" was selected, download the application and set the PC based on object module file contents.
- If "reset target" was selected, 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 "execute until main" was selected, 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: Ctrl-c.

Examples

To 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 "Emulator Built-In Commands" on page 239.

Cache disabling

(See "Preparing for Emulation" on page 58.) The default "sstep.ini" file sent with the PPC603 Single Step sets the HID0 register to 0, indicating that caches will be disabled.

Error conditions

"!ERROR 800! Invalid command: bcast"

This message 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 MPC8000 family.

1. Verify that the emulation module is connected to the target.
2. Next, check that your emulation module is programmed with firmware for the Motorola MPC8000:

   See "To display the emulation module firmware version information" on page 100. If the emulation module is not programmed with the proper firmware, see "To update emulation module firmware" on page 99.

"command socket connection failed: WSAECONNREFUSED: connection refused"

This message usually means the emulation module is not at port #6470. See "To change the port number of an emulation module" on page 195.

"unrecognized hostname"

This message 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 emulation probe or to the logic analysis system.

  If you are unable to ping the emulator or logic analysis system, refer to "Solving LAN Communication Problems" on page 252 or the logic analysis system on-line help, respectively, for more information.

See Also

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

The *SDS SingleStep Users Guide*.

See "Configuring the Emulation Module" on page 177 for more information on configuration options and the "cf" command.

12

Coordinating Logic Analysis with Processor Execution

This chapter describes how to use a logic analyzer, an emulation module, and other features of your HP 16600A/16700A-series logic analysis system to gain insight into your target system.

What are some of the tools I can use?

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

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

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

- The 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. You can 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 on-line help contains quick reminders of how to perform common measurements.

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

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

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

Stopping Processor Execution on a Logic Analyzer Trigger

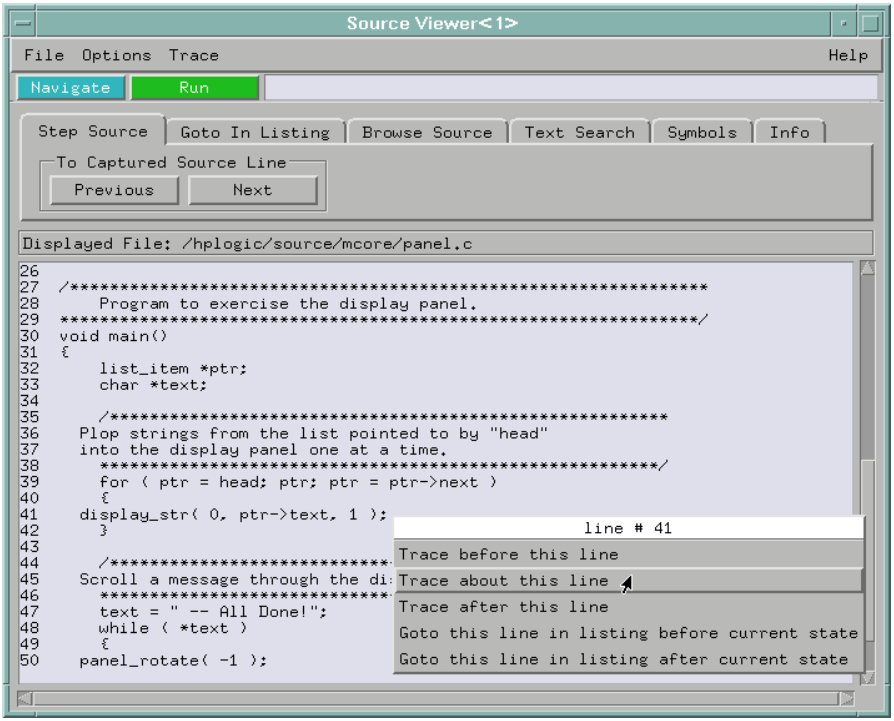
You can trigger the emulation module from the logic analyzer using either the Source Viewer window or the Intermodule window. If you are using the HP B4620B Source Correlation Tool Set, using the Source Viewer window is the easiest method.

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

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

1 In the Source 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.



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

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

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

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

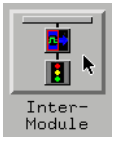
4 If your target system is not already running, click **Run** in the emulation Run Control window to start your target.

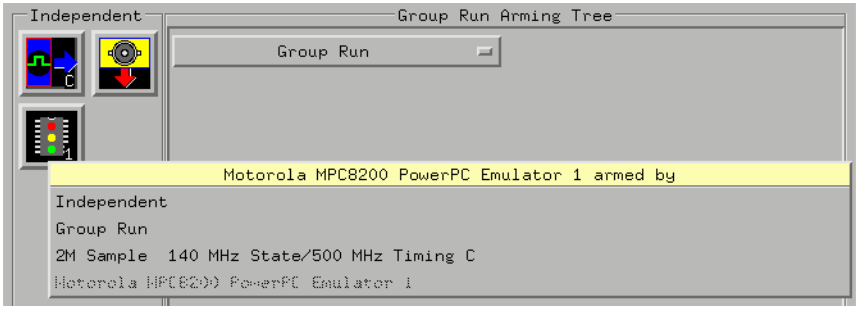
To stop the processor when the logic analyzer triggers (Intermodule window)

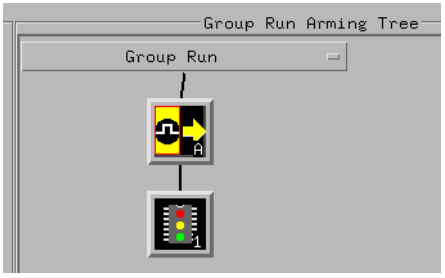
Use the Intermodule window if you do not have the HP B4620B Source Correlation Tool Set or if you need to use a more sophisticated trigger than is possible in the Source Viewer window.

1 Create a logic analyzer trigger.

2 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.

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

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

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

To minimize the "skid" effect

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

To minimize the skid effect:

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

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

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

See Also "To configure reset operation" on page 182.

To stop the analyzer and view a measurement

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

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

Example

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

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

   If the trigger position is not "End", the analyzer may continue waiting for more states.
6. The user 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. This kind of measurement is easier than setting up an intermodule measurement trigger.

Some possible uses for this measurement are:

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

To capture a trace before the processor halts

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

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

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

Triggering the Logic Analyzer when Processor Execution Stops

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

If you are only using a state analyzer to capture the processor bus, it will be much simpler to trace until a processor halts (see "Tracing Until the Processor Halts" on page 227).

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.
- In addition, an "In Monitor" signal is generated any time a debugger or other user interface reads a register, reads memory, sets breakpoints or steps. Care must be taken to not falsely trigger the logic analyzers listening to the "In Monitor" signal.

Group Run

The intermodule bus signals can still be active even without a Group Run. The following setups can operate independently of Group Run:

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

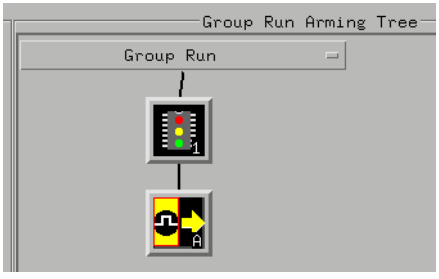
Here are some examples:

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

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

Group Run into an emulation module does not mean that the Group Run will Run the emulation module. The emulation module Run, Break, Step, and Reset are independent of the Group Run of the Analyzers.

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



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 which appear in the listing.

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

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

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

To trigger the analyzer when the processor halts

Remember: if you are only using a state analyzer to capture the processor bus then it will be much simpler to trace until a processor halts (see "Tracing Until the Processor Halts" on page 227).

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. 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.

NOTE: 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 stopped.

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

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

7 If necessary, in the logic analyzer window, 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 logic analyzer, click **Stop** if needed to complete the measurement.

To trigger the analyzer when the processor reaches a breakpoint

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

Remember: if you are only using a state analyzer to capture the processor bus, it will be much simpler to trace until a processor halts (see "Tracing Until the Processor Halts" on page 227).

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 which insert an illegal instruction into your program at the breakpoint location you will not need to do the Reset, Break sequence. Instead, you must take care to properly insert your software breakpoint in your RAM program location.
5. 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.

NOTE: 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 up until the processor stops, but may continue running.

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

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.

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 239
Emulation module built-in commands do not work	1 Check that the emulation module has been properly configured for your target system.	page 177
	2 Run the emulation module performance verification tests.	page 254
	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 this manual.	page 29, page 58
"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 182
Some commands fail	Check the "restrict to real-time runs" configuration.	page 183

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)

Emulator Built-In Commands

The emulator has some built-in commands (sometimes called the "terminal interface") which you can use for troubleshooting.

You can enter the built-in commands 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 16600A-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:

Useful built-in commands

b	Break—go into the background monitor state
cf	Configuration—read or write configuration options
help	Help—display on-line 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
mtest	Memory test—test target memory system
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
p	No target power
R	Emulation reset
c	Checkstop
r	Target reset
?	Unknown state

Examples

To set register r0, then view r0 to verify that is was set, enter:

```
U>rst
M>b
M>reg r0=1234abcd
M>reg r0
  reg r0=1234abcdf
```

To break execution then step a single instruction, enter:

```
U>b
M>s
 PC=fff00108
M>
```

To determine what firmware version is installed in the emulator, 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.

Solving Problems with the Target System

What to check first

Verify that the cf options are correct for your target.

- Try some basic built-in commands using the Command Line window or a telnet connection:

```
U>rst
U>
```

This should reset the target and display a "U>" prompt if 'cf reset=run'.

```
U>rst
M>
```

This should reset the target and display an "M>" prompt if 'cf reset=stop'.

```
M>m 0..=abcd1234
M>m 0..
  00000000 abcd1234 abcd1234 abcd1234 abcd1234
  00000010 abcd1234 abcd1234 abcd1234 abcd1234
  00000020 abcd1234 abcd1234 abcd1234 abcd1234
  00000030 abcd1234 abcd1234 abcd1234 abcd1234
  00000040 abcd1234 abcd1234 abcd1234 abcd1234
  00000050 abcd1234 abcd1234 abcd1234 abcd1234
  00000060 abcd1234 abcd1234 abcd1234 abcd1234
  00000070 abcd1234 abcd1234 abcd1234 abcd1234
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.

1 Check that the emulation module firmware matches your processor. To do this, enter:

```
M>ver
```

See Also "Emulator Built-In Commands" on page 239 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	Vdd
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	Vdd
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

If JTAG signals are disconnected

If the JTAG port on the target board seems to be disconnected from the microprocessor, and an analysis probe is connected to the target system:

- ❑ Check that the LED next to the JTAG port on the analysis probe is lighted. The analysis probe must be connected to at least one logic analyzer cable to provide power.
- ❑ Check that the DIP switch on the analysis probe is configured to connect the JTAG signals to the target board. (See page 96.)

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
M>cf proc=MPC8240
```

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 "?>"	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 $\overline{\text{TRST}}$ signals. Check the firmware revision.
If the initial prompt is “U>”	Processor is running and the emulation module is scanning the instruction register correctly.

Now you can do some more tests:

3 Enter the reset command:

```
U>rst
U>
```

If the prompt after rst is "?>"	A bad status code (0xXX) was received from the processor. Valid status is 0x01, any other status indicates bad scan of the instriction register 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=stop" (no external bus cycles used in this mode), then enter the "rst" command again: *>cf reset=stop *>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. 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, 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.

If you can get to the "M>" prompt, continue with more tests:

5 At the "M>" prompt , check register and memory access:

```
M>reg r0=12345678
M>reg r0
  reg r0=12345678
M>
```

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>
```

If the target memory system is configured, this should write abcd1234 to memory starting at 0 and then read back the same values.

- 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=run;rst;w 5
  #waiting for 5 seconds...
U>b
M>
```

- Repeat above memory test.
- Check memory access size using the mo command:

```
M>mo
```

- Check access size matches target memory -a0. The default access size -a0 is the 64-bit access.

```
M>mo -d4 -a0
```

- Repeat above memory test.

6 At the "M>" prompt , check the processor's revision level:

For example:

```
M> reg pvr
   reg pvr=xxxxxxxx
M>
```

At the time of this release the pvr part number is 00910101 or 00810101.

If you see memory-related problems

1 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>
```

- RAM must be at address 0.
- Read value not equal to the written value implies that the memory controller is not setup correctly.

2 Hand load a little program:

```
start: addi r1,1 - 0x38210001
       nop       - 0x60000000
       nop       - 0x60000000
       bra start - 0x4bffffff4
```

The opcode 0x4bffffff4 is a branch to a relative offset, so this program can be placed at any start address.

```
M>reg r1=0
M>m -a2 -d2 10000=3821,1,6000,0,6000,0,4bff,fff4
M>r 10000
U>reg r1
  reg r1=00034567 # or some number
U>reg r1
  reg r1=00102333 # or some number
U>
```

This program will loop forever, incrementing r1. This is a good test program to load once a memory system is up to make sure the microprocessor can run code out of memory.

If running from reset causes problems

Running from reset may cause some problems once background is entered. To ensure proper operation, the DER register must have bits 31,30,29,28 set (0x0000000f), and the SYPCR register must have the 'Disable watchdog freeze' bit set (0x00000080).

If you see the "!ASYNC_STAT 173!" error message

If after a break, the following error arises:

```
!ASYNC_STAT 173! MSR.RI bit not set - Break may not be
recoverable
```

This indicates that the MSR.RI bit is not set, implying that a non-maskable break was needed, and the interrupt may not be recoverable. If this occurs while breaking out of regular code, then the MSR.RI bit was not set in the boot code. This can be fixed by 'ORing' in 0x00000002 into the SRR1 register and

resuming the run.

If you see the "!ERROR 145!" error message

If the following error arises:

```
!ERROR  145! Unable to soft stop - freezing the processor
clocks
```

The processor is in an unstable state and should be reset via a hard reset or a power-on reset. The 'rst' command may not be able to bring the processor into a known state.

Solving LAN Communication Problems

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 LEDs 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.
- ❏ Refer to the online help in the logic analysis system for more information.

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.

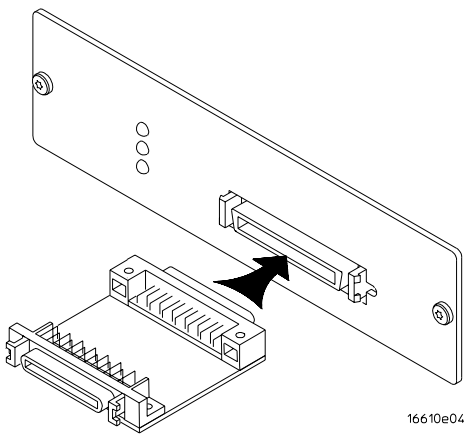
If there are many subnet masks in use on the local subnet, the logic analysis system may take a very long time to connect to the network after it is turned on.

Solving Emulation Module Problems

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 performance verification tests 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 on screen.

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: HPE3499B 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!
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.


  HPE3499B Series Emulation System
     Version:   A.07.56 28Sep98
     Location:  Generics

  HPE3453A Motorola MPC8200 Embedded PowerPC Emulator
     Version:   A.01.00 18Feb99

M>
```

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.

- ❏ Check that the loopback test board is connected.
- ❏ If the problem persists, contact HP for assistance.

If the particular failure you see is not listed below, contact HP for assistance.

TEST 5: Target Probe Feedback TestTEST 6: Boundary Scan Master Test
TEST 7: I2C Test
TEST 8: Data Line 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.

Part 4

Reference

14

Specifications and Characteristics

Analysis Probe Operating Characteristics

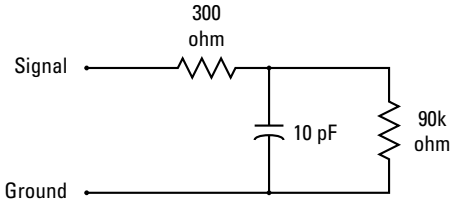
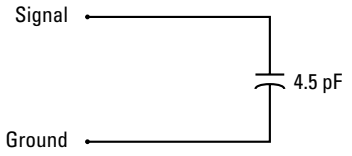
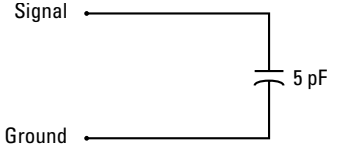
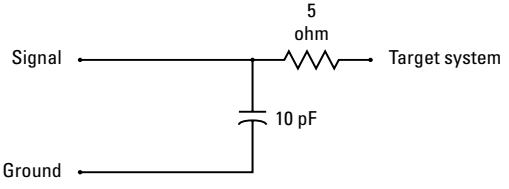
The following operating characteristics are not specifications, but are typical operating characteristics for the HP E8128A analysis probe for the MPC8240 PowerQUICC II.

Operating Characteristics	
Microprocessor Compatibility	Motorola MPC8240
Package Supported	BGA
Logic Analyzers Supported	HP 16550A HP 16554/55/56/57 HP 16600/1/2/3A HP 16710/11/12A HP 16715/16/17A
Accessories Required	For state and timing analysis, the HP E8161A BGA probing kit and the HP E5346A high-density cables are required.
Optional Accessories	The HP 16610A emulation module or the HP E3453A emulation probe can be connected to the analysis probe.
Probes Required	At least four 16-channel probes are required for disassembly.

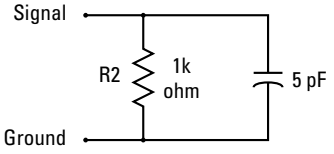
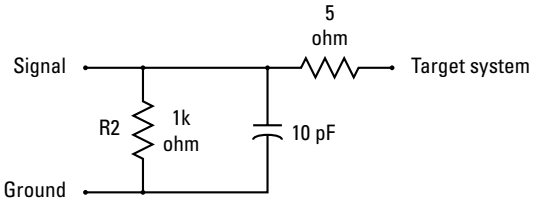
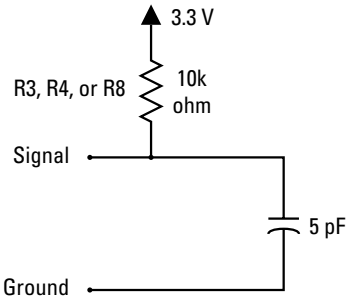
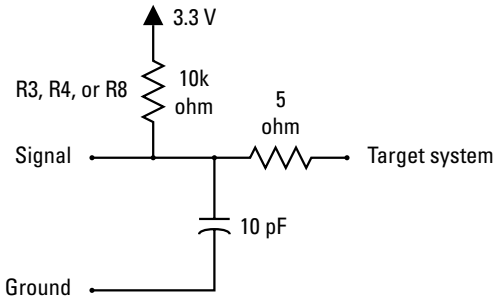
Electrical Characteristics	
Power Requirements	300mA @ 5.0V, supplied by the logic analyzer, CAT 1, Pollution degree 2. Approximatley 0.1*u*F decoupling on GVdd, LVdd, 2.5V (Vdd), and 3.3V (OVdd) supplies. Maximum draw of 4mA from target system 3.3V (OVdd).
Signal Line Loading	(See the processor signal line loading table that follows.)

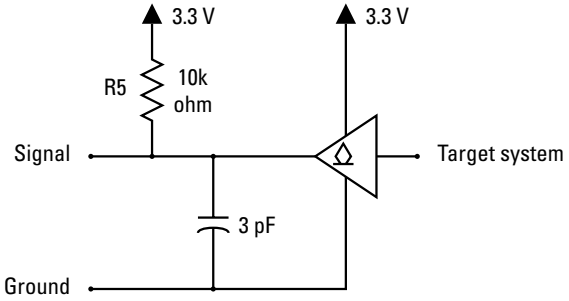
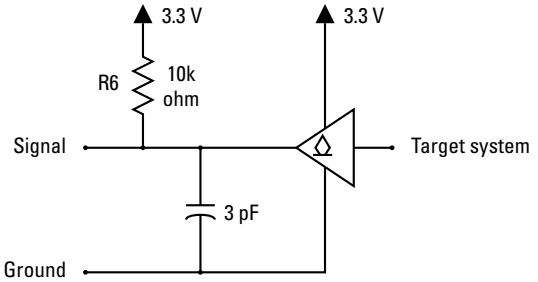
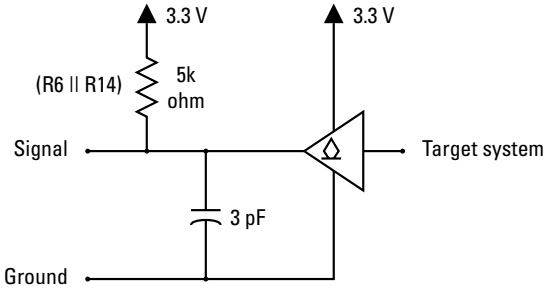
Environmental Characteristics	
Temperature, Operating	0 to + 50 degrees C +32 to +122 degrees F
Altitude, Operating	4,600 m 15,000 feet
Humidity	Up to 75% noncondensing. Avoid sudden, extreme temperature changes which could cause condensation on the circuit board.
Pollution	IEC pollution degree 2. Normally only dry non-conductive pollution occurs. Occasionally a temporary conductivity caused by condensation may occur. Indoor use only.

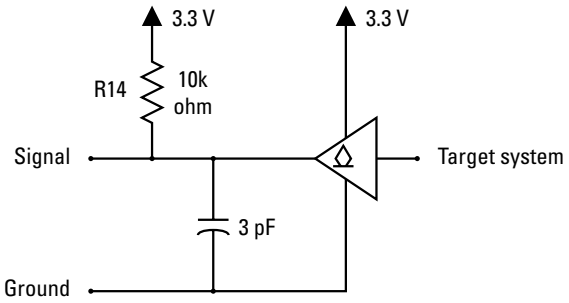
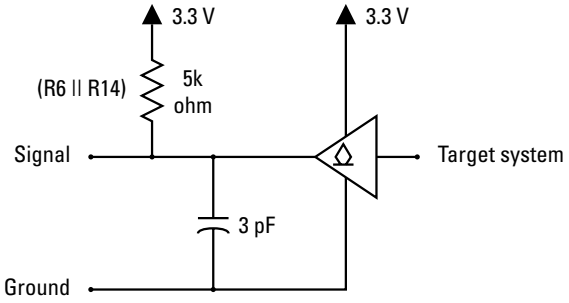
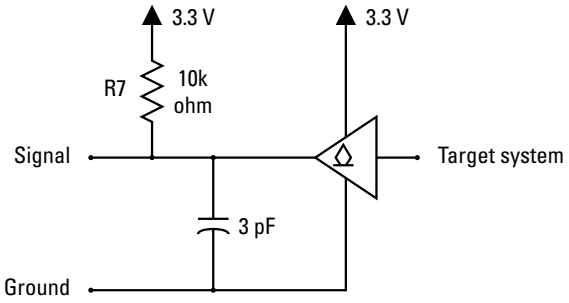
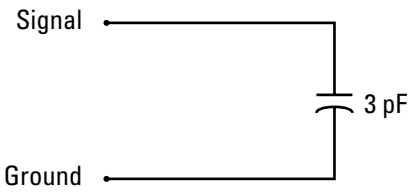
Processor Signal Line Loading

Signal	MPC 8240 Pin	Analysis Probe DIP Switch	Equivalent Load (includes logic analyzer)
Any probed signal		N/A	 (HP E5346A)
SDRAM_ CLK[0]	D1	N/A	
TDO*	AC21	S1 = OFF	
		S1=ON	

*Signal is on the JTAG port. The equivalent load does not include the emulation module /probe.

Signal	MPC 8240 Pin	Analysis Probe DIP Switch	Equivalent Load (includes logic analyzer)
TDI*	AF23	S1=OFF	
		S1=ON	
TCK* TMS* TRST*	AF22 AE22 AE23	S1 = OFF	
		S1 = ON	

Signal	MPC 8240 Pin	Analysis Probe DIP Switch	Equivalent Load (includes logic analyzer)
$\overline{\text{SRESET}}$*	B16	N/A	
$\overline{\text{HRST\_CPU}}$*	A19	S2 = OFF	
		S2 = ON	

Signal	MPC 8240 Pin	Analysis Probe DIP Switch	Equivalent Load (includes logic analyzer)
HRST_CTRL*	A20	S2 = OFF	
		S2 = ON	
CHKSTOP_IN	D14	N/A	
All other Signals		N/A	

Emulation Module Operating Characteristics

The following operating characteristics are not specifications, but are typical operating characteristics for the HP 16610A emulation module and MPC8000 target interface module.

Operating Characteristics	
Microprocessor Compatibility	Motorola MPC8240 and MPC8260 PowerQUICC II microprocessors.
Environmental Characteristics (Temperature, Altitude, Humidity)	The HP 16610A emulation module meets the environmental characteristics of the logic analysis system in which it is installed. For indoor use only.

Emulation Module Electrical Characteristics

Signal	Characteristic	Symbol	Min	Max	Unit
TDO, $\overline{\text{CHECKSTOP}}$	Input High Voltage	V_{ih}	2.0	5.5	V
	Input Low Voltage	V_{il}		0.8	V
	Input Current	I_i		+/- 1	µA
	Input Capacitance	C_{in}		15	pF
TDI, TCK, TMS, $\overline{\text{TRST}}$[1]	Output High Voltage	V_{oh} @ I_{oh} = -32 ma	2.0	2.8	V
	Output Low Voltage	V_{ol} @ I_{ol} = 64 mA; V_{CC} = 4.5V		0.55	V
TDI, TMS, $\overline{\text{TRST}}$	Output Capacitance	C_o		25	pF
TCK	Output Capacitance	C_o		45	pF
+3.3 V Power Sense[2]	Input High Voltage	V_{ih}	2.0	5.3	V
	Input Low Voltage	V_{il}	-0.3	0.8	V
$\overline{\text{SRESET}}$, $\overline{\text{HRESET}}$[3]	Output Low Voltage	V_{ol} @ I_{ol} = 12 mA		0.5	V
	Output Capacitance	C_o		25	pF

Notes:

[1] These signals must not be actively driven by the target system whenthe debug port is being used.

[2] Power Sense is used only to determine target powered status. The emulator does not draw power from this source.

[3] Open drain outputs, pulled up to a generated voltage equivalent to the Power Sense voltage with a 2.61 KΩ pullup resistor.

15

General-Purpose ASCII (GPA) Symbol File Format

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

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

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

You can typically use 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.
```

Part 5

Service Guide

To return a part to Hewlett-Packard for service

1. Follow the procedures in the "Troubleshooting..." chapters 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 get replacement parts

The repair strategy for the emulation module is board replacement. However, the following tables list some mechanical parts that may be replaced if they are damaged or lost. Contact your nearest Hewlett-Packard Sales Office for further information.

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 allows you to 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.

Analysis Probe Replaceable Parts

HP Part Number	Description
E5346A	High-density cable
E8127-66501	Analysis probe circuit board
E8127-87606	Double header
E8127-87607	Extender
E8125-03801	Pry tool
E8161-60001	Probing kit

Emulation Module Exchange Assemblies

HP Part Number	Description
16600-66515	Emulation module

Emulation Module Replaceable Parts

HP Part Number	Description
E3494-61604	16-pin target cable
E3496-61601	50-pin control cable

HP Part Number	Description
16700-61608	Emulation module interface cable
E3452-66502	Target Interface Module
E3496-66502	Loopback test board

These part numbers are subject to change without notice.

To clean the instrument

If the instrument requires cleaning:

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

A

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."

D

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.

E

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 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 also *emulation probe*.

emulation probe An emulation probe is a stand-alone 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 also *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."

F

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

G

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.

H

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.

I

inverse assembler Software that decodes microprocessor bus states (captured by the login analyzer) into assembly language mnemonics. Typically, inverse assemblers are included with analysis probes, but when the processor can be in any type of chip package, as is the case with microprocessor cores, the inverse assembler is a separate product and connections for the logic analyzer are designed into the target system.

J

JTAG (OnCE) port See *debug port*.

jumper Moveable direct electrical connection between two points.

L

label A name that you assign to a number of logic analysis channels. Typically, these names map to signal and/or bus names in the target system.

M

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.

P

preprocessor See *analysis probe*.

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

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/16700A-series logic analysis systems.

R

run control probe See *emulation probe* and *emulation module*.

S

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 stand-alone logic analyzer has a predefined set of hardware components which provide a specific set of capabilities. It is designed to perform logic analysis. A stand-alone 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.

T

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 (TIM) 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.

trigger specification A set of conditions that must be true before the instrument triggers. See the printed or on-line documentation for your logic analyzer for details.

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.

1

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

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 Motorola MPC8240 User's Guide*.

Publication number
E8128-97003, December 1999
Printed in USA.

Print history is as follows:
E8128-97002, October 1999
E8128-97000, March 1999
E8128-97001, July 1999

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.