User's Guide

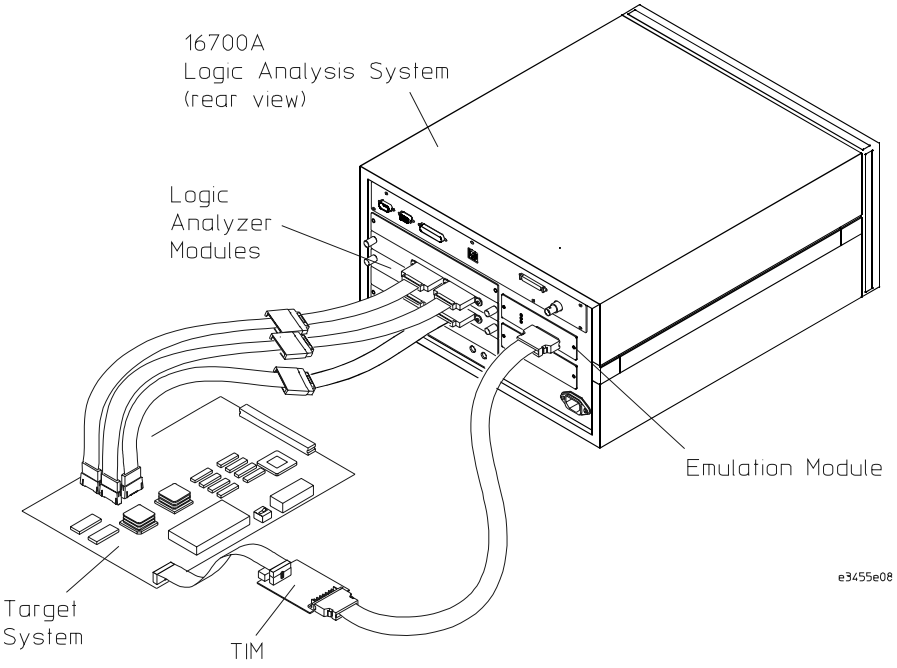
Publication number E2631-97001
June 2000

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

Solutions for the Motorola M•CORE

The Agilent Technologies E9512A emulation solution lets you use the Agilent Technologies 16600/16700A-series logic analysis system to debug and characterize Motorola M•CORE (with RLB Integration Module) target systems.

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



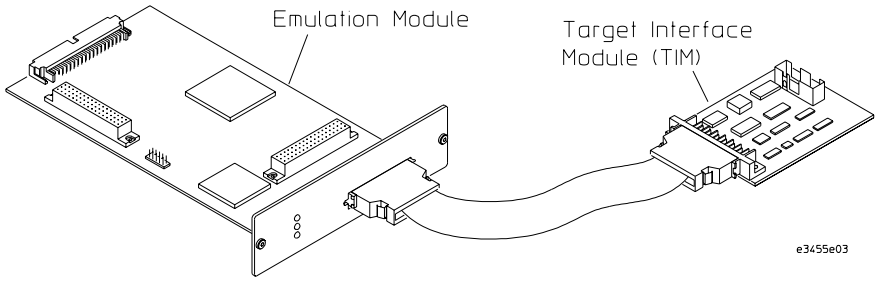
Inverse Assembler

The inverse assembler for M•CORE processors with the RLB Integration Module (RIM), along with 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 Agilent Technologies E9612A Option 001 when ordered separately.)

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).

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



Source Correlation Tool Set

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

In This Book

This book describes the following products:

Product	Supports	Includes
Agilent Technologies E9612A Option 001 inverse assembler	M•CORE with RIM	The Agilent Technologies E2631A inverse assembler
Agilent Technologies E9512A Option 001 emulation solution	M•CORE with RIM	Agilent Technologies E9612A Option 001 inverse assembler, Agilent Technologies 16610A emulation module, target interface module (TIM), and Agilent Technologies B4620B source correlation tool set

Before you use this book, you should have already set up the Agilent Technologies 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," describes supplied and required equipment, target system design considerations, setting up the logic analysis system, and probing the target system.
- Part 2, "Using the Logic Analyzer," describes logic analyzer and inverse assembler configuration, how to set up triggers, how to interpret the captured data when it's displayed, and how to troubleshoot problems.
- Part 3, "Using the Emulation Module," describes how to use the Emulation Control Interface, how to configure the emulation module, how to use debuggers, how to coordinate emulation control and logic analysis, and how to troubleshoot emulation module problems.
- Part 4, "Reference," describes specifications and characteristics and the general-purpose ASCII symbol file format.
- Part 5, "Service Guide," describes how to return parts for service, how to get replacement parts, and how to clean the instrument.

See Also

The Agilent Technologies 16600A/16700A-series logic analysis system's on-line help for more information on using the Agilent Technologies B4620B source correlation tool set.

Contents

Solutions for the Motorola M•CORE—At a Glance

Inverse Assembler 3
Emulation Module (and Target Interface Module) 3
Source Correlation Tool Set 3

In This Book

Part 1 Installation Guide

Overview of Installation and Setup 16

1 Equipment and Requirements

Equipment and Software Supplied 20
Inverse Assembler 20
Emulation Module 20
Source Correlation Tool Set 21

Additional Equipment and Software Required 22
Inverse Assembler 22
Emulation Module 22
Source Correlation Tool Set 22

Other Optional Equipment and Software 23

Contents

2 Preparing the Target System

Preparing for Logic Analysis (and Inverse Assembly) 26
RLB Integration Module (RIM) Support Only 26
Supported Logic Analyzers 31
Choosing the Type of Logic Analyzer Connector 32
Using High-Density Connectors 32
Using Medium-Density Connectors 36

Preparing for Emulation 38
Required Signals for JTAG Emulation 38
Optional Signals for JTAG Emulation 38
JTAG Connector Mechanical and Pinout Information 39

3 Setting Up the Logic Analysis System

Power-ON/Power-OFF Sequence 42
To power-ON the 16600A/16700A-series logic analysis systems 42
To power-OFF 42

Installing Logic Analyzer Modules 43

Installing the Emulation Module 44
To install in a 16700A-series logic analysis system 44
To install in a 16600A-series logic analysis system 47
To test the emulation module 49

Installing Software 50
To install software from CD-ROM 51

Using the Setup Assistant 53

Contents

4 Probing the Target System

Connecting the Logic Analyzer to the Target System 56
To connect the 16550A logic analyzer 57
To connect a one-card 16554/55/56/57 logic analyzer 58
To connect a two-card 16554/55/56/57 logic analyzer 59
To connect the 16600A logic analyzer 61
To connect the 16601A logic analyzer 62
To connect the 16602A logic analyzer 63
To connect the 16603A logic analyzer 64
To connect the 16710/11/12A logic analyzers 65

Connecting the Emulation Module to the Target System 66
To connect to a target system JTAG (OnCE) port 67
To update emulation module firmware 69
To display the emulation module firmware version information 70
To verify communication with the target system 70

Part 2 Using the Logic Analyzer

5 Configuring the Logic Analyzer

Loading Configuration Files 74
To load configuration files (and the inverse assembler) 74
To set the inverse assembler preferences 77

Loading Symbol Information 80
To view predefined symbols for the M•CORE 80
To load object file symbols 81

Changing the Analysis Mode 84
To change to state analysis 84
To change to timing analysis 85

6 Capturing M•CORE Execution

Setting Up Logic Analyzer Triggers 89
To set up logic analyzer triggers 89
To compensate for relocated code 90

Triggering on Source Code 92
To set up triggers based on source code 92
To avoid capturing library code execution 93

7 Displaying Captured M•CORE Execution

To display the captured state data 96
To display symbols 97
To interpret the inverse assembled data 97
To use the inverse assembler filters 98
To view the source code associated with captured data 101
To display captured timing analysis mode data 104

8 Troubleshooting the Logic Analyzer

Solving Logic Analyzer Problems 109
Intermittent data errors 109
Unwanted triggers 109
No activity on activity indicators 110
No trace list display 110
Analyzer won't power up 110

Contents

Solving Probing Problems 111
Target system will not boot up 111
Erratic trace measurements 112
Capacitive loading 112

Solving Inverse Assembler Problems 113
No inverse assembly or incorrect inverse assembly 113
Inverse assembler will not load or run 114

Solving Intermodule Measurement Problems 115
An event wasn't captured by one of the modules 115

Logic Analyzer Messages 116
". . . Inverse Assembler Not Found" 116
"Measurement Initialization Error" 116
"No Configuration File Loaded" 118
"Selected File is Incompatible" 118
"Slow or Missing Clock" 118
"Time from Arm Greater Than 41.93 ms" 119
"Waiting for Trigger" 119

Part 3 Using the Emulation Module

9 Using the Emulation Control Interface

To start from the main System window 125
To start from the Workspace window 126

10 Configuring the Emulation Module

Entering Emulation Module Commands 129
To use the Emulation Control Interface 129
To use the built-in command interface 131
To use a debugger interface 132

Setting the M•CORE Configuration Options 133
To configure the processor type 133
To configure the JTAG clock speed 134
To configure the “Break In” type 135
To configure restriction to real-time runs 135

Testing the Emulation Module and Target System 136
To test memory accesses 136
To test by running a program 137

11 Using Debuggers (with the Emulation Module)

Setting Up Debugger Software 142
To connect the logic analysis system to the LAN 143
To change the port number of an emulation module 144
To verify LAN communication with the emulation module 144
To view logic analysis system windows next to the debugger 145

Using the Software Development Systems Debugger 147
To get started 148
To send commands to the emulation module 150
On-chip breakpoints and debugging ROM code 151
Error conditions 151

Contents

12 Coordinating Logic Analysis with Processor Execution

Stopping Processor Execution on a Logic Analyzer Trigger 156
To stop on a source line trigger (Source Viewer window) 156
To stop on any trigger (Intermodule window) 158
To minimize the "skid" effect 159
To stop the analyzer and view a measurement 159

Tracing Until the Processor Halts 161
To capture a trace before the processor halts 161

Triggering the Logic Analyzer when Processor Execution Stops 163
To trigger the analyzer when the processor halts 166
To trigger the analyzer when the processor reaches a breakpoint 167

13 Troubleshooting the Emulation Module

Troubleshooting Guide 171

Status Lights 172

Built-In Commands 173
To telnet to the emulation module 173
To use the built-in commands 174

Solving Target System Problems 176
What to check first 176
To interpret the initial prompt 177
If you see memory-related problems 178

Solving LAN Communication Problems 180
If LAN communication does not work 180
If it takes a long time to connect to the network 180

Contents

Solving Emulation Module Problems 182
To run the performance verification tests using the logic analysis system 182
To run complete performance verification tests using a telnet connection 183
If a performance verification test fails 184

Part 4 Reference

14 Specifications and Characteristics

Emulation Module Operating Characteristics 190
Emulation Module Electrical Characteristics 190

15 General-Purpose ASCII (GPA) Symbol File Format

GPA Record Format Summary 193
SECTIONS 194
FUNCTIONS 195
VARIABLES 196
SOURCE LINES 197
START ADDRESS 197
Comments 198

Part 5 Service Guide

To return a part to Agilent Technologies for service 200
To get replacement parts 201
To clean the instrument 202

Glossary

Contents

Index

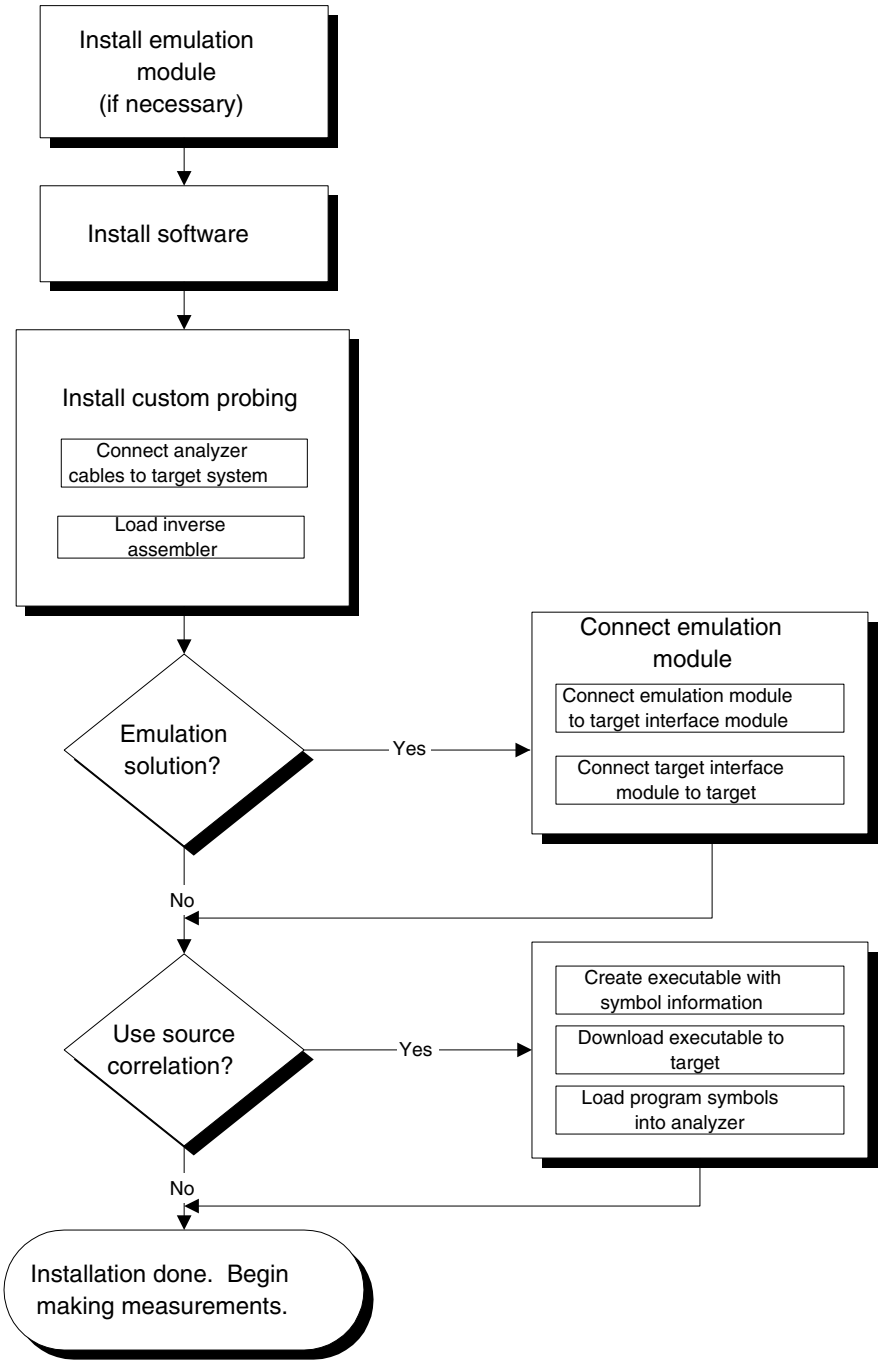
Part 1

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 "Equipment and Requirements" chapter on page 19.
2. If you need to install an emulation module in an Agilent Technologies 16600A/16700A-series logic analysis system, see "Installing the Emulation Module" on page 44.
3. Install the software. See "Installing Software" on page 50.
4. If you have an Agilent Technologies 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 53.


Install emulation module (if necessary)
Install software
Install custom probing
Connect analyzer cables to target system
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.

1

Equipment and Requirements

Equipment and Software Supplied

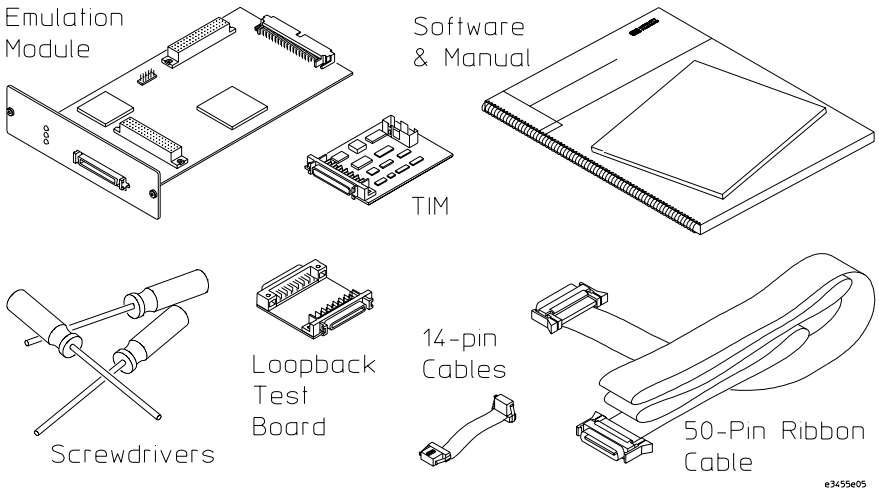
Listed below is the equipment and software supplied with the Agilent Technologies E9512A Option 001 M•CORE emulation solution (which includes an inverse assembler, emulation module, and source correlation tool set).

Inverse Assembler

The inverse assembler (Agilent Technologies E9612A Option 001 when ordered separately) includes:

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

Emulation Module



The emulation module includes:

- An Agilent Technologies 16610A emulation module.

  If you ordered an emulation module as part of your Agilent Technologies 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 14-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 (Agilent part number E3496-66502).

Source Correlation Tool Set

The source correlation tool set includes:

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

The Agilent Technologies B4620B source correlation tool set software is already installed on the Agilent Technologies 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 Agilent Technologies E9512A Option 001 emulation solution for the M•CORE with RIM.

Inverse Assembler

The inverse assembler 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 to capture M•CORE processor execution.

Refer to the the "Preparing the Target System" chapter on page 25 for more information on headers, terminations, and supported logic analyzers.

Emulation Module

The emulation module requires:

- An Agilent Technologies 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 Agilent Technologies 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 Agilent Technologies 16600A/16700A-series logic analysis system.

Other Optional Equipment and Software

The emulation module works with several debuggers offered by other vendors.

2

Preparing the Target System

Preparing for Logic Analysis (and Inverse Assembly)

The M•CORE inverse assembler for the Agilent Technologies 16600/16700-series logic analysis system is only for M•CORE processors with the RLB Integration Module (RIM).

In order to provide inverse assembled trace listings, the logic analyzer must probe certain RIM signals. Probing optional RIM signals will enhance the capability of the inverse assembler and improve the triggering and store-qualification of the logic analyzer. However, you may decide not to probe optional signals because of cost, pin, or space constraints.

You can design connectors in your target system for logic analyzer probe pods. The high-density connectors take the least amount of circuit board space. You can also use medium-density connectors which take more circuit board space and have some clock speed limitations.

This section describes these considerations in more detail.

- RLB Integration Module (RIM) Support Only
- Supported Logic Analyzers
- Using High-Density Connectors
- Using Medium Density Connectors

RLB Integration Module (RIM) Support Only

In the M•CORE architecture, the RISC Local Bus (RLB) is the interface between the core and peripherals. In the case of a bonded-out M•CORE microprocessor, the RLB signals will reach external pins. However, the RLB is usually internal to an ASIC and another memory controller is used, such as the RLB Integration Module (RIM), or the External Interface Module (EIM).

The M•CORE inverse assembler works with RLB Integration Module

(RIM) memory controller. The M•CORE inverse assembler does not support MMC2001 and EIM based devices or the Rev 1.5 (RLB) Bondout.

The M•CORE inverse assembler supports the Emulation and Master Modes of the RIM Memory Controller. The Fast and Single Modes are not supported since the general-purpose ports are multiplexed to the external pins, instead of the address and data signals.

Although many devices have a RIM memory controller, the RIM bus must be brought to external pins for the inverse assembler to work. If the GPIO signals are enabled on the external pins, the inverse assembler will not work. A common debugging technique is to design in a PRU (Port Replacement Unit) which externally emulates the general-purpose ports, and allows the bus to be brought to the external pins.

Agilent Technologies supports any RIM version that adheres to the required signal list and timing diagrams. This includes the RIM 1.5 and RIM+ memory controllers.

Required RIM Signals

Each signal required by the inverse assembler is described below.

$\overline{\text{SHS}}$. The $\overline{\text{SHS}}$ (show strobe) signal is used by the logic analyzer to sample the address and data. The $\overline{\text{SHS}}$ signal is used for both external and internal accesses. The logic analyzer uses the following acquisition algorithm based on the $\overline{\text{SHS}}$ signal.

- Falling Edge: A[22:0], $\overline{\text{CS}}$[4:1], CSE[1:0], R/$\overline{\text{W}}$, TSIZ[1:0],TC[2:0]
- Rising Edge: D[31:0]

Emulation Mode: In emulation mode, the $\overline{\text{SHS}}$ signal is automatically provided as a strobe for capturing address and data during show cycles.

Master Mode: In master mode, the $\overline{\text{SHS}}$ signal is available after some RIM control registers are configured. In the Module Configuration Register (MCR, Address 0x00EFFF00), the SHEN bit (Bit 5) should be set to 1. In the Port E Pin Assignment Register (PEPAR, Address 0x00EFFF34), Bits 7-0 should all be set to 1. Configuring these registers will provide the proper signals for the inverse assembler.

A[22:1] - ADDR. The address signals must be present to capture accurate software flow. The address lines specify which memory location is being addressed. The address lines are sampled on the falling edge of the $\overline{\text{SHS}}$ signal. Note that some versions of the RIM memory controller do not have A0 brought to an external pin. For this reason the signal is optional.

D[31:16] - DATA. The most significant 16-bits of data are required for inverse assembly. The inverse assembler will work with either a 16-bit or 32-bit data bus. In the case of the 16-bit data bus, it is assumed there is a 16-bit memory port size. NOTE: Internal 32-bit accesses will not be supported if the lower data bits are not included.

$\overline{\text{CS[4:1]}}$. The $\overline{\text{CS}}$ (chip select) signals are used to decode different external memory banks. The inverse assembler will use these signals to determine whether an external or internal access has occurred and also to reconstruct the 32-bit logical address for the different external memory regions.

CSE[1:0]. The CSE (emulation chip select) signals are required to determine the nature of an internal access and to correctly decode accesses to a PRU (Port Replacement Unit).

R/$\overline{\text{W}}$. The R/$\overline{\text{W}}$ signal determines whether the current access is a read or a write cycle and will be used by the inverse assembler to distinguish instruction fetches from data writes.

TSIZ[1:0]. The TSIZ (transfer size) signals are required by the inverse assembler to determine the valid data on the bus.

Optional RIM Signals

Each optional signal, and the impact of its presence or absence, is described below.

A[0]. The least significant address pin is not required for inverse assembly. Some versions of the RIM memory controller do not bring A0 to an external pin. In this case, the byte enable signals EB[3:0] are required to properly decode 8-bit accesses to memory.

D[15:0]. The inverse assembler does not require the lower 16-bits of

data because the RIM memory controller can be configured for 16-bit memories. If these signals are present, the inverse assembler will analyze them and display the correct data. NOTE: Internal 32-bit accesses will not be supported if the lower data bits are not included.

$\overline{\text{EB[3:0]}}$. The $\overline{\text{EB[3:0]}}$ (Enable Byte) signals are not required by the logic analyzer. If the A[0] signal is available, the byte enables will not be used at all. However, if the A[0] signal is not available, the byte enables will be used to decode 8-bit accesses.

CLKOUT. The inverse assembler does not use the CLKOUT (Clock Out) signal to sample data, since the $\overline{\text{SHS}}$ signal is not synchronous to the clock. The CLKOUT signal can be of value for bus timing measurements or to sample the address and data on every clock state.

$\overline{\text{TEA}}$. The $\overline{\text{TEA}}$ (Transfer Error Acknowledge) signal is used to determine a failed bus cycle.

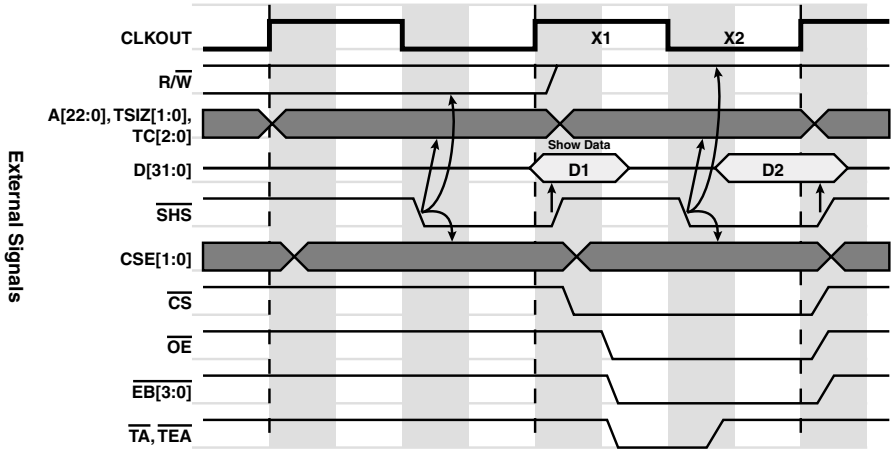
PSTAT[3:0]. The PSTAT signals give information about the current state of the internal pipeline. However, these signals are synchronous to the system clock and cannot be used in conjunction with the $\overline{\text{SHS}}$ signal. Therefore, the inverse assembler does not require them. The PSTAT signals are defined for an optional 5th pod and are intended to be used in a timing mode.

TC[2:0]. The TC (Transfer Code) signals are available on some versions of the RIM memory controller. If they are available, they will be used to determine whether the cycle is an instruction or data access. User memory map information is required if the signals are not available.

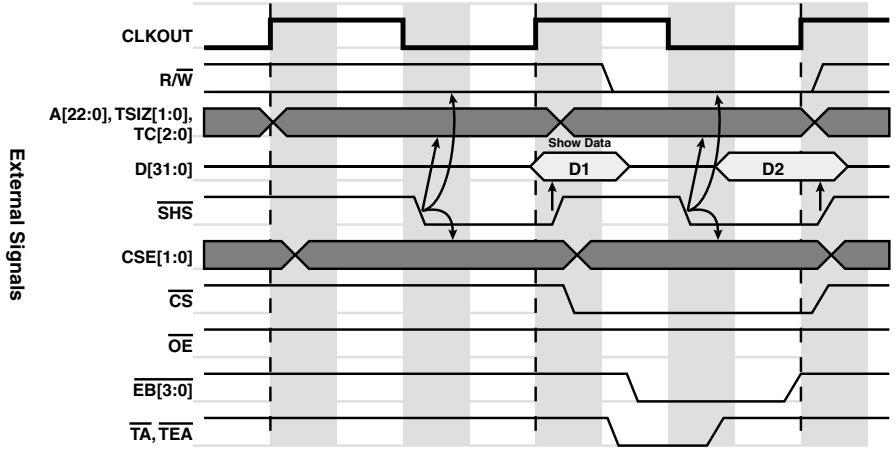
Required RIM Timing Relationships

The following diagrams show the required interaction between the bus signals for proper logic analysis and inverse assembly.

RIM - Internal show cycle followed by an external read cycle



RIM - Internal show cycle followed by an external write cycle



Supported Logic Analyzers

The number of RIM signals required by the inverse assembler means that four Agilent Technologies logic analyzer pods (68 channels) are required. If you choose to probe optional signals, an additional logic analyzer pod is required.

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

The inverse assembler works with logic analysis system software version A.01.20 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:

Agilent Technologies Logic Analyzer	Channel Count	State Speed	Timing Speed (Full/Half Channels)	Memory Depth (Full/Half Channels)
16550A	102/card	100 MHz	250/500 MHz	4K/8K samples
16554A (one or more cards)	68/card	110 MHz	125/250 MHz	500K/1M samples
16555A (one or more cards)	68/card	110 MHz	250/500 MHz	1M/2M samples
16555D (one or more cards)	68/card	110 MHz	250/500 MHz	2M/4M samples
16556A (one or more cards)	68/card	100 MHz	200/400 MHz	1M/2M samples
16556D (one or more cards)	68/card	100 MHz	200/400 MHz	2M/4M samples
16557D (one or more cards)	68/card	135 MHz	250/500 MHz	2M/4M samples
16600A	204	100 MHz	125/250 MHz	64K/128K samples
16601A	136	100 MHz	125/250 MHz	64K/128K samples
16602A	102	100 MHz	125/250 MHz	64K/128K samples
16603A	68	100 MHz	125/250 MHz	64K/128K samples
16710A	102/card	100 MHz	250/500 MHz	8K/16K samples
16711A	102/card	100 MHz	250/500 MHz	32K/64K samples
16712A	102/card	100 MHz	250/500 MHz	128K/256K samples

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.

Choosing the Type of Logic Analyzer Connector

M•CORE external bus signals are connected to a logic analyzer via a cable that attaches to a header connector that is designed into the target system. Items to consider when selecting a type of connector are:

- Board space required by the header connector.
- Ensuring that the logic analyzer is properly terminated.
- Ensuring that the signal pins connect to the proper logic analyzer probes.

Medium- or high-density connectors are available for connecting to an Agilent Technologies logic analyzer.

Pin Configuration	Density	Header Part Number	Required Termination	Notes
2 rows x 10 pins	Medium	Agilent: 1251-8106 or 3M: 2520-6002	Agilent part number: 01650-63203	Not recommended above 50MHz.
2 rows x 19 pins	High	Agilent: 1252-7431 or AMP: 2-767004-2	Agilent Technologies E5346A	Each connector supports two logic analyzer pods.

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

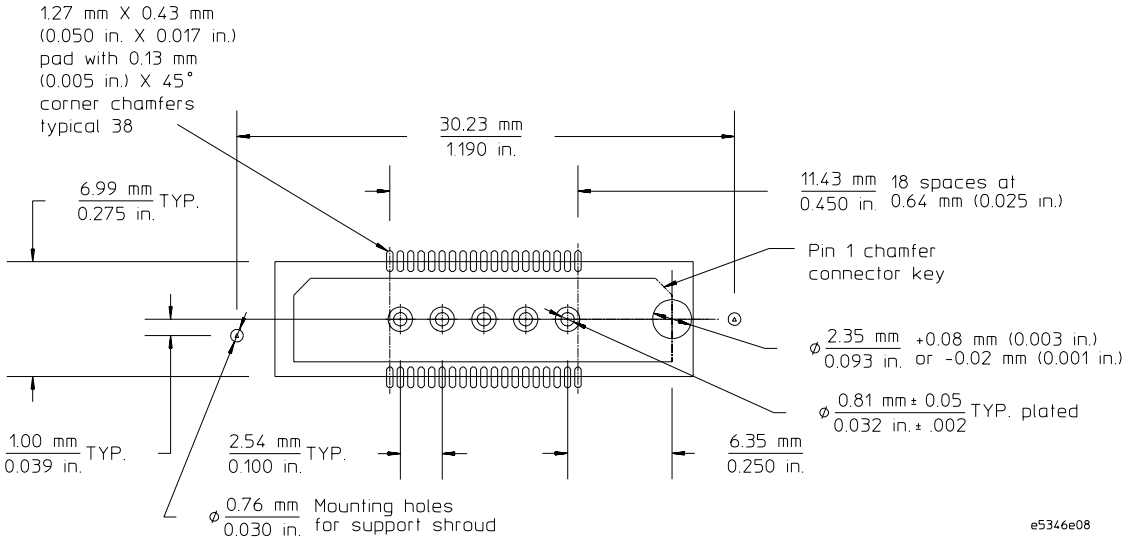
- Each Mictor connector requires one Agilent Technologies E5346A high-density termination adapter cable to attach to the logic analyzer. This is a Y-cable where the single end connects to the high-density header connector, and each of the two opposite ends connects to a logic analyzer pod.
- Any probed signal line must be able to supply a minimum of 600 mV to the probe tip and handle a minimum loading of 90 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 (Agilent part number E5346-60002) can be used to route the signals to the correct pods.
- A plastic shroud (Agilent part number E5346-44701) is available to secure the mechanical connection of the high-density cable to the Mictor header connector.

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

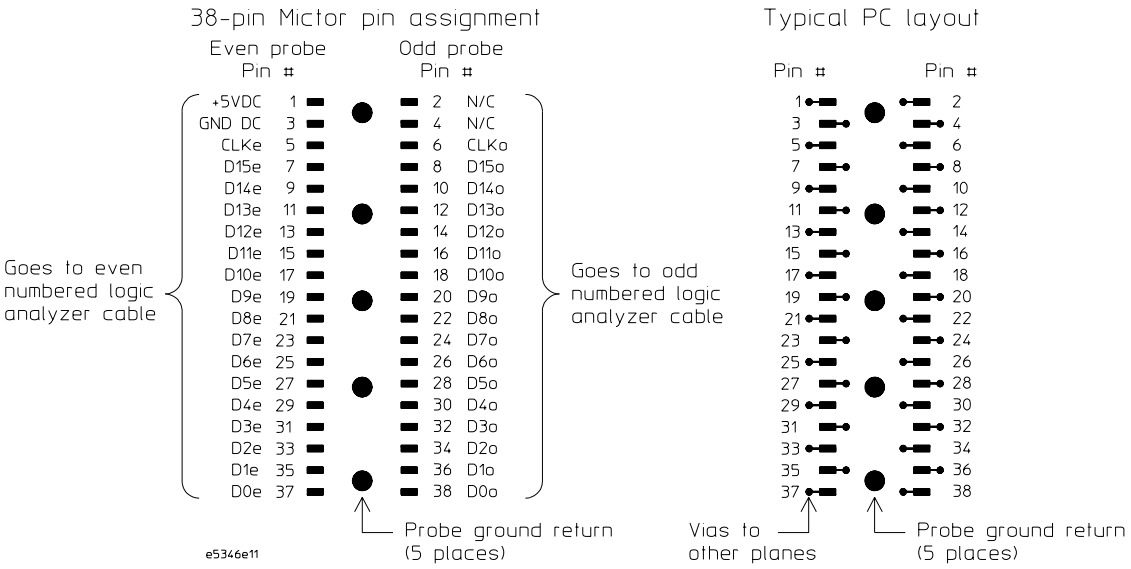
http://www.tm.agilent.com/tmo/datasheets/English/E5346A.html

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.

RIM Signal to High-Density Connector Pin Mapping

Pins 1 through 4 of the Mictor connectors (marked NC) must be a true no connect. The signals are used for other functions unavailable to target probing. They cannot be connected to anything on the target system, including ground.

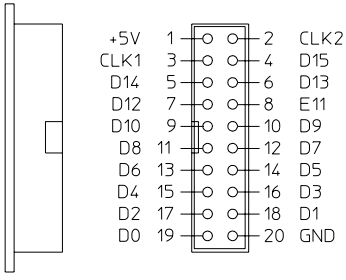
Mictor Connector 1				Mictor Connector 2				Mictor Connector 3 (Optional)			
Even		Odd		Even		Odd		Even		Odd	
Pin	M•CORE RIM Signal	Pin	M•CORE RIM Signal	Pin	M•CORE RIM Signal	Pin	M•CORE RIM Signal	Pin	M•CORE RIM Signal	Pin	M•CORE RIM Signal
1	NC	2	NC	1	NC	2	NC	1	NC	2	NC
3	NC	4	NC	3	NC	4	NC	3	NC	4	NC
5	$\overline{\text{SHS}}$	6	CLKOUT	5	$\overline{\text{TA}}$	6	$\overline{\text{TEA}}$	5	User Def.	6	User Def.
7	D31	8	D15	7	R/$\overline{\text{W}}$	8	A15	7	User Def.	8	TC[2]
9	D30	10	D14	9	CSE1	10	A14	9	User Def.	10	TC[1]
11	D29	12	D13	11	CSE0	12	A13	11	User Def.	12	TC[0]
13	D28	14	D12	13	TSIZ1	14	A12	13	User Def.	14	CAN_RXD
15	D27	16	D11	15	TSIZ0	16	A11	15	User Def.	16	CAN_TXD
17	D26	18	D10	17	$\overline{\text{CS4}}$	18	A10	17	User Def.	18	$\overline{\text{RSTOUT}}$
19	D25	20	D9	19	$\overline{\text{CS3}}$	20	A9	19	User Def.	20	$\overline{\text{RESET}}$
21	D24	22	D8	21	$\overline{\text{CS2}}$	22	A8	21	User Def.	22	$\overline{\text{EB3}}$
23	D23	24	D7	23	$\overline{\text{CS1}}$	24	A7	23	User Def.	24	$\overline{\text{EB2}}$
25	D22	26	D6	25	A22	26	A6	25	User Def.	26	$\overline{\text{EB1}}$
27	D21	28	D5	27	A21	28	A5	27	User Def.	28	$\overline{\text{EB0}}$
29	D20	30	D4	29	A20	30	A4	29	User Def.	30	$\overline{\text{OE}}$
31	D19	32	D3	31	A19	32	A3	31	User Def.	32	PSTAT3
33	D18	34	D2	33	A18	34	A2	33	User Def.	34	PSTAT2
35	D17	36	D1	35	A17	36	A1	35	User Def.	36	PSTAT1
37	D16	38	D0	37	A16	38	A0	37	User Def.	38	PSTAT0
Logic Analyzer Pod 2		Logic Analyzer Pod 1		Logic Analyzer Pod 4		Logic Analyzer Pod 3		Logic Analyzer Pod 6		Logic Analyzer Pod 5	

Using Medium-Density Connectors

Medium-density connectors carry 16 signals plus one clock. These connectors are an older technology and are not recommended for system clock speeds above 50 Mhz.

- Each 16-signal medium-density header connector requires approximately 1.4” x 0.4” of printed-circuit board space.
- For each board connector, a 100 KOhm termination adapter (Agilent Part 01650-63203) is required. The termination adapter connects between the header connector and the 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, the general-purpose probes supplied with the logic analyzer can be used to route the signals to the correct pods.
- On-board termination is not required.
- 3M straight header part number: 2520-6002
- 3M right-angle header part number: 2520-5002

Below are the pinouts for the medium-density header connector:



Pin 20 must be connected to ground.

Application Note 1244-1, *Minimizing Intrusion Effects When Probing With a Logic Analyzer* describes this connector in more detail.

RIM Signal to Medium-Density Connector Pin Mapping

Pins 1 and 2 of the medium-density connectors (marked NC) must be a true no connect. The signals are used for other functions unavailable to target probing. They cannot be connected to anything on the target system, including ground.

Medium-Density Connector 1		Medium-Density Connector 2		Medium-Density Connector 3		Medium-Density Connector 4		Medium-Density Connector 5 (Optional)	
Pin	**M•CORE RIM Signal**	**Pin**	**M•CORE RIM Signal**	**Pin**	**M•CORE RIM Signal**	**Pin**	**M•CORE RIM Signal**	**Pin**	**M•CORE RIM Signal**
1	NC	1	NC	1	NC	1	NC	1	NC
2	NC	2	NC	2	NC	2	NC	2	NC
3	$\overline{\text{SHS}}$	3	CLKOUT	3	$\overline{\text{TA}}$	3	$\overline{\text{TEA}}$	3	User Def.
4	D31	4	D15	4	R/$\overline{\text{W}}$	4	A15	4	TC[2]
5	D30	5	D14	5	CSE1	5	A14	5	TC[1]
6	D29	6	D13	6	CSE0	6	A13	6	TC[0]
7	D28	7	D12	7	TSIZ1	7	A12	7	CAN_RXD
8	D27	8	D11	8	TSIZ0	8	A11	8	CAN_TXD
9	D26	9	D10	9	$\overline{\text{CS4}}$	9	A10	9	$\overline{\text{RSTOUT}}$
10	D25	10	D9	10	$\overline{\text{CS3}}$	10	A9	10	$\overline{\text{RESET}}$
11	D24	11	D8	11	$\overline{\text{CS2}}$	11	A8	11	$\overline{\text{EB3}}$
12	D23	12	D7	12	$\overline{\text{CS1}}$	12	A7	12	$\overline{\text{EB2}}$
13	D22	13	D6	13	A22	13	A6	13	$\overline{\text{EB1}}$
14	D21	14	D5	14	A21	14	A5	14	$\overline{\text{EB0}}$
15	D20	15	D4	15	A20	15	A4	15	$\overline{\text{OE}}$
16	D19	16	D3	16	A19	16	A3	16	PSTAT3
17	D18	17	D2	17	A18	17	A2	17	PSTAT2
18	D17	18	D1	18	A17	18	A1	18	PSTAT1
19	D16	19	D0	19	A16	19	A0	19	PSTAT0
20	GND	20	GND	20	GND	20	GND	20	GND
Logic Analyzer Pod 2		**Logic Analyzer Pod 1**		**Logic Analyzer Pod 4**		**Logic Analyzer Pod 3**		**Logic Analyzer Pod 5**	

Preparing for Emulation

To use the Agilent Technologies emulation module with your target system, your target system must have a JTAG (OnCE) port connector. The JTAG port is the communication port used to control the M•CORE embedded On-Chip Emulation (OnCE) system.

This section describes the target system requirements necessary for emulation via the JTAG (OnCE) port.

Required Signals for JTAG Emulation

Vdd. Vdd is used to sense when the target has power. The signal is also used as a reference voltage for all signals driven to the target by the emulation module. Note that at no time is more than 1mA drawn from the target system through the Vdd line.

$\overline{\text{RST}}$. This signal is required to reset the microprocessor.

TDI, TDO, $\overline{\text{TRST}}$, TCK, TMS. All of the JTAG signals are required to communicate with the OnCE port.

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

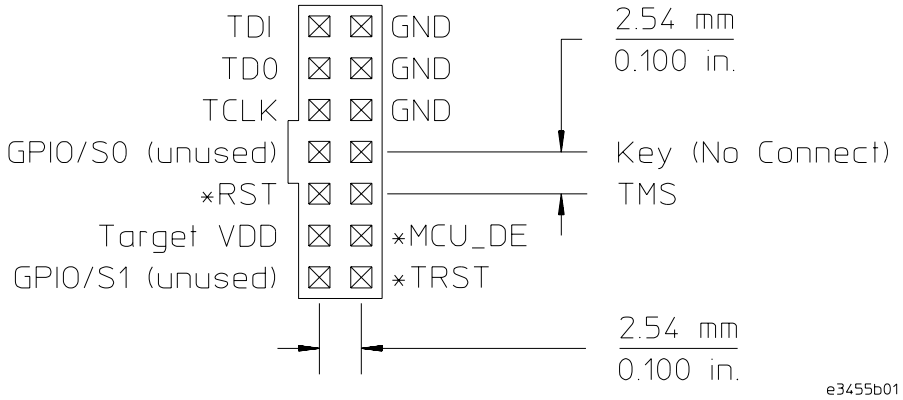
The TDI, TCK, TMS and $\overline{\text{TRST}}$ signals must not be actively driven by the target system when the debug port is being used.

Optional Signals for JTAG Emulation

$\overline{\text{MCU\_DE}}$. The $\overline{\text{MCU\_DE}}$ signal places the M•CORE into debug mode and is used by the emulation module after receiving a “BREAK IN” signal from the logic analyzer. This allows the logic analyzer triggering

capability to be used for complex breakpoints. Without $\overline{\text{MCU\_DE}}$, the processor must be stopped by using the JTAG scan chains, and the processor may not stop until several hundred instructions have been executed. This may cause complications when attempting to correlate the state of the logic analyzer and the emulation module after the breakpoint has been reached. This signal is optional but is strongly recommended.

JTAG Connector Mechanical and Pinout Information



NOTE:

$\overline{\text{MCU\_DE}}$, $\overline{\text{RST}}$, and all of the JTAG signals except TDO require a 10K pull-up resistor for deterministic operation in the absence of an emulation module.

The connector on the target system should be placed as close as possible to the processor to ensure signal integrity.

The connector on the target system should be placed as close as possible to the processor to ensure signal integrity.

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 16600A/16700A-series logic analysis systems

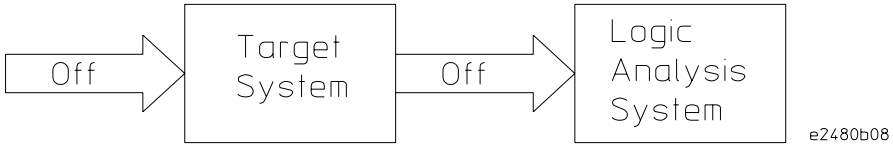
Ensure the target system is powered off.

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 Agilent Technologies 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:

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

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

To install in a 16700A-series logic analysis system

Or, to install in an Agilent Technologies 16701A expansion frame:

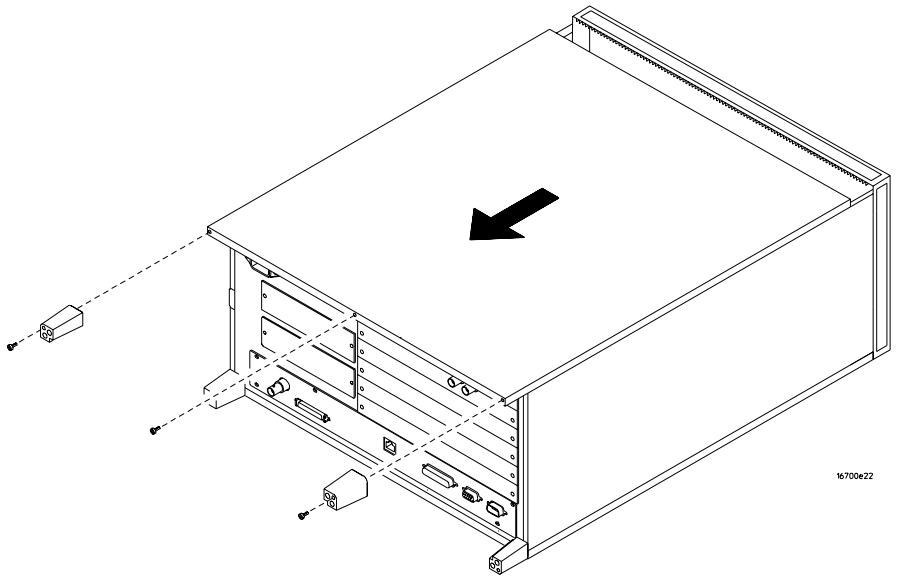
You will need T-10 and T-15 Torx screw drivers.

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

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

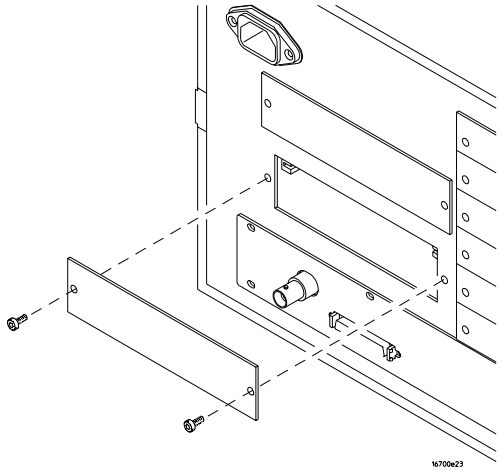
2. Turn the logic analysis system frame upside-down.

3. Remove the bottom cover.

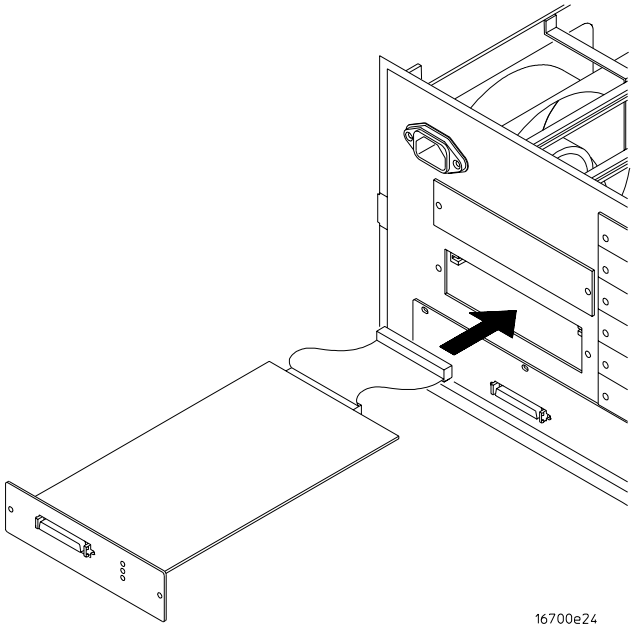


4 Remove the slot cover.

You may use either slot.

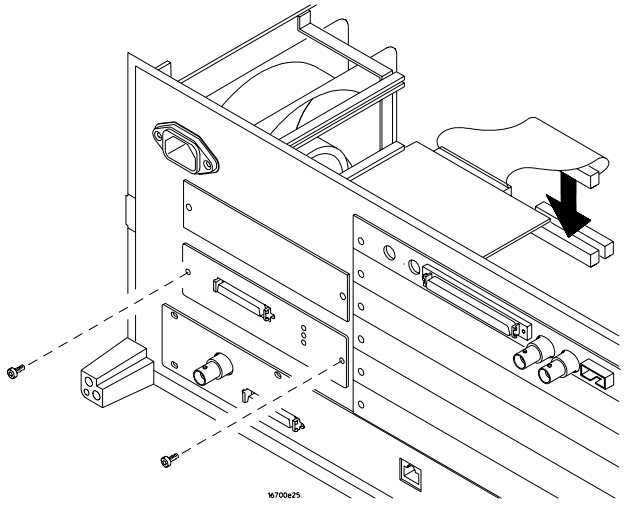


5 Install the emulation module.



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 69 for information on giving the emulation module a "personality" for your target processor.

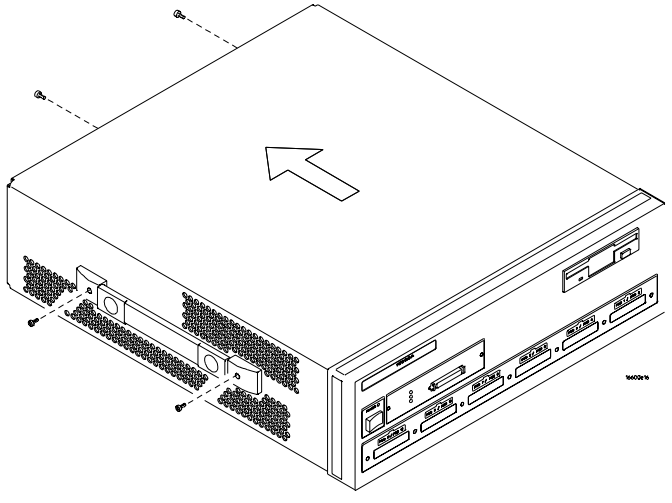
To install in a 16600A-series logic analysis system

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

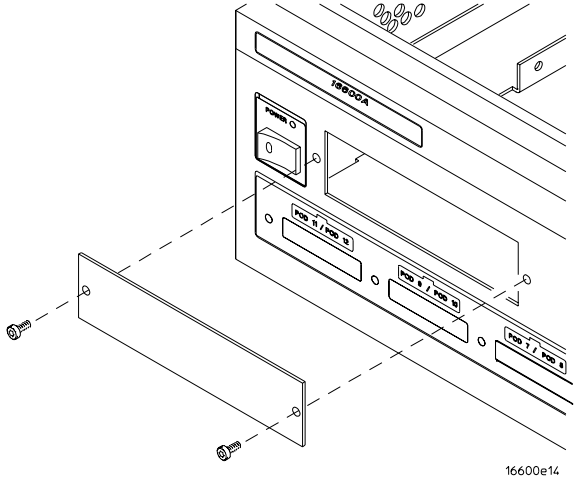
1 Turn off the logic analysis system and REMOVE THE POWER CORD.

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

2 Slide the cover back.

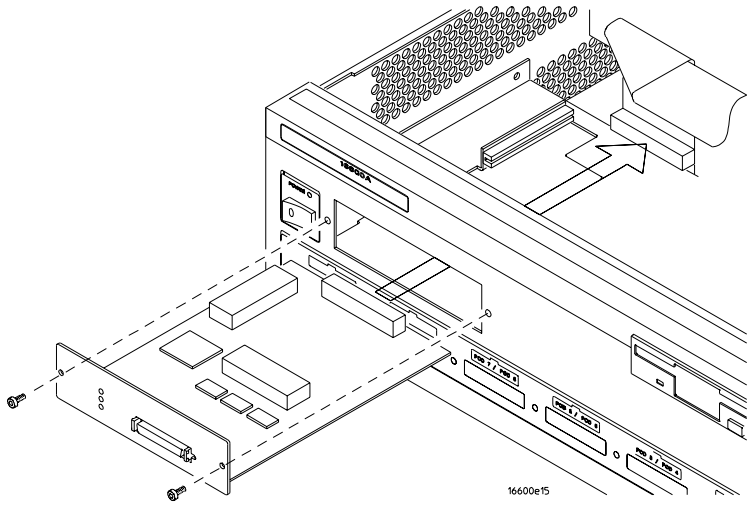


3 Remove the slot cover.



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 69 for information on giving the emulation module a "personality" for your target processor.

To test the emulation module

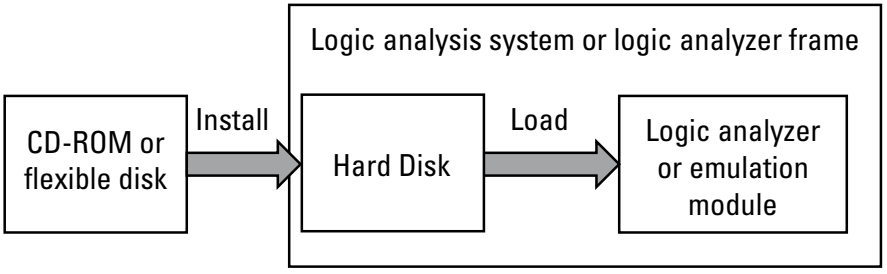
If this is the first time that you have used the emulation module, you should run the built-in performance verification tests before you connect to a target system. Refer to the "Troubleshooting the Emulation Module" chapter on page 169 for information on performance verification.

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 Agilent Technologies 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 Agilent Technologies 16600A/16700A-series logic analysis system's operating system, installation may take approximately 15 minutes.

1. Turn on the CD-ROM first, and then turn on the logic analysis system.
2. Insert the CD-ROM in the drive.
3. Click the System Admin icon.
4. Click Install... .

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

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

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

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

   The configuration files are stored in /logic/configs/hp/processor.

   The inverse assemblers are stored in /logic/ia.

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

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

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 Agilent Technologies 16600A and 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.

4

Probing the Target System

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.)

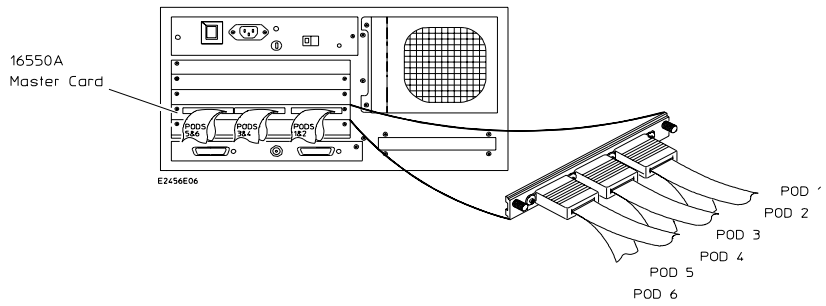
This section identifies connections to each Agilent Technologies logic analyzer supported by the inverse assembler. They are shown in the following order:

- 16550A logic analyzer.
- 16554/55/56/57 logic analyzers (one or two cards).
- 16600A logic analyzer.
- 16601A logic analyzer.
- 16602A logic analyzer.
- 16603A logic analyzer.
- 16710/11/12A logic analyzers.

Number of Pods Used/Required

The M•CORE inverse assembler requires a minimum of four pods. The logic analyzer configuration file for 32-bit memory controller systems assigns signals for five pods. If fewer than five pods are used, only configuration file definitions for the first four pods will be used.

To connect the 16550A logic analyzer



Use this table to connect cables from the Agilent Technologies 16550A logic analyzer to the connectors in the target system.

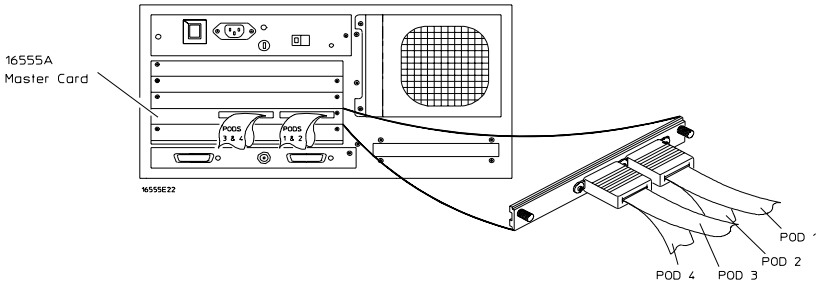
	Agilent Technologies 16550A Logic Analyzer Pod:					
	Pod 6	**Pod 5**	**Pod 4**	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:		3, Odd	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:		5	3	4	1	2

Configuration Files for the Agilent Technologies 16550A Logic Analyzer

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_4.

When the D[15:0] signals aren't connected to the logic analyzer, move the connector on pod 5 to pod 1 and use configuration file CMCRIM_2.

To connect a one-card 16554/55/56/57 logic analyzer



Use this table to connect cables from the one-card Agilent Technologies 16554/55/56/57 logic analyzer to the connectors in the target system.

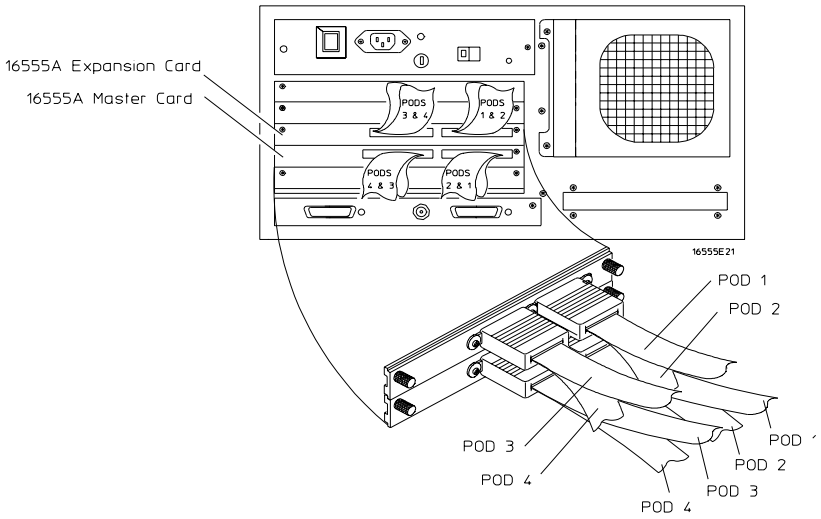
	Agilent Technologies 16554/55/56/57 Logic Analyzer Pod:			
	Pod 4	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:	3	4	1	2

Configuration Files for the One-Card Agilent Technologies 1655455/56/57 Logic Analyzers

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_3.

When the D[15:0] signals aren't connected to the logic analyzer, move the high-density connector "3, Odd" or medium-density connector "5" to pod 1 and use configuration file CMCRIM_1.

To connect a two-card 16554/55/56/57 logic analyzer



Use this table to connect cables from the two-card Agilent Technologies 16554/55/56/57 logic analyzer to the connectors in the target system.

	Agilent Technologies 16554/55/56/57 Logic Analyzer Expansion Card Pod:			
	Pod 4	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:				3, Odd
Medium-Density Connector:				5

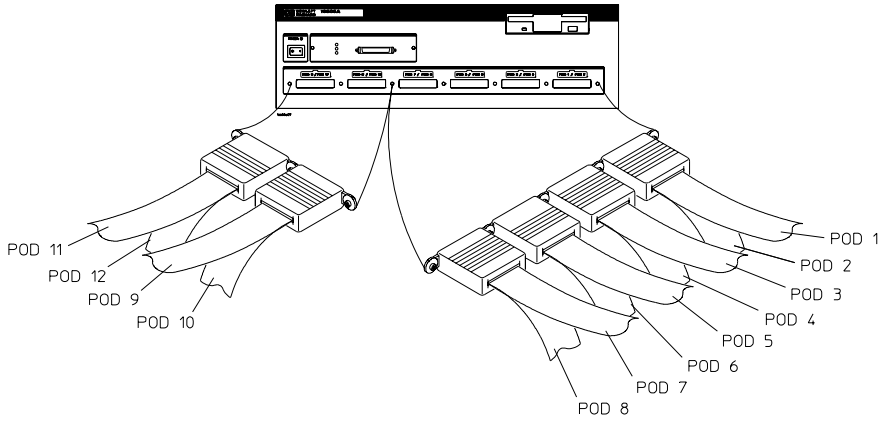
	Agilent Technologies 16554/55/56/57 Logic Analyzer Master Card Pod:			
	Pod 4	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:	3	4	1	2

Configuration Files for the Two-Card Agilent Technologies 1655455/56/57 Logic Analyzers

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_3.

When the D[15:0] signals aren't connected to the logic analyzer, move the connector on expansion card pod 1 to master card pod 1 and use configuration file CMCRIM_1.

To connect the 16600A logic analyzer



Use this table to connect cables from the Agilent Technologies 16600A logic analyzer to the connectors in the target system.

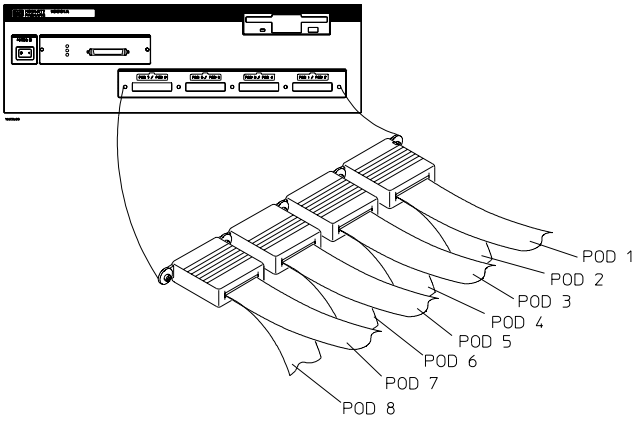
	Agilent Technologies 16600A Logic Analyzer Pod:						
	Pods 12-7	**Pod 6**	**Pod 5**	**Pod 4**	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:			3, Odd	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:			5	3	4	1	2

Configuration Files for the Agilent Technologies 16600A Logic Analyzer

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_3.

When the D[15:0] signals aren't connected to the logic analyzer, move the connector on pod 5 to pod 1 and use configuration file CMCRIM_1.

To connect the 16601A logic analyzer



Use this table to connect cables from the Agilent Technologies 16601A logic analyzer to the connectors in the target system.

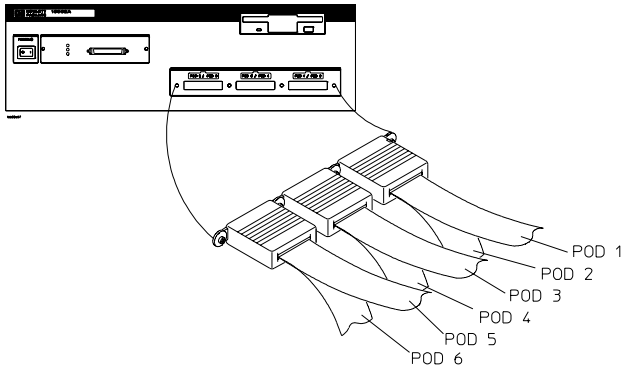
	Agilent Technologies 16601A Logic Analyzer Pod:						
	Pods 8-7	**Pod 6**	**Pod 5**	**Pod 4**	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:			3, Odd	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:			5	3	4	1	2

Configuration Files for the Agilent Technologies 16601A Logic Analyzer

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_3.

When the D[15:0] signals aren't connected to the logic analyzer, move the connector on pod 5 to pod 1 and use configuration file CMCRIM_1.

To connect the 16602A logic analyzer



Use this table to connect cables from the Agilent Technologies 16602A logic analyzer to the connectors in the target system.

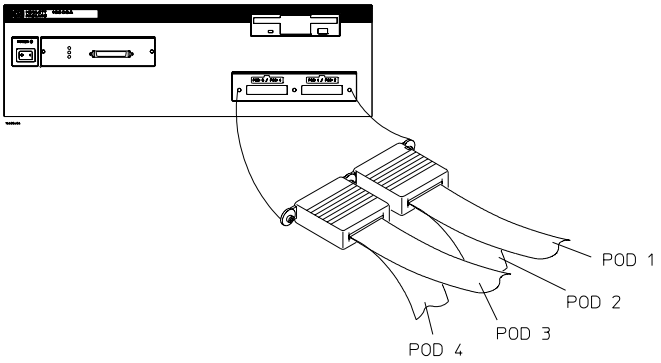
	Agilent Technologies 16602A Logic Analyzer Pod:					
	Pod 6	**Pod 5**	**Pod 4**	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:		3, Odd	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:		5	3	4	1	2

Configuration Files for the Agilent Technologies 16602A Logic Analyzer

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_3.

When the D[15:0] signals aren't connected to the logic analyzer, move the connector on pod 5 to pod 1 and use configuration file CMCRIM_1.

To connect the 16603A logic analyzer



Use this table to connect cables from the Agilent Technologies 16603A logic analyzer to the connectors in the target system.

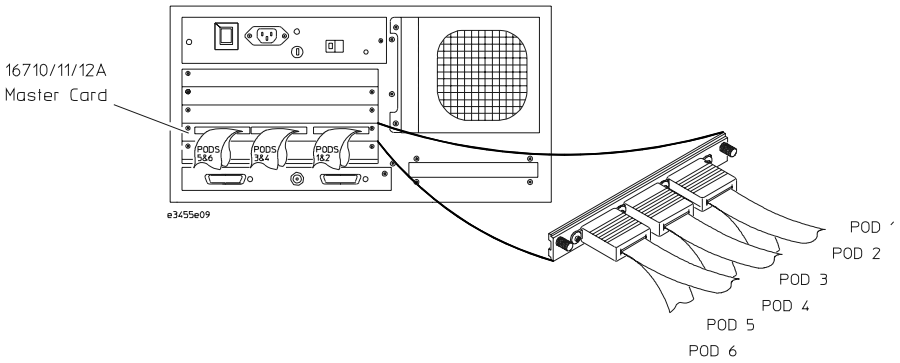
	Agilent Technologies 16603A Logic Analyzer Pod:			
	Pod 4	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:	3	4	1	2

Configuration Files for the Agilent Technologies 16603A Logic Analyzer

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_3.

When the D[15:0] signals aren't connected to the logic analyzer, move the high-density connector "3, Odd" or medium-density connector "5" to pod 1 and use configuration file CMCRIM_1.

To connect the 16710/11/12A logic analyzers



Use this table to connect cables from the Agilent Technologies 16710/11/12A logic analyzers to the connectors in the target system.

	Agilent Technologies 16550A Logic Analyzer Pod:					
	Pod 6	**Pod 5**	**Pod 4**	**Pod 3**	**Pod 2**	**Pod 1**
High-Density Connector:		3, Odd	2, Even	2, Odd	1, Even	1, Odd
Medium-Density Connector:		5	3	4	1	2

Configuration Files for the Agilent Technologies 16710/11/12A Logic Analyzers

When the optional D[15:0] signals are connected to the logic analyzer, use configuration file CMCRIM_3.

When the D[15:0] signals aren't connected to the logic analyzer, move the connector on pod 5 to pod 1 and use configuration file CMCRIM_1.

Connecting the Emulation Module to the Target System

This section shows you how to connect the emulation module to the JTAG (OnCE) connector in 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.

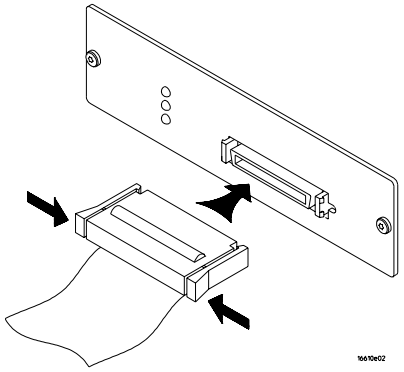
See Also

For information on designing a JTAG (OnCE) port on your target board, see “Preparing for Emulation” on page 38.

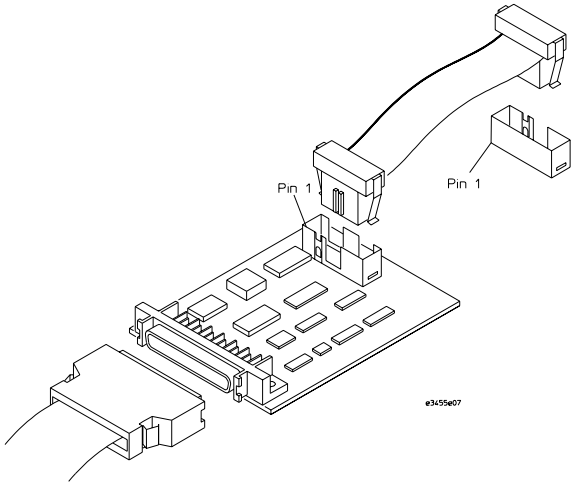
For a list of the parts supplied with the emulation module, see “Emulation Module” on page 20.

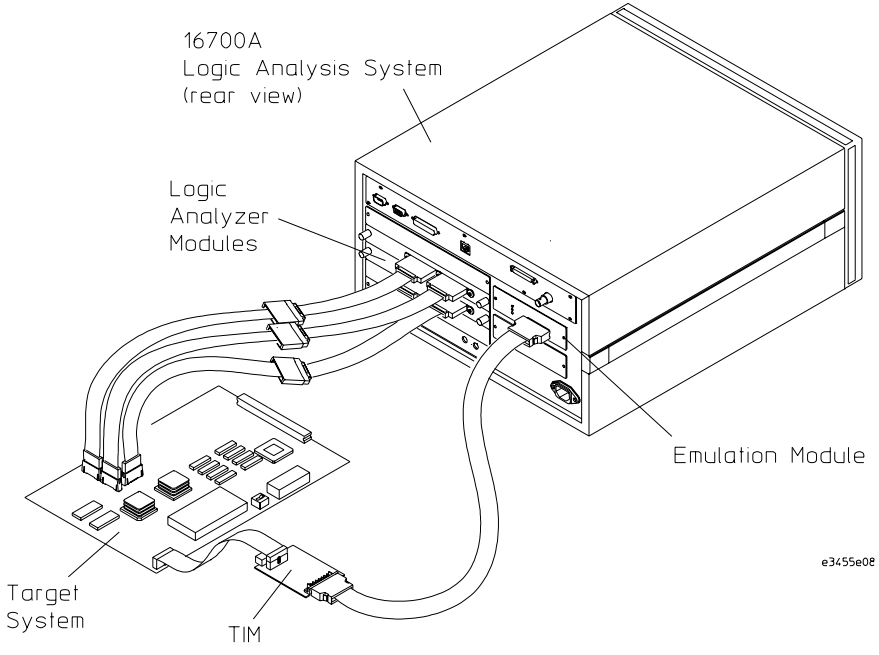
To connect to a target system JTAG (OnCE) port

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 14-pin cable into the target interface module.
6. Plug the other end of the 14-pin cable into the JTAG port OnCE connector on the target system.





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

See Also

For information on designing a target system for use with the emulation module, see “Preparing for Emulation” on page 38.

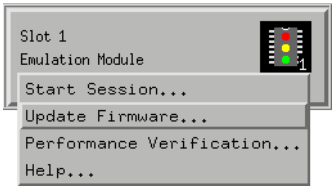
To update emulation module firmware

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 Agilent Technologies.

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 50.

To display the emulation module firmware version information

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

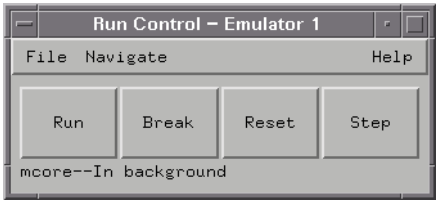
There are usually two firmware version numbers: one for "Generics" and one for the personality of your processor.

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 and whether your target system has a 16- or 32-bit data bus.

The M•CORE 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 /logic/configs/hp/mcore/ exists.

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

2 Using File Manager, select the configuration file you want to load in the /logic/configs/hp/mcore/ 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 M•CORE analysis by loading the appropriate M•CORE configuration file. Loading configuration files also

automatically loads the inverse assembler. The configuration file names are located at the bottom of the table showing the connections for your particular logic analyzer.

3 Close File Manager.

Logic Analyzer Configuration Files

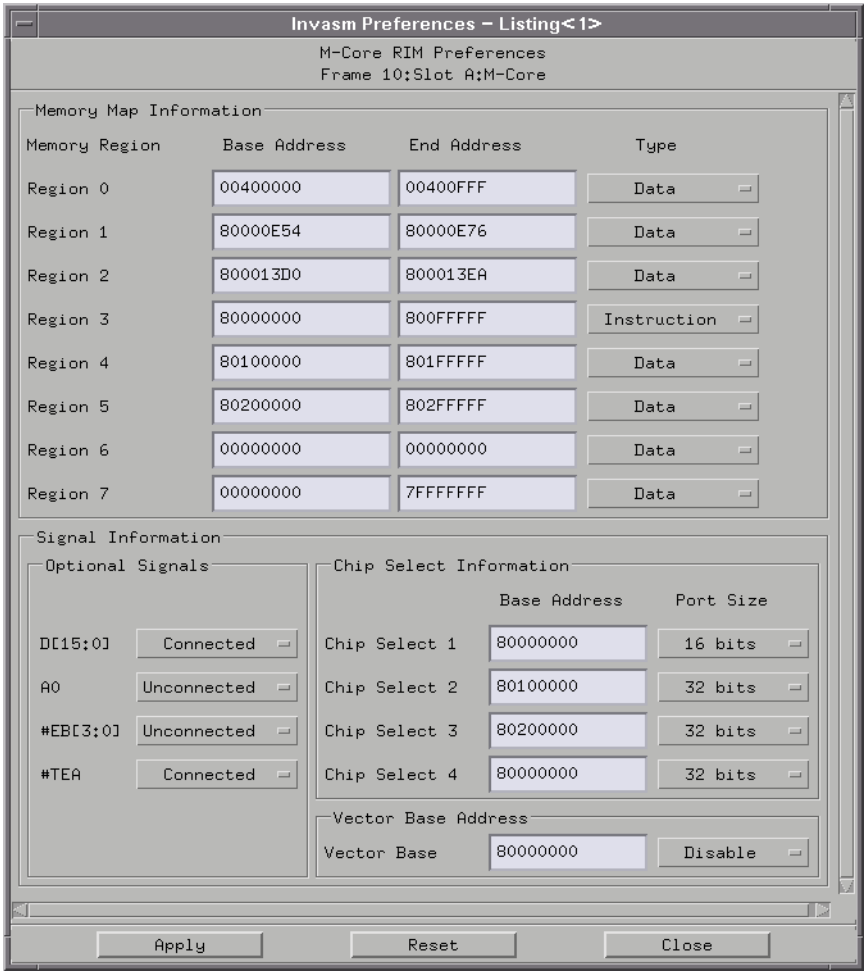
Agilent Technologies Analyzer Model	Analyzer Description (full/half channels)	Configuration File - D[15:0] Connected	Configuration File - D[15:0] Not Connected
16550A	100 MHz state 250/500 MHz timing 4K/8K samples	CMCRIM_4	CMCRIM_2
16554A (one or more cards)	70 MHz state 125/250 MHz timing 0.5M/1M samples	CMCRIM_3	CMCRIM_1
16555A/D (one or more cards)	110 MHz state 250/500 MHz timing 2M/4M samples (D)	CMCRIM_3	CMCRIM_1
16556A/D (one or more cards)	100 MHz state 200/400 MHz timing 2M/4M samples (D)	CMCRIM_3	CMCRIM_1
16557D (one or more cards)	135 MHz state 250/500 MHz timing 2M/4M samples	CMCRIM_3	CMCRIM_1
16600A	100 MHz state 125/250 MHz timing 64K/128K samples	CMCRIM_3	CMCRIM_1
16601A	100 MHz state 125/250 MHz timing 64K/128K samples	CMCRIM_3	CMCRIM_1
16602A	100 MHz state 125/250 MHz timing 64K/128K samples	CMCRIM_3	CMCRIM_1
16603A	100 MHz state 125/250 MHz timing 64K/128K samples	CMCRIM_3	CMCRIM_1

Logic Analyzer Configuration Files			
Agilent Technologies Analyzer Model	**Analyzer Description (full/half channels)**	**Configuration File - D[15:0] Connected**	**Configuration File - D[15:0] Not Connected**
16710A	100 MHz state 250/500 MHz timing 8K/16K samples	CMCRIM_3	CMCRIM_1
16712A	100 MHz state 250/500 MHz timing 128K/256K samples	CMCRIM_3	CMCRIM_1
16711A	100 MHz state 250/500 MHz timing 32K/64K samples	CMCRIM_3	CMCRIM_1

Configuration File Label Descriptions		
Label	**STAT Bits**	**Description**
ADDR		Address bus signals A0-A22.
CLKOUT	STAT[9]	Clock Out output signal.
#CS4-1	STAT[3:0]	Chip Select output signals.
CSE1-0	STAT[7:6]	Emulation Mode Chip Select output signals.
DATA		Data bus signals D0-D31 if 32-bit memory controller, D16-D31 if 16-bit memory controller.
#EB3-0	STAT_B[8:5]	Enable Byte output signals.
#OE	STAT_B[4]	Output Enable output signal.
R/#W	STAT[8]	Read/Write output signal.
#SHS	STAT[10]	Show Cycle Strobe output signal.
SIZ1-0	STAT[5:4]	Transfer Size output signals.
STA3-0	STAT_B[3:0]	PSTAT signals (information about the internal pipeline).
STAT		Microprocessor status and control signals.
STAT_B		Optional microprocessor status and control signals.
#TA	STAT[12]	Transfer Acknowledge signal.
TC2-0	STAT_B[15:13]	Transfer Codes output signals.
#TEA	STAT[11]	Transfer Error Acknowledge signal.

To set the inverse assembler preferences

The 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.



Memory Map Information

Lets you define up to 8 memory regions so that the inverse assembler can distinguish between instruction reads and data reads. (Also, in the Filter dialog, you can specify color coding for each memory region.)

The inverse assembler assumes all memory regions are valid, so lower-numbered regions take precedence and should be used before higher-numbered regions. Region 0 has the highest priority, and region 7 has the lowest priority.

Notice that the addresses in the preferences dialog are 32-bit values; these are the same as RLB addresses or the addresses that appear under the SW_ADDR label that is generated by the inverse assembler.

Base Address. Defines the starting address of the region.

End Address. Defines the ending address of the region.

Type. Specifies whether accesses in this region are instruction accesses or data accesses.

Signal Information

Lets you specify which optional signals are connected, the port size for each chip select, and the vector base address so that the inverse assembler can use this information when disassembling captured execution.

Optional Signals. The inverse assembler can use optional signals to provide further information. Your selections in this area tell the inverse assembler which of the optional signals have been connected.

D[15:0]. The inverse assembler does not require the lower 16-bits of data because the RIM memory controller can be configured for 16-bit memories. If these signals are present, the inverse assembler will analyze them and display the correct data. NOTE: Internal 32-bit accesses will not be supported if the lower data bits are not included.

A[0]. The least significant address pin is not required for inverse assembly. Some versions of the RIM memory controller do not bring A0 to an external pin. In this case, the byte enable signals EB[3:0] are required to properly decode 8-bit accesses to memory.

$\overline{\text{EB[3:0]}}$. If the A0 signal is connected, these signals are not used; however, if the A0 signal is not connected, these signals are used to decode 8-bit accesses.

$\overline{\text{TEA}}$. This signal is used to determine a failed bus cycle.

Chip Select Information. For each of the four chip selects, you can tell the inverse assembler the base address and port size.

Vector Base Address. If enabled, the inverse assembler displays the vector numbers and descriptions of the vectors in the table at the given address. Because the vector table can be moved, you can disable the display of this information.

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.

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

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

To view predefined symbols for the M•CORE

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 M•CORE inverse assembler contain predefined symbols for logic analyzer labels.

To display the predefined symbols for the M•CORE:

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 Description		
Label	**Encoding**	**Symbol**
#CS4-1	xxx0	CS1
	xx01	CS2
	x011	CS3
	0111	CS4
CSE1-0	00	EXT
	01	INT
	10	REG
	11	IMB3
#EB3-0	0011	EB2-EB3
	0111	EB3
	1011	EB2
	1100	EB0-EB1
	1101	EB1
	1110	EB0
	1111	NONE
R/#W	0	write
	1	read
SIZ1-0	00	4 bytes
	01	1 byte
	10	2 bytes
	11	-----
#TEA	0	ERROR
	1	----

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.

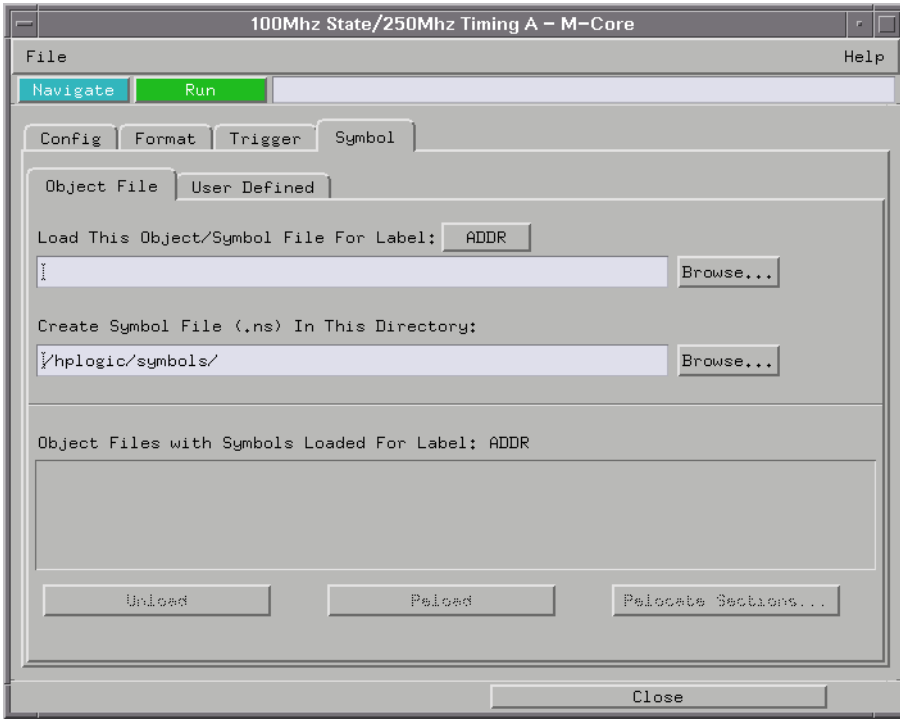
If your compiler generates object files in a format that the logic analyzer doesn't understand, you can use a General-Purpose ASCII (GPA) symbol file (see the "General-Purpose ASCII (GPA) Symbol File Format" chapter on page 191).

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

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

   Make sure the label is ADDR.

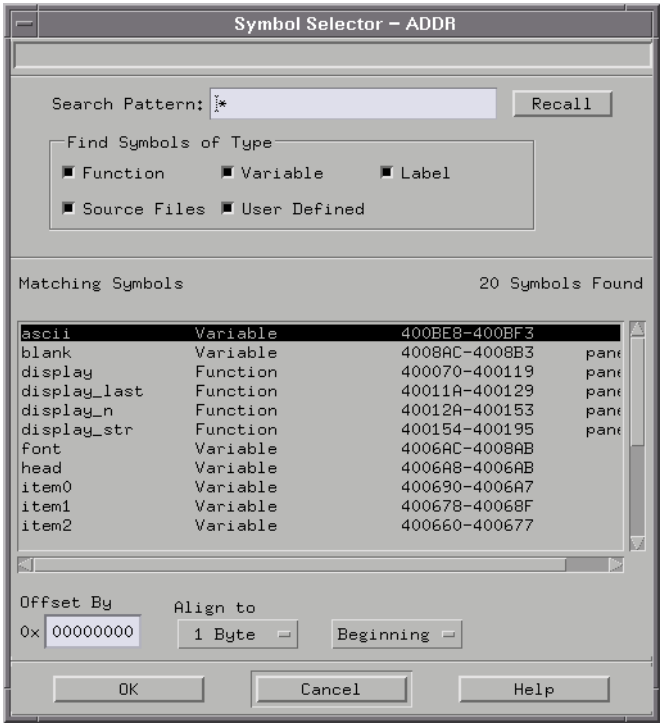
   From this dialog you can select object files and load their symbol information.



When you load object file symbols into a logic analyzer, a database 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.


Symbol Selector – ADDR
Search Pattern: *
Recall
Find Symbols of Type
Function
Variable
Label
Source Files
User Defined
Matching Symbols
20 Symbols Found
ascii Variable 400BE8-400BF3
blank Variable 4008AC-4008B3
display Function 400070-400119
display_last Function 40011A-400129
display_n Function 40012A-400153
display_str Function 400154-400195
font Variable 4006AC-4008AB
head Variable 4006A8-4006AB
item0 Variable 400690-4006A7
item1 Variable 400678-40068F
item2 Variable 400660-400677
Offset By
0x 00000000
Align to
1 Byte
Beginning
OK
Cancel
Help

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 modes.

To change to state analysis

In state mode, the logic analyzer uses the $\overline{\text{SHS}}$ signal from the M•CORE target system to capture data synchronously. This mode allows inverse assembly of M•CORE instructions and is the default mode set up by the configuration files.

To configure the logic analyzer for state mode:

1 Load the appropriate logic analyzer configuration file (see "Loading Configuration Files" on page 74).

The state configuration files set up the rising edge of the K clock (K↑) as the master clock signal and the falling edge of the K clock (K↓) as the slave 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.

The analysis mode is set in the Config tab of the logic analyzer setup window.

To configure the logic analyzer for timing analysis:

1. Load the appropriate logic analyzer configuration file (see "Loading Configuration Files" on page 74).
2. Open the logic analyzer's Setup window.
3. Select the Config tab.
4. Change the type option from State to Timing.

6

Capturing M•CORE 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 M•CORE 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 74).

You can load program object file symbols into the logic analyzer when configuring it (see "Loading Symbol Information" on page 80).

This chapter describes setting up logic analyzer triggers when using the Agilent Technologies E9612A Option 001 inverse assembler and the Agilent Technologies B4620B source correlation tool set.

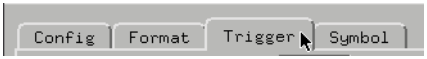
See the "Displaying Captured M•CORE Execution" chapter on page 95 for information on displaying captured data.

Setting Up Logic Analyzer Triggers

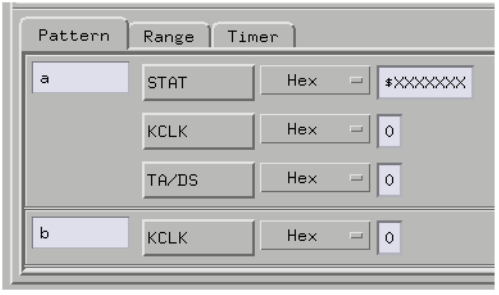
To set up logic analyzer triggers

1 Open the logic analyzer's Setup window.

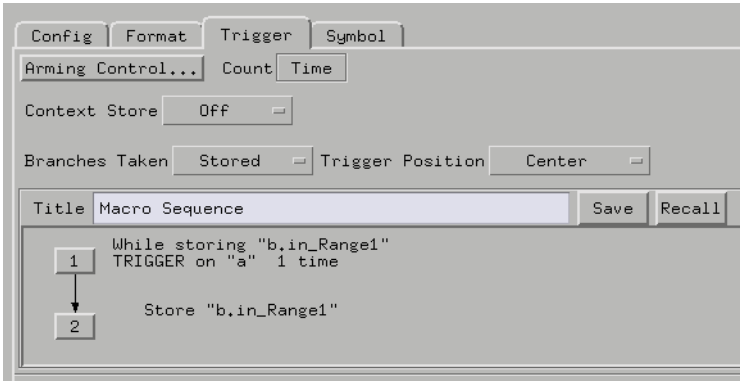
2 Select the Trigger tab.



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



4 Set up the trigger sequence.



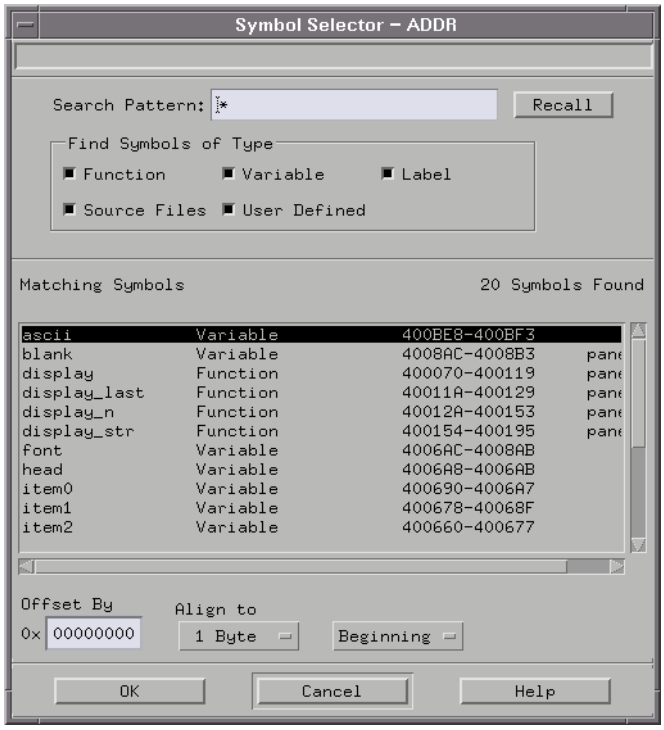
5 Run the measurement.

See Also

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

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.

Triggering on Source Code

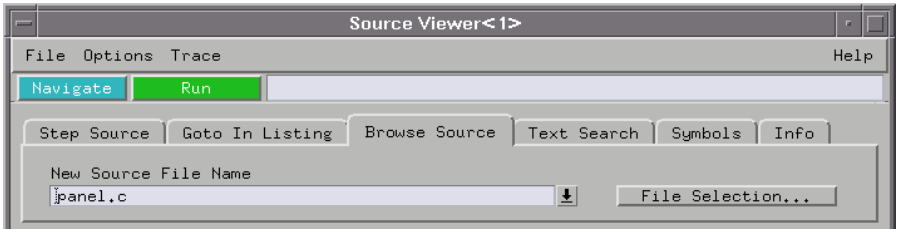
When setting up trigger specifications to capture M•CORE execution:

- 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).

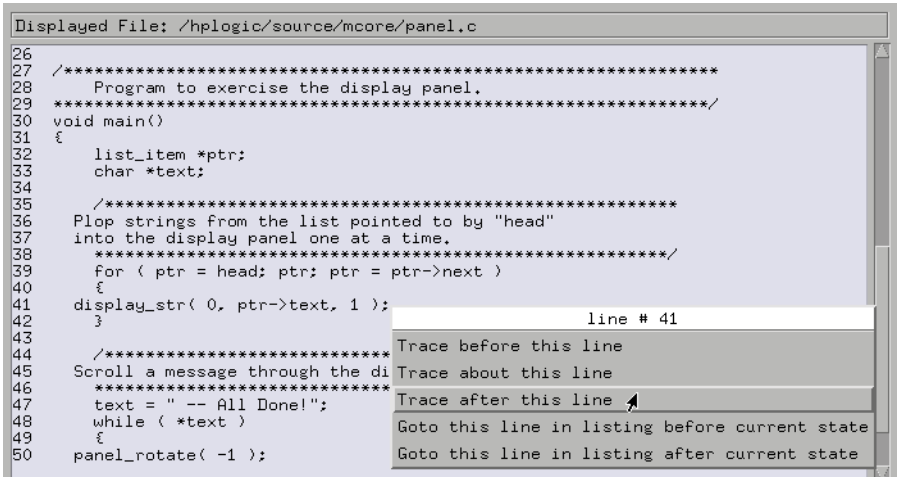
To set up triggers based on source code

1 Open the Source Viewer window.

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



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



4 Run the measurement.

To avoid capturing library code execution

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

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

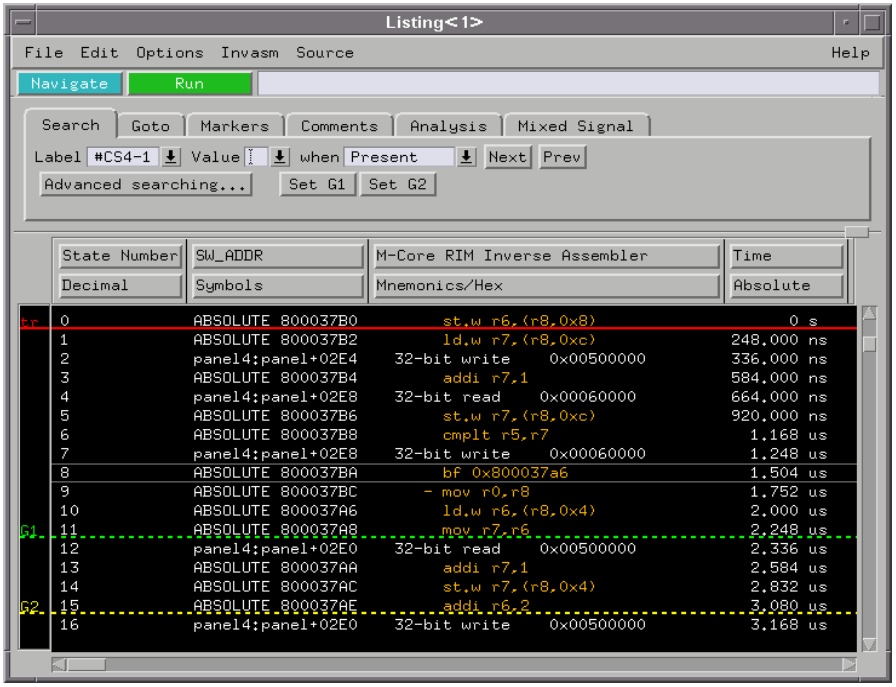
7

Displaying Captured M•CORE 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 IAMCRIME, and it is located in the /logic/ia directory.

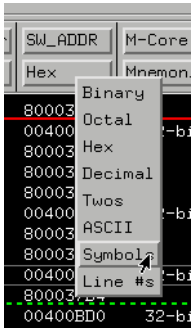
See Also

"To use the inverse assembler filters" on page 98 for information on displaying or hiding certain types of microprocessor bus cycles.

The Agilent Technologies 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 81.

To interpret the inverse assembled data

A column to the left of the M•CORE instruction in the Listing window can contain a symbol which provides further explanation of the state.

State Symbols	
+	Executed instruction due to condition code passed.
-	Unexecuted instruction due to condition code failed.
?	When a branch is around one instruction and the inverse assembler cannot tell whether the instruction was executed.

The inverse assembler may also display the following messages:

"Address Not In The Memory Map!"

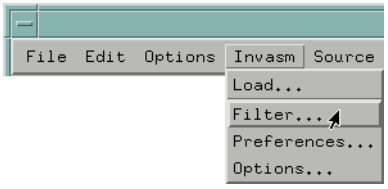
This message appears when there is no memory region set up for a captured address value (see "To set the inverse assembler preferences" on page 77).

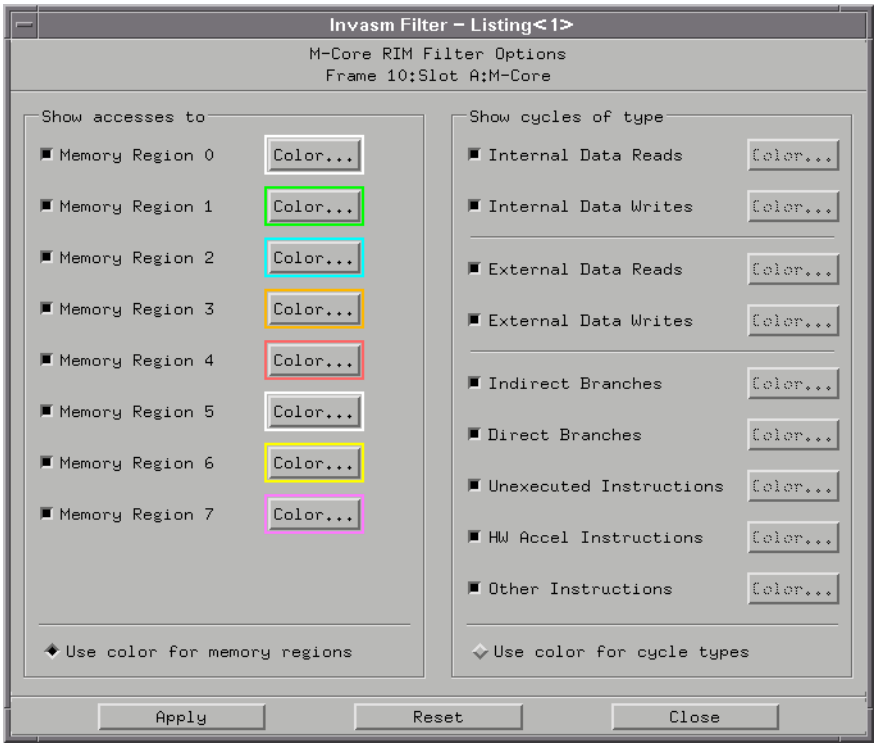
"Data Retrieval Error"

This message appears when the $\overline{\text{TEA}}$ signal is asserted (0).

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 region 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 unexecuted instructions will leave only instructions that were executed.
- 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 region accesses (when they are displayed). Color can be used for

distinguishing between memory region accesses or cycle types, but not both at the same time.

You can display or hide the following types of cycles:

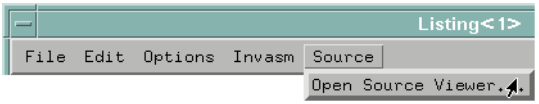
- Internal Data Reads.
- Internal Data Writes.
- External Data Reads.
- External Data Writes.
- Indirect Branches.
- Direct Branches.
- Unexecuted Instructions.
- HW Accel Instructions.
- Other Instructions.

See Also

The address ranges for memory regions 0-7 are specified in the Preferences menu (see "To set the inverse assembler preferences" on page 77).

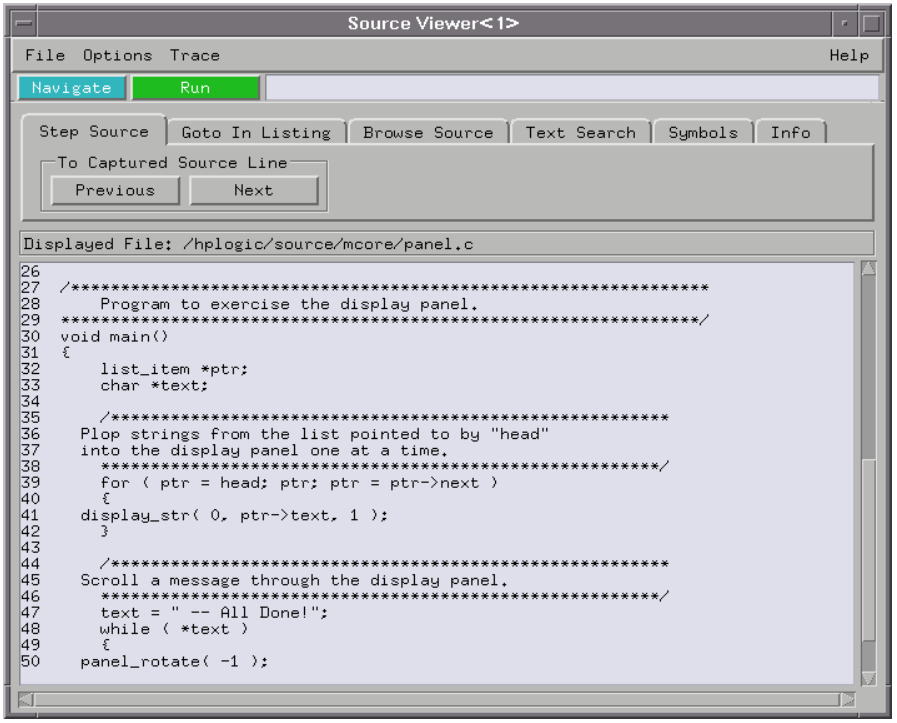
To view the source code associated with captured data

- 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 81).



Inverse Assembler Generated SW_ADDR (Software Address) Label

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

The “Goto this line in listing” commands in the Agilent Technologies 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 Agilent Technologies B4620B source correlation tool set can often read and access the correct source code file from information contained in the symbol file if the source code files have not been moved since they were compiled.

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

Source File Version Control. If the source code files are under a source code or version control utility, check the file names and paths carefully. These utilities can change source code file paths and file names. If these names are changed from the information contained in the symbol file, the source correlation tool set will not be able to find the proper source code file. These version control utilities usually

provide an “export” command that creates a set of source code files with unmodified names. The source correlation tool set can then be given the correct path to these files.

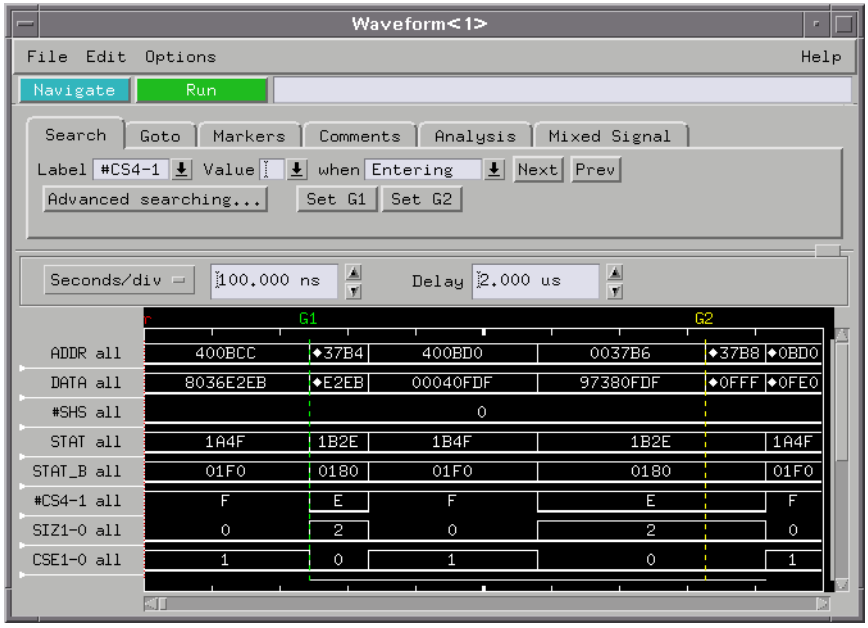
See Also

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

To display captured timing analysis mode data

- Open the Waveform display for your logic analyzer.

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


Waveform<1>
File Edit Options
Help
Navigate
Run
Search
Goto
Markers
Comments
Analysis
Mixed Signal
Label #CS4-1
Value
when Entering
Next
Prev
Advanced searching...
Set G1
Set G2
Seconds/div
100.000 ns
Delay
2.000 us
G1
G2
ADDR all
400BCC
◆37B4
400BD0
0037B6
◆37B8
◆0BD0
DATA all
8036E2EB
◆E2EB
00040FDF
97380FDF
◆0FFF
◆0FE0
#SHS all
0
STAT all
1A4F
1B2E
1B4F
1B2E
1A4F
STAT_B all
01F0
0180
01F0
0180
01F0
#CS4-1 all
F
E
F
E
F
SIZ1-0 all
0
2
0
2
0
CSE1-0 all
1
0
1
0
1

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 Agilent Technologies service center.

CAUTION: When you are working with the analyzer, be sure to power down both the analyzer and the target system before disconnecting or connecting cables, 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 112 for information on other sources of intermittent data errors.

Unwanted triggers

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

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

  The logic analyzer captures prefetches, even if they are not executed. When you are specifying a trigger condition or a storage qualification that follows an

instruction that may cause branching, an unused prefetch may generate an unwanted trigger.

No activity on activity indicators

- ❏ Check for loose cables or board connections.
- ❏ Check for bent or damaged pins.

No trace list display

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

- ❏ Check your trigger sequencer specification to ensure that it will capture the events of interest.
- ❏ Try stopping the analyzer; if the trace list is partially filled, this should display the contents of trace memory.

Analyzer won't power up

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

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

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 Agilent Technologies Sales Office if you need further assistance.

Target system will not boot up

If the target system will not boot up after connecting the 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 112. 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 56 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 the "Configuring the Logic Analyzer" chapter on page 73 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 may be incorrect because a portion of the execution trace was not visible to the logic analyzer.

❏ 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 the "Configuring the Logic Analyzer" chapter on page 73 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.

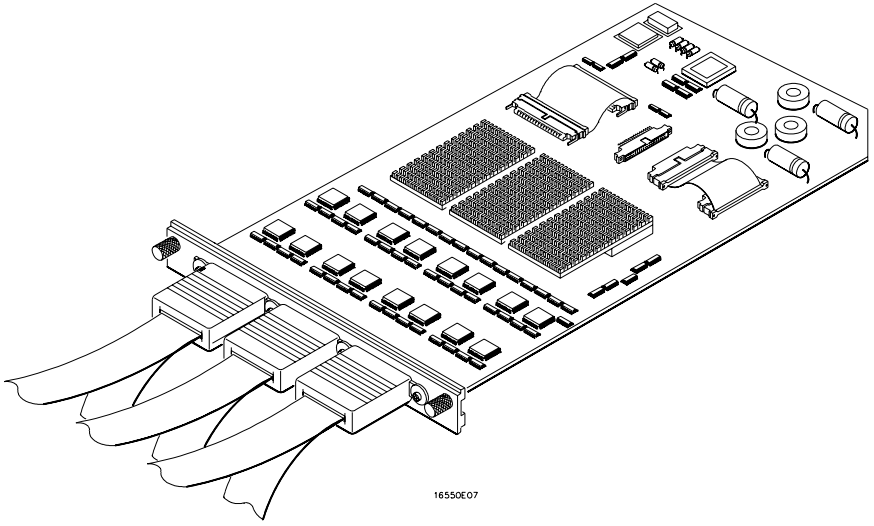
Ensure that the inverse assembler file is not renamed or deleted, and that it is located in the correct directory:

- For Agilent Technologies 16600A/16700A-series logic analysis systems it should be in /logic/ia.
- For other logic analyzers it should be in the same directory as the configuration file.

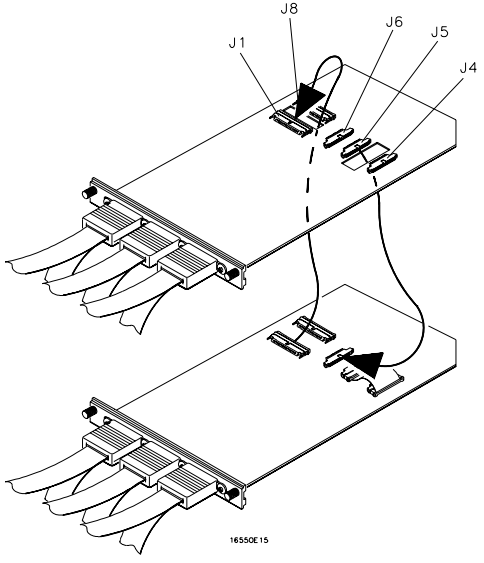
"Measurement Initialization Error"

This error occurs when you have installed the cables incorrectly for one or two Agilent Technologies 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 Agilent Technologies 16550A Installations



Cable Connections for Two-Card Agilent Technologies 16550A Installations



See Also

The *Agilent Technologies 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 the "Configuring the Logic Analyzer" chapter on page 73 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 Agilent Technologies 16600A/16700A-series logic analysis system frame or in the Agilent Technologies 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 56 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 Agilent Technologies 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.

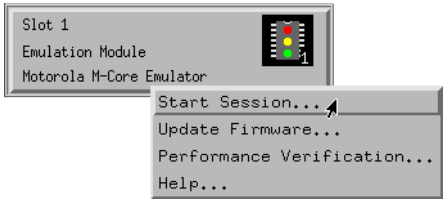
- Most debuggers cannot tolerate another interface issuing commands and may not start properly if another interface is running.

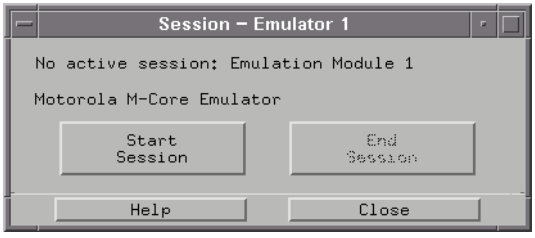
See Also

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

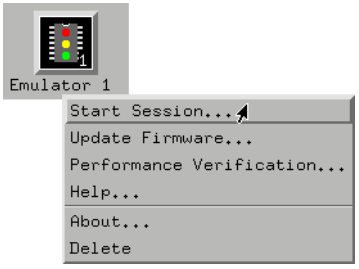
1. In the System window, click the emulation module icon.
2. Select Start Session....

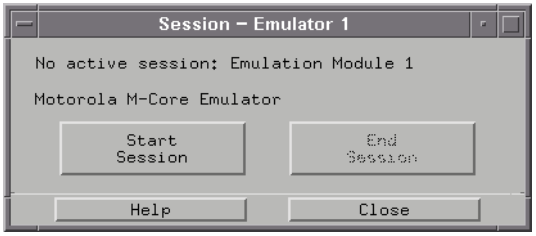




To start from the Workspace window

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





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 Agilent Technologies 16600A or 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 Agilent Technologies emulation module.

To use the Emulation Control Interface

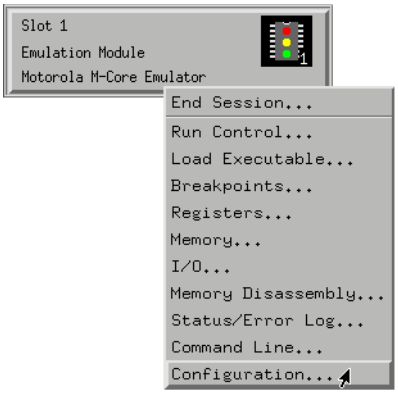
The easiest way to configure the emulation module is to use the Emulation Control Interface.

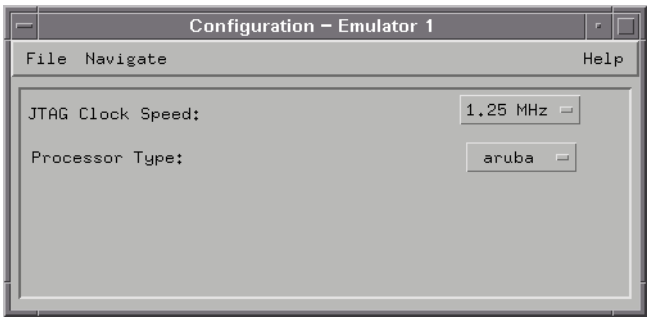
1 Start an Emulation Control Interface session.

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 information on each of the configuration options.

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

To use the built-in command interface

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

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

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

   ```
   telnet LAN_address 6472
   ```

2. Enter cf to see the current configuration settings.

3. Use the cf command to change the configuration settings.

See Also

Enter help cf for help on the configuration commands.

For information on connecting using telnet, and for information on other built-in commands, see "Built-In Commands" on page 173.

Example

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

```
cf - display or set emulation configuration

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

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

--- VALID CONFIGURATION <item> NAMES ---

  proc     - Set type of target processor
  breakin  - BNC break in control
  rrt      - Set restriction to real time runs
  speed    - Set JTAG Clock Divisor
  trigout  - Trigger out control
M>
```

To use a debugger interface

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

See Also

Information about specific debuggers in the “Using Debuggers (with the Emulation Module)” chapter on page 139.

Your debugger manual.

Setting the M•CORE Configuration Options

You must configure the emulation module to work with your target system.

The following options can be configured using the Emulation Control Interface or using built-in commands:

- Processor type.
- Processor clock speed.
- "Break In" type.

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

- Restriction to real-time runs.

To configure the processor type

Processor type configuration		
Value	**Emulation module configured for**	**Built-in command**
mcore	M•CORE (Default Revision 1.5 Core)	cf proc = mcore
MMC2001	MMC2001	cf proc = MMC2001
aruba	RIM-based devices	cf proc = aruba

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 the JTAG clock speed

The Agilent Technologies emulator needs to be configured to communicate at a rate which is compatible with your target processor. The JTAG Clock speed is independent of processor clock speed. In general, cf speed=7 can always be used and provides the best performance. With some target systems that have additional loads on the JTAG lines or with target systems that do not quite meet the requirements described in the "Preparing the Target System" chapter on page 25, setting speed to a slower setting may enable the probe to work.

Processor clock speed configuration

Value	TCK is at least	Built-in command
10 MHz	10 MHz (default)	cf speed = 7
5 MHz	5 MHz	cf speed = 6
2.5 MHz	2.5 MHz	cf speed = 5
1.25 MHz	1.25 MHz	cf speed = 4
625 KHz5	625 KHz	cf speed = 3
312 KHz	312 KHz	cf speed = 2
156 KHz	156 KHz	cf speed = 1
78 KHz	78 KHz	cf speed = 0

To configure the "Break In" type

This option affects how the emulation module will react to a trigger in an intermodule measurement.

"Break In" type configuration	
Value	**What happens when the emulation module is triggered**
Maskable	A trigger will immediately cause a maskable break. If the maskable break fails, a non-maskable break will be attempted. The delay between an attempted maskable break and the non-maskable break will allow many instructions to be executed. (Default)
NonMaskable	A trigger will immediately cause a non-maskable break. Use this value if you are trying to halt the processor in an interrupt service routine. The processor may not be able to continue running after the break.

To configure restriction to real-time runs

Real-time runs configuration		
Value	**Emulation module configured for**	**Built-in command**
no	Allows commands which temporarily break to the monitor. Examples include commands which display memory or registers.	cf rrt = no
yes	No commands are allowed which break to the monitor, except "break," "reset," "run," or "step." The processor must be explicitly stopped before these commands can be performed. (Default)	cf rrt = yes

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

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 the "Troubleshooting the Emulation Module" chapter on page 169 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 Debuggers (with the Emulation Module)

Several companies sell source debuggers which work with the Agilent Technologies emulation module.

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 Agilent Technologies emulation probe (which is also known as a "processor probe" or "software probe") or an Agilent Technologies emulation module.

- A LAN connection between the PC or workstation that is running the debugger and the emulation probe or the Agilent Technologies 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.

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 44).
2. Connect the emulation module to your target system (see "Connecting the Emulation Module to the Target System" on page 66).
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 143).
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 145).
5. Configure the emulation module (see the "Configuring the Emulation Module" chapter on page 127).

   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.

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.

6. Begin using your debugger.

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 Agilent Technologies 16600A/700A-series logic analysis system)
6474	Slot 2 (Second emulation module in an Agilent Technologies 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 Agilent Technologies 16600A/700A-series logic analysis system)
6476	Slot 2 (Second emulation module in an Agilent Technologies 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 Agilent Technologies 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 “logic”.

   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 “logic”.

   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.

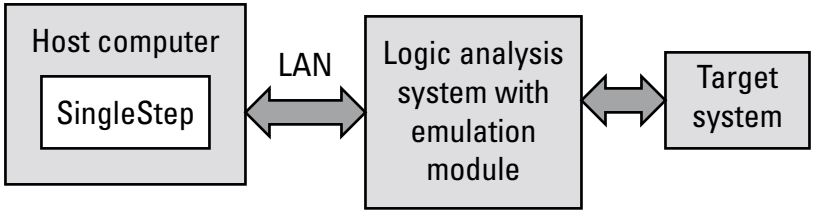
Using the Software Development Systems Debugger

Compatibility

Version 7.3 of the SingleStep debugger from Software Development Systems, Inc. is one debugger that connects to the Agilent Technologies 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 and programmed with the register values in the configuration file (<filename>.cfg).
- Hardware breakpoints are disabled.
- Software breakpoints are enabled.
- All breakpoints are cleared.
- main() _exit breakpoints are set, if that option is selected.

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 Agilent Technologies 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 144 for more information on port numbers.

2 Configure the emulation module with the processor clock speed.

   In the Debug dialog box, click the Connection tab and then enter the JTAG clock speed. Select 10 MHz for processors running at greater than 25 MHz.

3 Initialize the target system.

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

   SingleStep provides a way for target initialization to occur without running application code through the use of the "_config" alias. _config is used to define a list of commands that will be used to initialize the target after a reset. The _config alias should be defined in the sstep.ini file (in the "cmd" directory) and will point to a file of type .cfg which contains the actual initialization commands.

   An alternate way of creating the _config alias is to use the Target

Configuration tab in the “Debug” dialog box.

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

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

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

5 Download the application and run:

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

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

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

Examples

To see the speed that the emulation module is using to communicate with the target system you would issue the following command in the SingleStep command window:

```
control -c "cf speed"
```

To change the speed to match a 25 MHz processor clock you would issue the following command in the command window:

```
control -c "cf speed=7"
```

For more information about "terminal interface" commands see "Built-In Commands" on page 173.

On-chip breakpoints and debugging ROM code

The M•CORE has a built-in hardware breakpoint capability. When SingleStep steps one source line or sets a user defined breakpoint, it will first try to use a software breakpoint. If the breakpoint does not work because the breakpoint address is located in ROM, SingleStep will automatically attempt to use one of the available hardware breakpoints. For more information, see the SingleStep release notes.

To debug ROM based code, unselect “load application image” in the options tab of the “Debug” dialog.

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 M•CORE 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 M•CORE:

   See “To display the emulation module firmware version information” on page 70. If the emulation module is not programmed with the proper firmware, see “To update emulation module firmware” on page 69.

“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 144.

"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 logic analysis system.

  If you are unable to ping the logic analysis system, refer to "Solving LAN Communication Problems" on page 180 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 the "Configuring the Emulation Module" chapter on page 127 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 Agilent Technologies 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 Agilent Technologies B4620B source correlation tool set, to relate the analysis trace to your high-level source code.

Which assembly-level listing should I use?

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

- The Listing tool shows processor states that were captured during a "Run" of the logic analyzer. Those states are disassembled and displayed in the Listing window.
- The Emulation Control Interface shows the disassembled contents of a section of memory in the Memory Disassembly window.
- Your debugger shows your program as it was actually assembled, and (if it supports the emulation 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 Agilent Technologies B4620B source correlation tool set.
- Your debugger shows which line of code corresponds to the current value of the program counter on your target system. Use your debugger to set breakpoints.

Where can I find practical examples of measurements?

The Measurement Examples section in the 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

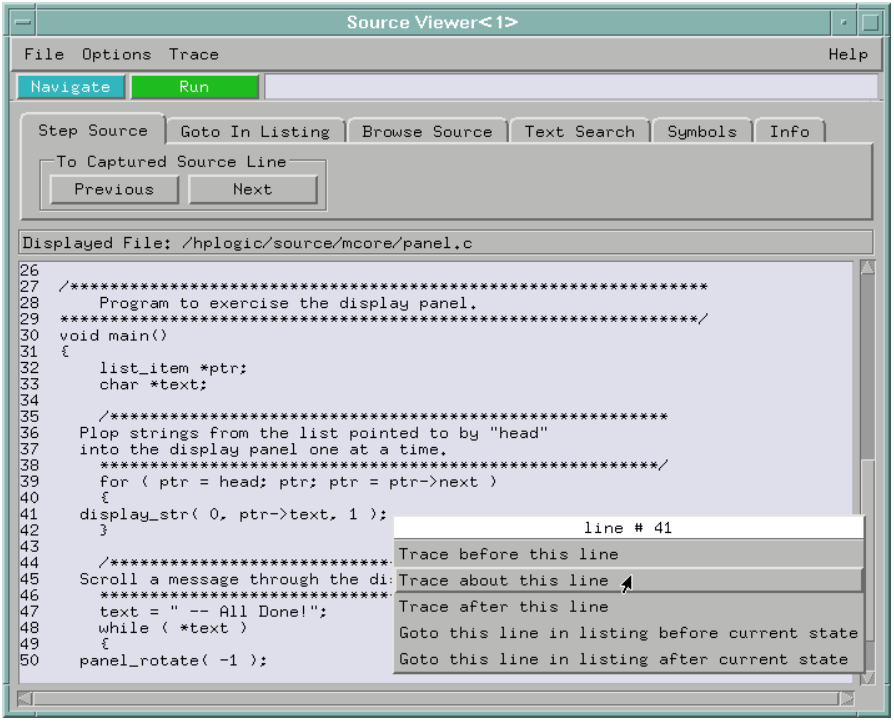
You can trigger the emulation module from the logic analyzer using either the Source Viewer window or the Intermodule window. If you are using the Agilent Technologies B4620B source correlation tool set, using the Source Viewer window is the easiest method.

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

If you have the Agilent Technologies 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.

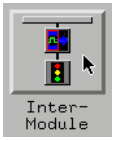
3 Click Run in the Source window (or other logic analyzer window).

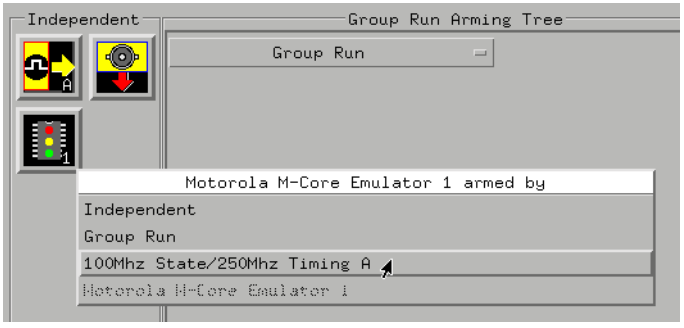
4 If your target system is not already running, click Run in the emulation Run Control window to start your target.

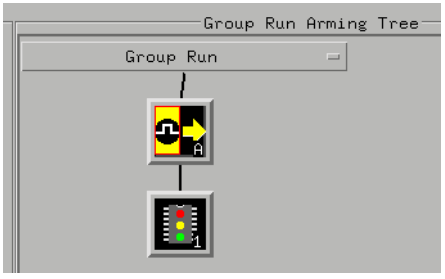
To stop on any trigger (Intermodule window)

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

1. Create a logic analyzer trigger.
2. 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 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 the JTAG clock speed" on page 134.

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.

Some possible uses for this measurement are:

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

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

To capture a trace before the processor halts

1. Set the logic analyzer to trigger on nostate.
2. Set the trigger point (position) to End.
3. In a logic analyzer window, 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 161).

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.

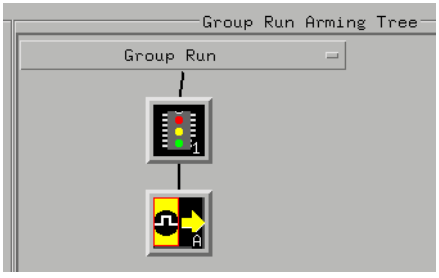
Here are some examples:

- 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 161).

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 161).

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 173
Emulation module built-in commands do not work	1 Check that the emulation module has been properly configured for your target system.	page 127
	2 Run the emulation module performance verification tests.	page 182
	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 "Preparing the Target System."	page 25, page 176
"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 134
Some commands fail	Check the "restrict to real-time runs" configuration.	page 135

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)

Built-In Commands

The emulation module has some built-in "terminal interface" commands which you can use for troubleshooting.

You can access the terminal interface using:

- A telnet (LAN) connection.
- The Command Line window in the Emulation Control Interface.
- A "debugger command" window in your debugger.

To telnet to the emulation module

You can establish a telnet connection to the emulation module if:

- A host computer and the logic analysis system are both connected to a local-area network (LAN).
- And, the host computer has the telnet program (often part of the operating system or an internet software package).

To establish a telnet connection:

1. Find out the port number of the emulation module.

   The default port number of the first emulation module in an Agilent Technologies 16600A/700A-series logic analysis system is 6472. The default port of a second module in an Agilent Technologies 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
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
r	Target reset
?	Unknown state

Examples

To set register R0 and view R0 to verify that it was set, enter:

```
R>rst -m
M>reg r0=ffff
M>reg r0
  reg R0=0000ffff
```

To break execution then step a single instruction, enter:

```
M>b
M>s
  PC= xxxxxxxx
M>
```

To determine what firmware version is installed in the emulation module, enter:

```
M>ver
```

See Also

Use the help command for more information on these and other commands. Note that some of commands listed in the help screens are generic commands for Agilent Technologies emulators and may not be available for your product.

If you are writing your own debugger, contact Agilent Technologies for more information.

Solving Target System Problems

This section describes how to determine whether your target system is causing problems with the operation of the emulation module.

What to check first

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

```
U>rst
R>
```

This should reset the target and display an “R>” prompt.

```
R>b
M>
```

This should stop the target and display an “M>” prompt.

```
M>reg r1
  reg r1=00000000
M>
```

This should read the value of the GPR1 register (the value will probably be different on your target system).

```
M>m 0..
  00000000 7c3043a6 7c2802a6 7c3143a6 4bf04111
  00000010 00000000 00000000 00000000 00000000
  00000020 00000000 00000000 00000000 00000000
  00000030 00000000 00000000 00000000 00000000
  00000040 00000000 00000000 00000000 00000000
  00000050 00000000 00000000 00000000 00000000
  00000060 00000000 00000000 00000000 00000000
  00000070 00000000 00000000 00000000 00000000
M>
```

This should display memory values starting at address 0.

```
M>s
```

This should execute one instruction at the current program counter.

If any of these commands don’t work, there may be a problem with the

design of your target system, a problem with the revision of the processor you are using, or a problem with the configuration of the emulation module.

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

```
M>ver
```

See Also "Built-In Commands" on page 173 for information on entering built-in commands.

To interpret the initial prompt

The initial prompt can be used to diagnose several common problems. To get the most information from the prompt, follow this procedure:

1 Connect the emulation module to your target system.

2 Set the default configuration settings. Enter:

```
M>init -c
```

You can enter this command at any prompt. The emulation module will respond with the same information as printed by the "ver" command.

If the response is "!ERROR 905! Driver firmware is incompatible with ID of attached device"	Make sure the target interface module is connected to the cable of the emulation module, then try the "init -c" command again.
If the initial prompt is "p>"	Check pin 6 on header, 3.3V (V_{DD}).
If the initial prompt is "M>"	The processor entered debug mode without the help of the emulation module. Is another debugger connected?
If the initial prompt is "U>"	The emulation module is scanning the instruction register correctly.

Now you can do some more tests:

3 Enter the reset command:

```
U>rst
R>
```

The "R>" prompt is a good response that indicates $\overline{\text{RESET}}$ is working.

If the prompt after rst is "U>", the $\overline{\text{RESET}}$ lines are not working. Continue with more tests:

4 Enter the break command:

```
U>b
M>
```

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:

```
M>m -d4 -a4 200000=2002,1200,1200,F7FC
M>reg r2=0
M>
```

This means: Add 1, GPR2, NOP, NOP, JMP .-4

Set the PC to this program:

```
M>reg PC=200000
M>
```

Step, then check the register:

```
M>s
  PC=200002
M>reg r2
  reg r2=00000001
M>
```

This should return "reg r2=00000001".

Step some more and verify that GPR1 increments after every four steps:

```
M>s 4
  PC=200002
M>reg r2
  reg r2=00000002
M>
```

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.

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.

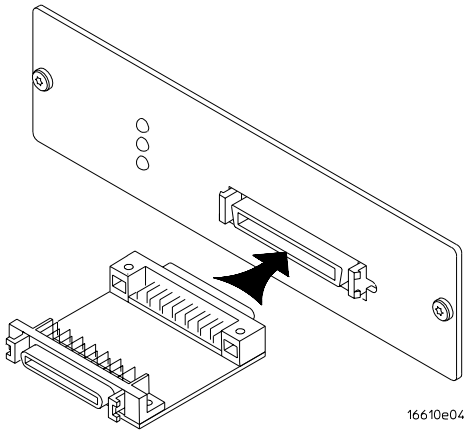
If there are many subnet masks in use on the local subnet, the logic analysis system may take a very long time to connect to the network after it is turned on.

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 (Agilent 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 (Agilent 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 Ctrl-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: E3499C Series Emulation System
    Test  1: Powerup PV Results                              Passed!
    Test  2: Target Probe Feedback Test                      Passed!
    Test  3: Boundary Scan Master Test                       Passed!
    Test  4: I2C Test                                        Passed!
    Test  5: Data Lines Test                                 Passed!
  PASSED  Number of tests: 1          Number of failures: 0

        Copyright (c) Agilent Technologies 1987-2000
All Rights Reserved. Reproduction, adaptation, or translation without prior
written permission is prohibited, except as allowed under copyright laws.

  E3499C Series Emulation System
    Version    A.07.56 28Sep98
    Location:  Generics

  E3455A Motorola M-Core Emulator
    Version:   A.01.02 28Sep98
M>
```

You may get an error like "!ERROR 172! Bad status code (0xff) from the hard reset sequence" just before the prompt. This is because the self-test loopback connector is installed instead of being connected to a real target system. You may also get a "?>" prompt for the same reason, and this is normal and expected. Any errors after the "PASSED Number of tests: 1 Number of failures: 0" line can be ignored.

If a performance verification test fails

There are some things you can do if a failure is found on one of these tests. Details of the failure can be obtained through using a -v option ("verbose" level) of 2 or more.

> If the particular failure you see is not listed below, contact Agilent Technologies for assistance.

TEST 5: Target Probe Feedback Test

A verbose output on this test can be extensive. For example, the following is the output of this test if you forget to plug in the loopback test board.

```
p>pv -t5 -v2 1

  Testing: E3499A Series Emulation System
    Test # 5: Target Probe Feedback Test                        failed!
      Bad 20 Pin Status Read when unconnected = 0x7fb7
                             Expected Value = 0xffb7
      Bad 20 Pin Status Read when connected = 7fb7
                             Expected Value = 0x7fb7
```

```
     Output 19 Low not received on Input 11
     Output 11 Low not received on Input 19
     Output 13 Low not received on Input 1
     Output 12 High not received on Input 6
     Output 12 and Input 6 not pulled high on release
     Output 8 Low not received on Input 10
     Output 7 Low not received on Input 20
     Output 4 Low not received on Input 14
     Output 2 Low not received on Input 18
FAILED Number of tests: 1           Number of failures: 1
```

If the you get a verbose output like this, check to make sure that the loopback test board was connected properly.

TEST 6: Boundary Scan Master Test
TEST 7: I2C Test

If these tests are not executed, check that you have connected the loopback test board.

If these tests fail, return the emulation module to Agilent Technologies for replacement.

Part 4

Reference

14

Specifications and Characteristics

Emulation Module Operating Characteristics

The following operating characteristics are not specifications, but are typical operating characteristics for the Agilent Technologies 16610A emulation module and M•CORE target interface module.

Operating Characteristics	
Microprocessor Compatibility	Motorola M•CORE rev 1.5 core and the MMC2001 microprocessors.
Environmental Characteristics (Temperature, Altitude, Humidity)	The Agilent Technologies 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

Maximum Ratings

Characteristics for the MPC500 Embedded PowerPC emulation module	**Symbol**	**Min**	**Max**	**Unit**
Input voltage range	Vin	-0.5	5.5	V
Input voltage range	Vtt	1.3	1.7	V
Input High Voltage	Vih	2/3Vtt + 0.2		V
Input Low Voltage	Vil		2/3Vtt - 0.2	V
Input High Current	Iih		-15	µA
Input Low Current	Iil		100	µA
Output High Voltage	Voh	2.4	3.3	V
Output Low Voltage	Vol		0.5	V
Output High Current	Ioh	8		mA
Output Low Current	Iol	-16		mA

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 Agilent Technologies 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 Agilent Technologies sales office. Ask them for the address of the nearest Agilent Technologies service center.
3. Package the part and send it to the Agilent Technologies service center.

   Keep any parts which you know are working. For example, if only the target interface module is broken, keep the emulation module and cables.
4. When the part has been replaced, it will be sent back to you.

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

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

In some parts of the world, on-site repair service is available. Ask an Agilent Technologies 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 Agilent Technologies Sales Office for further information.

Exchange assemblies are available when a repairable assembly is returned to Agilent Technologies. These assemblies have been set up on the "Exchange Assembly" program. This 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.

Emulation Module Exchange Assemblies	
Agilent Part Number	**Description**
E3455-69401	Programmed emulation module

Emulation Module Replaceable Parts	
Agilent Part Number	**Description**
0950-3043	Power Supply
E3494-61602	14-pin target cable
E3496-61601	50-pin cable
E3455-66501	Target Interface Module (M•CORE)
E3496-66502	Processor probe loopback test board
E3481-66507	Stimulus/self-test board for the Target Interface Module

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 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 probe.

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*.

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 Agilent Technologies 16500, 1660-series, or 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 Agilent Technologies 16505A prototype analyzer acts as an analysis and display processor for the Agilent Technologies 16500B/C logic analysis system. It provides a windowed interface and powerful analysis capabilities. Replaced by Agilent Technologies 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 Agilent Technologies' term for a set of tools for debugging your target system. A solution includes probing, inverse assembly, the Agilent Technologies B4620B source correlation tool set, and an emulation module.

stand-alone logic analyzer A stand-alone logic analyzer has a pre-defined 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 microprocessor) and makes them available for probing.

Index

Symbols

#CS4-1 label, 76, 81
#EB3-0 label, 76, 81
#OE label, 76
#SHS label, 76
#TA label, 76
#TEA label, 76, 81
+ state symbol, 98
? state symbol, 98

Numerics

1 byte predefined symbol, 81
1/4-flexible adapter, 206
14-pin cable, 21, 67, 201
16-bit data bus, 28
16-bit memories, 29, 78
16-bit memory port size, 28
2 bytes predefined symbol, 81
32-bit data bus, 28
32-bit logical address, 28
3M 2520-6002, 32
4 bytes predefined symbol, 81
50-pin cable, 67, 182, 183, 201
8-bit accesses to memory, 28, 29, 78

A

A[0] signal, 28, 29, 78
A[22:0] signals, 27
A[22:1] signals, 28
A0 signal, 28, 79
access size, 136
access to source code files, 103
acquire data, 154
active signals, 38
activity indicators, 110, 113
adapter, 33
adapters, 112
ADDR label, 76, 82, 113
address, 192
Address Not In The Memory Map! message, 98
address offset field, 90
address range, 192
  memory regions 0-7, 100
address signals, 28
Agilent 01650-63203 termination adapter, 32
Agilent 1251-8106, 32
Agilent 1252-7431, 32
Agilent Technologies 01650-63203 termination adapter, 36
Agilent Technologies 16550A logic analyzer, 31, 56, 57, 75, 116
Agilent Technologies 16554/55/56/57 logic analyzers, 56, 58, 59
Agilent Technologies 16554A logic analyzer, 31, 75
Agilent Technologies 16555A logic analyzer, 31
Agilent Technologies 16555A/D logic analyzer, 75
Agilent Technologies 16555D logic analyzer, 31
Agilent Technologies 16556A logic analyzer, 31
Agilent Technologies 16556A/D logic analyzer, 75
Agilent Technologies 16556D logic analyzer, 31
Agilent Technologies 16557D logic analyzer, 31, 75
Agilent Technologies 16600A logic analyzer, 31, 56, 61, 75
Agilent Technologies 16601A logic analyzer, 31, 56, 62, 75
Agilent Technologies 16602A logic analyzer, 31, 56, 63, 64, 75
Agilent Technologies 16603A logic analyzer, 31, 56
Agilent Technologies 16610A emulation module, 4, 21, 190
Agilent Technologies 16710/11/12A logic analyzers, 56, 65
Agilent Technologies 16710A logic analyzer, 31, 76
Agilent Technologies 16711A logic analyzer, 31, 76
Agilent Technologies 16712A logic analyzer, 31, 76
Agilent Technologies B4620B source correlation tool set, 2, 3, 4, 50, 88, 154, 155, 156, 158
Agilent Technologies E2631A inverse assembler, 4
Agilent Technologies E5346-44701 shroud, 33
Agilent Technologies E5346-60002 adapter, 33
Agilent Technologies E5346A, 32
Agilent Technologies E5346A high-density termination adapter cable, 33
Agilent Technologies E9512A Option 001 emulation solution, 2, 4, 20, 22
Agilent Technologies E9612A Option 001 inverse assembler, 3, 4, 20
Agilent Technologies sales office, 200
Agilent Technologies service center, 200
AMP 2-767004-2, 32
analysis arm, 166
analysis mode
  changing, 84
  state, 84, 104
  timing, 84
analysis probe, 203
arm, 115, 119, 166, 168
arm event, 166, 168
arm signal, 165
aruba processor type, 133
ASCII format symbol file, 192
assembly language mnemonics, 74
assembly-level listing, 154

Index

asynchronous sampling, 85

B

background mode, 171
background monitor, 174
base address, 78
base address, chip selects, 79
BKG LED, 183
board space, 32, 36
bonded-out M-CORE microprocessor, 26
boot firmware, 172
Boundary Scan Master Test, 185
break emulator on trigger, 157
BREAK IN signal, 38
Break In type, 135
break into monitor, 135, 165
break temporarily, 165
breakpoint, 161, 167
  triggering analyzer on, 167
breakpoint registers, 167
breakpoints, 39, 140, 151, 163
  clearing, 124
  hardware, 151
  setting, 124, 137
  software, 167
Breakpoints window, 137
built-in commands, 171, 173, 174, 176
  on-line help, 174
built-in performance verification test, 49
built-in terminal interface, 129, 131, 133, 171
bus activity, processor, 161
bus timing measurements, 29
byte enable signals EB[3:0], 28, 78

C

caches, 114
capacitive loading, 112
captured data, source code associated with, 101
captured execution, displaying, 95
capturing execution, 87
CD-ROM, 20, 31
center inline pins, 34
cf rrt, 135
cf speed, 134
cfsave -r command, 133
cfsave -s command, 133
channel assignments, 74
characteristics, 189
Chart tool, 154
chip select, 78
chip select information, 79
chip selects, 79
classes, 194
cleaning the instrument, 202
CLKOUT (Clock Out) signal, 29
CLKOUT label, 76
clock rate, 171
clock signals, 109
clock speed, 134
  processor, 159
clock speeds above 50 Mhz, 36
CMCRIM_1 configuration file, 58, 60, 61, 62, 63, 64, 65, 75
CMCRIM_2 configuration file, 57, 75
CMCRIM_3 configuration file, 58, 60, 61, 62, 63, 64, 65, 75
CMCRIM_4 configuration file, 57, 75
color, 99
color coding for memory regions, 78
Command Line window, 173, 176
command script files, 124
Comments, 198
complex breakpoints, 39
configuration files, 20, 50, 51, 74, 75, 80, 84, 85, 88, 96, 114, 116, 118, 128
  loading, 74
configuration items, 131
configuration options, 174
  setting, 133
Configuration window, 159
connector headers, 22
connectors for logic analyzer probe pods, 26
control execution, 154
cooling, 112
coordinating logic analysis, 153
counter overflow, 119
crash, system, 161
crosstalk, 38
CS (chip select) signals, 28
CS[4:1] signals, 27, 28
CS1 predefined symbol, 81
CS2 predefined symbol, 81
CS3 predefined symbol, 81
CS4 predefined symbol, 81
CS4-1 label, 76, 81
CSE (emulation chip select) signals, 28
CSE[1:0] signals, 27, 28
CSE1-0 label, 76, 81
custom probe fixtures, 112
cycle types, 100
cycling power, 174

D

D[15:0] signals, 28, 57, 58, 60, 61, 62, 63, 64, 65, 78
D[31:0] signals, 27
D[31:16] signals, 28
data accesses, 29, 78
DATA label, 76, 113
data reads, 78
Data Retrieval Error message, 98
data writes, 28
debug mode, 38, 172, 177
debug port, 67, 203

debugger, 22, 23, 124, 129, 154, 155, 161, 165, 166, 168, 171, 173, 177, 182
  connections, port numbers, 143
  using, 132
  writing your own, 175
decode 8-bit accesses, 29, 78, 79
deep memory logic analyzer, 102
default port number, 173
direct branches, 100
display timing analysis mode data, 104
Display tools, 154
displaying captured execution, 95
download programs, 124, 137
download symbol information, 80

E

EB[3:0] (Enable Byte) signals, 29, 79
EB0 predefined symbol, 81
EB0-EB1 predefined symbol, 81
EB1 predefined symbol, 81
EB2 predefined symbol, 81
EB2-EB3 predefined symbol, 81
EB3 predefined symbol, 81
EB3-0 label, 76, 81
EIM based devices, 27
elastomeric probe adapter, 203
electrical characteristics
  emulation module, 190
electrical problem, 171
electrostatic discharge, 44
emulation
  preparing target for, 38
Emulation Control Interface, 21, 22, 50, 69, 70, 129, 133, 136, 142, 154, 159, 161, 163, 166, 168, 171, 173, 182
  start from the Workspace window, 126
  starting from main system window, 124, 125
  using, 123, 129
Emulation Mode, 27
emulation module, 2, 3, 124, 154, 162, 166, 182, 183, 200, 203
  built-in commands, 171
  commands, entering, 129
  communication, 70
  configuring, 127
  connected to Port Out, 164
  electrical characteristics, 190
  environmental characteristics, 190
  equipment required, 22
  equipment supplied, 20
  exchange assemblies, 201
  firmware, 21, 50, 177
  firmware version, 70, 174
  firmware, updating, 69
  flash memory, 133
  icon, 158
  initialize, 174
  installing, 44
  loopback test board, 21
  microprocessor compatibility, 190
  operating characteristics, 190
  Port In connected to, 164
  port number, 173
  problems, 182
  repair strategy, 201
  replaceable parts, 201
  slot 1, 46
  status, 174
  status lights, 172
  stop processor on trigger, 158
  target system connection, 66
  telnet to, 173
  terminal interface, 129, 131
  testing, 49, 136
  trigger, 156
  trigger signal, 163
  triggering logic analyzers, 163
  troubleshooting, 169
  using the, 121
emulation modules
  connected in series, 164
emulation probe, 201, 203
emulation reset, 174
emulation solution, 2
emulator command script files, 124
emulator configuration, reading or writing, 124
Emulator icon, 126
emulator status, 124
end address, 78
entitlement certificate, 21
environmental characteristics
  emulation module, 190
equipment, 19
equipment and software
  optional, 23
  required, 22
  supplied, 20
erratic trace measurements, 112
!ERROR 172! Bad status code (0xff) from the hard reset sequence, 184
!ERROR 905! Driver firmware is incompatible with ID of attached device, 177
error messages, subnet mask, 180
ERROR predefined symbol, 81
Ethernet networks, 103
event wasn't captured, 115
examples of measurements, 155
exchange assemblies, 201
  emulation module, 201
executed instruction, 98
executing user code, 172
expansion frame, 173
EXT predefined symbol, 81
extender, 203
extenders, 112
external access, 28

external data reads, 100
external data writes, 100
External Interface Module (EIM), 26
external pins, 27
external read cycle, 30
external write cycle, 30

F

failed bus cycle, 29, 79
false trigger, 165
Fast Mode, 27
filter library code execution, 92
filters, inverse assembler, 98
firmware, 171
  corrupted, 172
  emulation module, 50, 69
  updating, 124
firmware version, 174, 175
  emulation module, 70
first emulation module, 173
fixed code offsets, 90
flash memory, 133
flexible adapter, 203
frequencies greater than 50 MHz, 112
ftp, 103
function, 92
function calls, 161
FUNCTIONS, 195

G

gateway address, 143, 180
General-Purpose ASCII (GPA)
  symbol file, 81
  symbol file format, 191
general-purpose flexible adapter, 204
general-purpose ports, 27
general-purpose probes, 36
generic commands, 175
glitches, 155
glossary, 203
GPA record format summary, 193
GPIO signals, 27
GPR1 register, 176
ground returns, 34
grounded wrist straps, 44
Group Run, 163, 164, 166, 168

H

halt the processor, 135, 157
hardware breakpoint registers, 167
hardware breakpoints, 151
hardware problem, 182
header connector, 32, 36
headers, 22
high-density adapter, 204
high-density connector
  mechanical specifications, 33
  pin assignment, 34
high-density connector to RIM signal pin mapping, 35
high-density connectors, 26, 32
high-level source code, 101, 154
high-level source debugger, 22
hostname, 143
HW accel instructions, 100

I

I/O memory, reading or writing, 124
I/O window, 136
I2C Test, 185
IAMCRIME inverse assembler file name, 96
illegal instruction, 167
IMB3 predefined symbol, 81
In Monitor, 164, 165
In Monitor signal, 163
incorrect inverse assembly, 113
incorrect signal levels, 109
indirect branches, 100
initial prompt, 177
inline pins, 34
input voltage, maximum, 32
installation, overview of, 16
installing logic analyzer modules, 43
installing software, 50
installing the emulation module, 44
instruction accesses, 29, 78
instruction fetches, 28
instruction reads, 78
instruction register, 177
instrument, cleaning the, 202
INT predefined symbol, 81
Intel Hex file, 137
Intel Hex format, 124
intermittent data errors, 109
intermodule bus, 124
intermodule measurement, 159, 163
  problems, 115
  trigger, 135, 161
Intermodule window, 156, 158, 164, 166, 167
internal 32-bit accesses, 28, 29, 78
internal access, 28
internal analyzer delays, 115
internal data reads, 100
internal data writes, 100
internal pipeline, 29
internal sample rate clock, 85
internal show cycle, 30
interrupt service routines, 135
Invasm menu, 96
inverse assembled data, 97
inverse assembler, 2, 3, 26, 50, 51, 74, 75, 96, 102, 114, 204
  equipment required, 22
  equipment supplied, 20
  file name, 74, 116
  filters, 98
  loading, 74
  messages, 98
  not found, 116
  preferences, 77

problems, 113
software, 20
will not load, 114
inverse assembly, 84, 114
incorrect, 113
IP address, 143, 180

J

JTAG (OnCE) port, 3, 21, 38, 66, 67
See *debug port*, 203
JTAG clock speed, 134
JTAG connector, 38
mechanical and pinout information, 39
JTAG emulation
optional signals, 38
required signals, 38
JTAG scan chains, 39
JTAG signals, 38, 39
jumper, 204

K

K clock, 84

L

label, 204
label names, 74
LAN activity, 180
LAN address, 173, 174, 180
LAN cable, 180
LAN communication
verifying, 144
LAN interface problems, 180
LAN LEDs, 180
LAN name, 173
LAN port assignment, 124
LAN protocols, 103
LAN system administrators, 103
latency, 159
layout, 112
library code execution, 93
line number, 92
linker map file, 192
Listing display window, 96
Listing tool, 154
Load Executable window, 137
loading configuration files, 74
loading object file symbols, 81
loading symbol information, 80
loading, minimum, 32
locked status line, 113
logic analysis
coordinating, 153
preparing target for, 26
logic analysis system, 180, 200
setting up, 41
software version, 31
logic analyzer, 108
channel assignments, 74
configuration files, 75, 80
configuring, 73
connecting to target system, 56
connector headers, 22
deep memory, 102
label names, 74
maximum input voltage, 32
messages, 116
modules, installing, 43
pods, 113, 118
solving problems, 109
stopping, 159
storage qualification, 92, 93
trigger setup, 89
triggering, 38
triggers, 88
troubleshooting, 107
using the, 71
logic analyzer pods, 31, 36
logic analyzers
supported, 31
loopback connector, 184
loopback test board, 21, 182, 183, 185
lower 16-bits of data, 28, 78
lower data bits, 28

M

mainframe logic analyzer, 204
male-to-male header, 204
marginal timing, 109
maskable break, 135
master clock dialog, 84
master clock signal, 84
Master Mode, 27
maximum input voltage, 32
mcore processor type, 133
MCR register, 27
MCU_DE signal, 38, 39
measurement examples, 155
measurement initialization error, 116
measurement, viewing, 159
mechanical information, JTAG connector, 39
mechanical specifications
high-density connector, 33
medium-density connector, 26, 32, 36
pinouts, 36
RIM signal pin mapping, 37
memory, 154
memory accesses
testing, 136
Memory Disassembly window, 154
memory management units, 90
memory managers, 114
memory map information, 29, 78
Memory Mnemonic window, 137
memory region, 78, 98, 99
accesses, 100
Memory window, 136
memory, mnemonic form, 124
memory, reading or writing, 124
messages
inverse assembler, 98
logic analyzer, 116

microprocessor bus cycles, 99
microprocessor caches, 114
microprocessor compatibility
  emulation module, 190
Mictor (Matched Impedance
    ConnecTOR) connectors, 32
minimum loading, 32
MMC2001, 27
  processor type, 133
mnemonics, assembly language, 74
Module Configuration Register, 27
Motorola S-Record file, 137
Motorola S-Record format, 124

N

NC (no connect), 35, 37
network access to source files, 103
NFS client/server, 103
no configuration file loaded, 118
no connect, 35
no inverse assembly, 113
no target system power, 172, 174
NONE predefined symbol, 81
non-maskable break, 135
NONRELOC attribute, 194
NORMAL attribute, 194
nostate, trigger on, 161
NULL pointer de-references, 155

O

object file formats, 80
object file symbols, 88
  loading, 81
object files, 192
occurrences remaining in level 1,
    166, 168
OE label, 76
on-board termination, 36
OnCE port, 38
On-Chip Emulation (OnCE), 38
on-chip hardware breakpoint
    registers, 167
one-card Agilent Technologies
    16550A installations, 117
on-line help for built-in commands,
    174
opcode fetch, 119
operating characteristics
  emulation module, 190
operating system, 51
optional 5th pod, 29
optional equipment and software,
    23
optional RIM signals, 26, 28
optional signals, 31, 78
optional signals for JTAG
    emulation, 38
oscilloscope, 109, 115, 167, 168
  measurement, 161
  modules, installing, 43
other instructions, 100
overview of installation, 16

P

pattern generator modules,
    installing, 43
patterns, 89
PEPAR register, 27
performance verification tests, 49,
    124, 171, 182, 183
  failures, 184
performance, profile system, 155
personality, 70
personality files, 50
pin mapping, signal to connector,
    35, 37
pin protectors, 112
pinout information, JTAG
    connector, 39
pipeline depth, 109
pipeline, internal, 29
plastic shroud, 33
pods, logic analyzer, 31, 56, 113,
    118, 205
point symbol, 192
poor connections, 109
Port E Pin Assignment Register, 27
Port In, 164
port number, 143, 144, 173
Port Out, 164
port size, 28, 78
  chip selects, 79
power supply, 201
power up, 110
power-on self test, 180
power-on sequence, 111
power-ON/OFF sequence, 42
predefined symbols, 80
preferences, inverse assembler, 77
prefetch queue, 109
prefetches, 109
preprocessor
  See *analysis probe*
preprocessor interface
  See *analysis probe*
probe adapter
  See *elastomeric probe adapter*
probe fixtures, custom, 112
probed signal lines, 31
probing the target system, 55
problems
  emulation module, 182
  intermodule measurement, 115
  inverse assembler, 113
  LAN interface, 180
  logic analyzer, 109
  probing, 111
  target system, 176
procedures, 195
processor activity, 161
processor bus, 168
  activity, 161
  state analysis of, 161
processor clock speed, 159
processor execution
  coordinating, 153
  stopping on analyzer trigger, 156

stops trigger the analyzer, 163
processor halt, 167, 168
trace before, 161
triggering on, 166
processor probe, 140
See *emulation probe*
processor revision, 177
processor support package, 50, 51, 74
processor type, 133
product number, 174
profile system performance, 155
program counter, 155, 165, 176
program entry point, 197
program functions, 195
program stack, 165
program, running a, 137
programmed emulation probe, 201
prompt, initial, 177
prototype analyzer, 205
PRU (Port Replacement Unit), 27, 28
PSTAT signals, 29, 76
PSTAT[3:0] signals, 29
pull-up resistor, 39
pulse shape requirements, 109

Q

qualified processor clock, 166, 168

R

R/#W label, 76, 81
R/W signal, 27, 28
R0 register, 175
ranges, 89
read cycle, 28
read predefined symbol, 81
real-time runs, 135, 171
recommended circuit board routing, 34
record header, 193
reference, 187
reference voltage, 38
REG predefined symbol, 81
regions of memory, 194
register values, initializing, 137
registers, 154
reading or writing, 124
Registers window, 137
relocated code, compensating for, 90
repair strategy, 201
repeat a command, 174
replaceable parts, 201
emulation module, 201
required RIM signals, 27
required signals for JTAG emulation, 38
requirements, 19
equipment and software, 22
RESET button, 174
RESET LED, 183
RESET signal, 178
reset state, 172
reset the target processor, 174
resources, 89
restrict to real-time runs, 135, 171
return a part, 200
rev 1.5 (RLB) bondout, 27
revision, processor, 177
RIM 1.5 and RIM+ memory controllers, 27
RIM control registers, 27
RIM signal to high-density connector pin mapping, 35
RIM signal to medium-density connector pin mapping, 37
RIM signals
optional, 28
required, 27
RIM timing relationships, 30
RISC Local Bus (RLB), 26
RLB addresses, 78
RLB Integration Module (RIM), 26
RLB signals, 26
ROM code, 151
RST signal, 38, 39
run control probe
See *emulation module*
See emulation probe
Run Control window, 70, 137, 157, 159, 166, 168
running a program, 137
running in background, 174
running user program, 174
Running, emulation module state, 164

S

sample period, 85
sample rate clock, internal, 85
search path, source code, 93, 103
second emulation module, 173
section relocation, 195
SECTIONS, 194
segments, 194
selected file is incompatible, 118
serial number, 200
service guide, 199
setting breakpoints, 167
setup and hold time requirements, 109
Setup Assistant, 42, 50, 53, 74, 205
Setup window, 80
setup, overview of, 16
SHEN bit, 27
show cycles, 27
shroud, 33
SHS (show strobe) signal, 27, 29, 84
SHS label, 76
shunt connector
See *jumper*
signal ground returns, 34
signal information, 78
signal integrity, 32, 39, 109

signal to connector pin mapping, 35
signal to pin mapping, 37
signal traces, 38
signals required for JTAG emulation, 38
Single Mode, 27
single-step, 140, 174
SingleStep debugger, 147
SIZ1-0 label, 76, 81
skew, 115
skid effect, 159
slave clock signal, 84
slot 1 (emulation module), 46
slow clock, 171
slow clock error message, 166, 168
slow or missing clock, 118, 171
software addresses, 74, 102
software breakpoints, 151, 167
Software Development Systems debugger, 147
software probe, 140
  See *emulation probe*
software supplied, 20
software version, 31
software, installing, 50
solution, 205
source code, 101, 154
  associated with captured data, 101
  search path, 93
  triggering on, 92
source correlation tool set, 3, 22, 101, 103
  included software, 21
source debugger, 22
source files
  network access to, 103
  search path, 103
  version control, 103
source line trigger, stopping processor execution, 156
SOURCE LINES, 197
Source Viewer window, 92, 155, 156, 158
Source window, 156
source-level listing, 155
specifications, 189
speed, JTAG clock, 134
speed, processor clock, 159
STA3-0 label, 76
stack, program, 165
stand-alone logic analyzer, 205
START ADDRESS, 197
STAT label, 76, 113
STAT_B label, 76
state analysis, 161, 206
state analyzer, 168
state mode, 84, 104
  changing to, 84
state of the processor, 154
state symbols, 98
status display, 166
status lights, 172
status lines, locked, 113
status, emulation module, 174
step a single instruction, 175
stop on any trigger, 158
stop the analyzer, 159
storage qualification, 92, 93, 109, 114
subnet, 180
subnet mask, 180
  error messages, 180
subroutines, 195
supplied equipment and software, 20
support shroud, 33
supported logic analyzers, 31
suppressing all other operations, 99
surface mount connector, 33
SW_ADDR label, 74, 78, 102
symbol file, 101
symbol file format, General-Purpose ASCII (GPA), 191
symbol information, 192
  loading, 80
symbol name, 192
Symbol Selector dialog, 82
symbols for inverse assembled states, 98
Symbols tab, 80
symbols, displaying, 97
synchronization, 113
System Admin window, 180
system administrators, 103
system clock, 29
system crash, 161
System Performance Analyzer tool, 154
system performance, profile, 155

T

TA label, 76
target control port, 206
target interface module, 3, 4, 21, 69, 182, 200, 201, 206
target power, 174
target power sense, 38
Target Probe Feedback Test, 184
target processor, reset, 174
target reset, 174
target system, 108, 113, 154
  connecting logic analyzer to, 56
  emulation module communication, 70
  emulation module connection, 66, 67
  preparing, 25
  probing, 55
  problems, 176
  testing, 136
  won't boot up, 111
TC (Transfer Code) signals, 29
TC[2:0] signals, 27, 29
TC2-0 label, 76
TCK signal, 38

TCP/IP protocol, 103
TDI signal, 38
TDO signal, 38
TEA (Transfer Error Acknowledge) signal, 29, 79, 98
TEA label, 76, 81
telnet, 103, 173, 176, 183
  port numbers, 143
temporary breaks, 165
terminal interface, 129, 131, 171, 173
termination, 22, 32
  on-board, 36
termination adapter, 36
TEST 5: Target Probe Feedback Test, 184
TEST 6: Boundary Scan Master Test, 185
TEST 7: I2C Test, 185
test, performance verification, 49
testing, emulation module, 49
threshold level, 109
TIM
  See *target interface module*
time from arm greater than 41.93 ms, 119
timing analysis, 161, 206
timing analysis mode, 84
  changing to, 85
  data, displaying, 104
timing analyzer, 167, 168
timing mode, 29
timing relationships, RIM, 30
timing requirements, 109, 112
TMS signal, 38
tools, 154
Torx T-10 screwdriver, 21
Torx T-15 screwdriver, 21
trace about this line, 156
trace until processor halt, 163
traces, signal, 38
transition board, 206
trigger condition, 109
trigger in an intermodule measurement, 135
trigger on anystate, 167
trigger on nostate, 161
trigger pattern, 119
trigger point, 161, 166, 167
trigger position, 161
trigger sequence, 90
trigger sequencer specification, 110
trigger specification, 206
triggering on source code, 92
troubleshooting, 107
troubleshooting guide, 171
TRST signal, 38
TSIZ (transfer size) signals, 28
TSIZ[1:0] signals, 27, 28
two-card Agilent Technologies 16550A installations, 117
type, memory region, 78

U

unexecuted instructions, 98, 100
unknown state, 174
unneeded information, 99
unused prefetch, 110
unwanted triggers, 109
updating emulation module firmware, 69
User Defined Symbols tab, 80
USER LED, 183
user-defined symbols, 80

V

valid data, 28
variable, 92
VARIABLES, 196
Vdd signal, 38
vector base address, 78, 79
vector table, 79
verify emulation module communication, 70
version control, source file, 103
version information, emulation module firmware, 70
version, software, 31
view a measurement, 159

W

waiting for trigger, 119
Waveform display, 104
word-aligned addresses, 119
write cycle, 28
write predefined symbol, 81

X

X Windows server software, 140, 145
X-Window client/server, 103

Y

Y-cable, 33

Index

Restricted Rights Legend

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subparagraph (C) (1) (ii) of the Rights in Technical Data and Computer Software Clause in DFARS 252.227-7013. Agilent Technologies, 3000 Hanover Street, Palo Alto, CA 94304 U.S.A. Rights for non-DOD U.S. Government Departments and Agencies are set forth in FAR 52.227-19 (c) (1,2).

Document Warranty

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

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

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

Safety

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

Warning

- Before turning on the instrument, you must connect the protective earth terminal of the instrument to the protective conductor of the (mains) power cord. The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. You must not negate the protective action by using an extension cord (power cable) without a protective conductor (grounding). Grounding one conductor of a two-conductor outlet is not sufficient protection.
- Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuseholders. To do so could cause a shock of fire hazard.
- Service instructions are for trained service personnel. To avoid dangerous electric shock, do not perform any service unless qualified to do so. Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.
- If you energize this instrument by an auto transformer (for voltage reduction), make sure the common terminal is connected to the earth terminal of the power source.
- Whenever it is likely that the ground protection is impaired, you must make the instrument inoperative and secure it against any unintended operation.
- Do not operate the instrument in the presence of flammable gasses or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.
- Do not install substitute parts or perform any unauthorized modification to the instrument.
- Capacitors inside the instrument may retain a charge even if the instrument is disconnected from its source of supply.

Safety Symbols

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

Hazardous voltage symbol.

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

WARNING

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

CAUTION

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

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

Product Warranty

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

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

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

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

Limitation of Warranty

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

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

Exclusive Remedies

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

Assistance

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

Certification

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

About this edition

This is the *Solutions for the Motorola M●CORE User's Guide*.

Publication number
E2631-97001, June 2000
Printed in USA.

Print history is as follows:
E2631-97000, November 1998
E2631-97001, June 2000

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