User's Guide

Publication number E2489-97004
July 2001

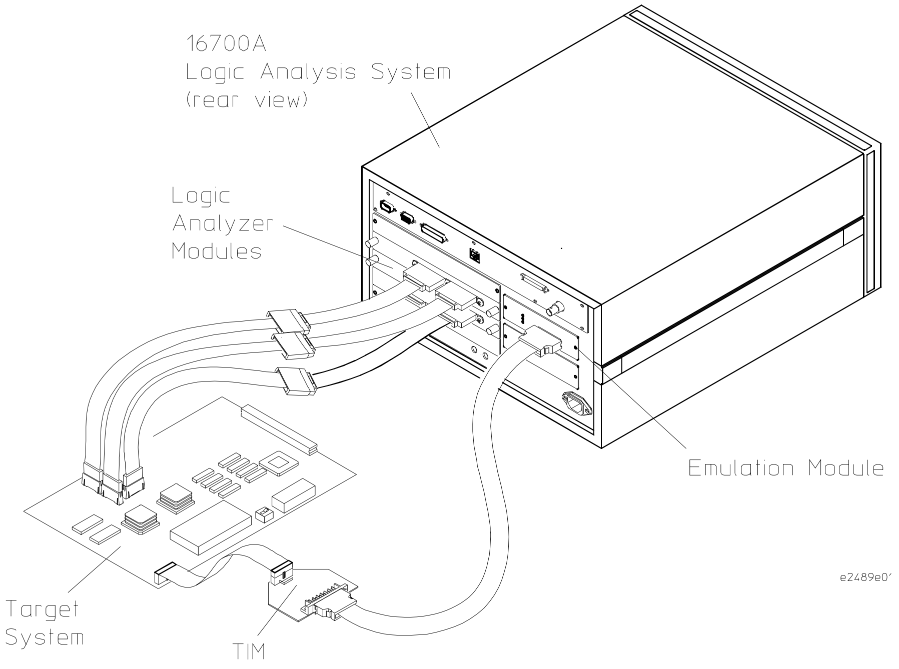
For Safety information, Warranties, and Regulatory information, see the pages behind the index.

Solutions for the Motorola MPC555

Solutions for the Motorola MPC555— At a Glance

The Agilent Technologies E9510A emulation solution lets you use the Agilent Technologies 16700-series logic analysis system to debug and characterize Motorola Embedded PowerPC MPC555 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 and Trace Reconstruction Tool

The inverse assembler for Motorola MPC555 processors, 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 E9610A Option 001 when ordered separately.)

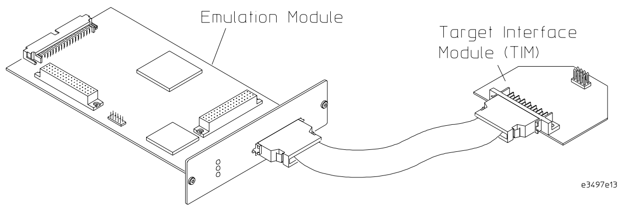
The inverse assembler can be used alone or with its trace reconstruction tool. When used alone, the inverse assembler reports activity seen on the external bus but doesn't reveal instructions executed out of MPC555 internal memory.

The trace reconstruction tool uses captured show cycles, captured VF/VFLS signals, and program image files (S-Records) to decode captured MPC555 execution into complete program traces. The trace reconstruction tool provides an accurate description of instruction execution, even when the instructions execute out of internal memory.

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 MPC555 microprocessor's debug 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:

Processor Supported	Product Ordered	Includes
MPC555	Agilent Technologies E9610A inverse assembler	E2489A inverse assembler
MPC555	E9510A emulation solution	E2489A inverse assembler 16610A emulation module target interface module (TIM) B4620B source correlation tool set

Before you use this book, you should have already set up the Agilent Technologies 16700-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 16700-series logic analysis system's on-line help for more information on using the Agilent Technologies B4620B source correlation tool set.

Contents

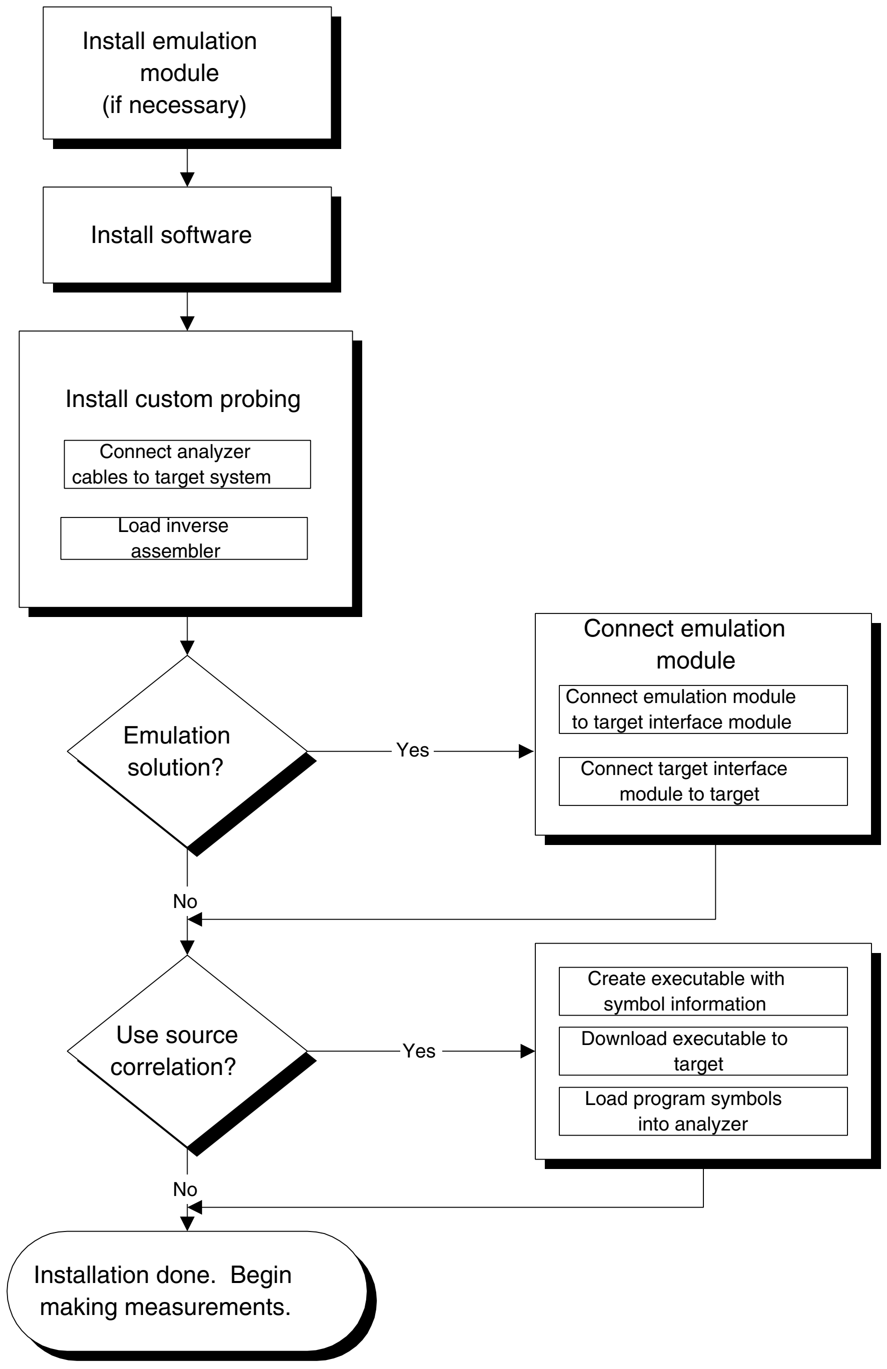
Part 1

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 Chapter 1, “Equipment and Requirements,” on page 19.
2. If you need to install an emulation module in an Agilent Technologies 16700-series logic analysis system, see “Installing the Emulation Module” on page 50.
3. Install the software. See “Installing Software” on page 57.
4. If you have an Agilent Technologies 16700-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 62.


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

1

Equipment and Requirements

Equipment and Software Supplied

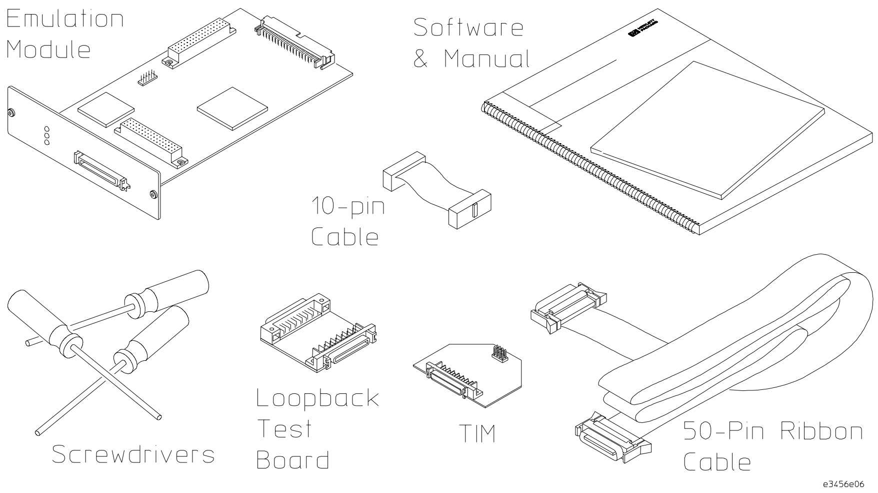
Listed below is the equipment and software supplied with the Agilent Technologies E9510A MPC555 emulation solution (which includes an inverse assembler, emulation module, and source correlation tool set).

Inverse Assembler

The inverse assembler (Agilent Technologies E9610A Option 001 when ordered separately) includes:

- Logic analyzer configuration files and the inverse assembler software on a CD-ROM (for Agilent Technologies 16700-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 16700-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 debug port in the target system.
- A 10-pin ribbon cable for connecting the target interface module (TIM) to a debug 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 16700-series logic analysis system software CD-ROM.

The Agilent Technologies B4620B source correlation tool set software is already installed on the Agilent Technologies 16700-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.

Trace Reconstruction Tool Requirements

The MPC555 inverse assembler works with or without the trace reconstruction tool.

Without the MPC555 trace reconstruction tool, the inverse assembler will disassemble external bus fetches only. Activity in cache is not revealed.

When using the trace reconstruction tool, the inverse assembler requires an S-Record file for direct/indirect branch show cycles. Complete program trace information is displayed in the Listing window, including activity in cache.

See “Displaying Data from the Trace Reconstruction Tool” on page 112 for instructions for using the trace reconstruction tool.

Additional Equipment and Software Required

Listed below is the additional equipment and software required by the Agilent Technologies E9510A emulation solution for the MPC555.

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 MPC555 processor execution.

Refer to the Chapter 2, "Preparing the Target System," on page 25 for more information on headers, terminations, and supported logic analyzers.

Emulation Module

The emulation module requires:

- An Agilent Technologies 16700-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 16700-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 16700-series logic analysis system.

Other Optional Equipment and Software

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

Unsupported Microprocessor Modes

The Agilent Technologies E2489A inverse assembler does not support the following MPC555 configurations:

LAST Burst Mode. LAST Burst Mode is not supported. BDIP mode is supported.

GPIO Pins. The primary signals on the 555 are needed for address reconstruction, data display, and status decoding. Using the general purpose ports will produce incorrect results in the listing display. If a Port Replacement Unit is installed on the target system, the external memory-mapped ports can still be used, as long as the primary signals are still sent to the logic analyzer.

The following primary status signals are used by the inverse assembler and the reconstruction tool and therefore must be available to the logic analyzer: RW, TS, TA, STS, VF[0:2], BURST, BDIP, OE, AT2, TSIZ[0:1], and BE[0:3].

Endian Mode. The inverse assembler supports big-endian mode. It does not support little-endian mode.

2

Preparing the Target System

Preparing for Logic Analysis and Inverse Assembly

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. Alternately, you can use medium-density connectors which take more circuit board space and have some clock speed limitations.

This section describes these considerations in more detail.

- Supported Logic Analyzers
- Using High-Density Connectors
- Using Medium-Density Connectors

Supported Logic Analyzers

The Agilent E2489A (MPC555) inverse assembler requires a minimum of six pods (102 logic analyzer channels). Additional logic analyzer pods are required if you choose to probe other optional signals. The logic analyzer configuration files assign signals to 14 pods. If fewer than 14 pods are available, only the configuration file definitions for the available pods will be used.

The inverse assembler works with logic analysis system software version A.02.40 or greater. The latest logic analysis system software version is on the CD-ROM shipped with this inverse assembler product.

The following logic analyzers can be used:

Agilent Technologies Logic Analyzer	Number of Cards	Channel Count
16550A	1 or 2 cards	102/card
16554A	2 or 3 cards	68/card
16555A	2 or 3 cards	68/card
16555D	2 or 3 cards	68/card
16556A	2, 3, or 4 cards	68/card
16556D	2, 3, or 4 cards	68/card
16557D	2, 3, or 4 cards	68/card
16710A	1 or 2 cards	102/card
16711A	1 or 2 cards	102/card
16712A	1 or 2 cards	102/card
16715A	2, 3, or 4 cards	68/card
16716A	2, 3, or 4 cards	68/card
16717A	2, 3, or 4 cards	68/card
16718A	2, 3, or 4 cards	68/card
16719A	2, 3, or 4 cards	68/card
16750A	2, 3, or 4 cards	68/card
16751A	2, 3, or 4 cards	68/card
16752A	2, 3, or 4 cards	68/card

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

MPC555 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	3M 2520-6002	Agilent part number 01650-63203	Not recommended above 50MHz.
2 rows x 19 pins	High	AMP 2-767004-2	Agilent E5346A	Each connector supports two logic analyzer pods.

More information is available by visiting www.agilent.com and entering the search string *Probing Solutions for Agilent Technologies Logic Analysis Systems.*

Information about 3m connectors is available on the web at: http://www.3m.com/us/electronics_mfg/interconnects/

Information about AMP connectors is available on the web at: http://www.amp.com/

Using High-Density Connectors

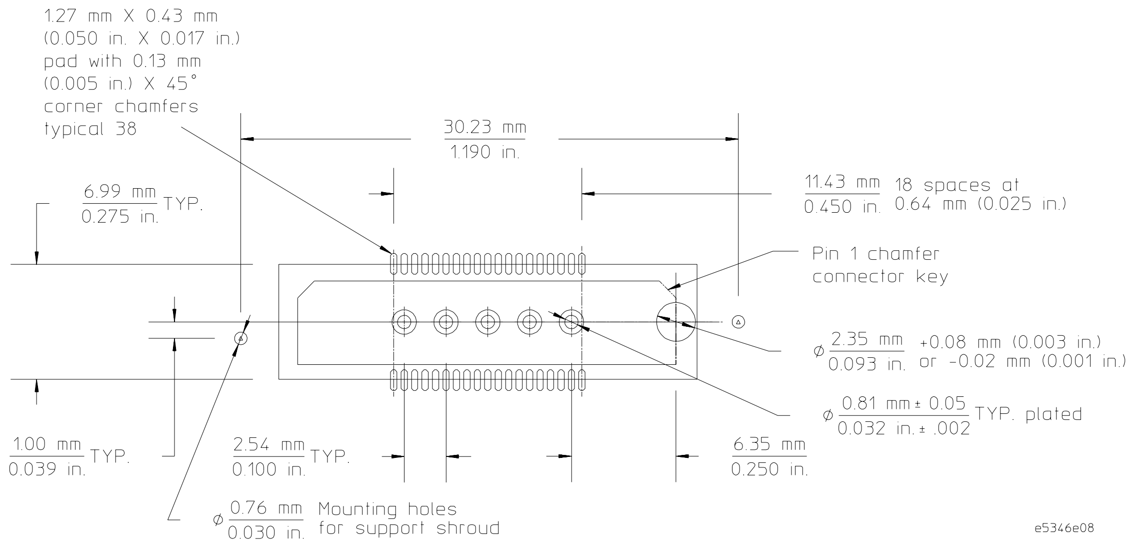
High-density AMP MICTOR (Matched Impedance ConnecTOR) connectors are recommended for connecting the target system to the logic analyzer because they require less board space and provide higher signal integrity than medium-density connectors. Each connector carries 32 signals and two clocks.

- 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.

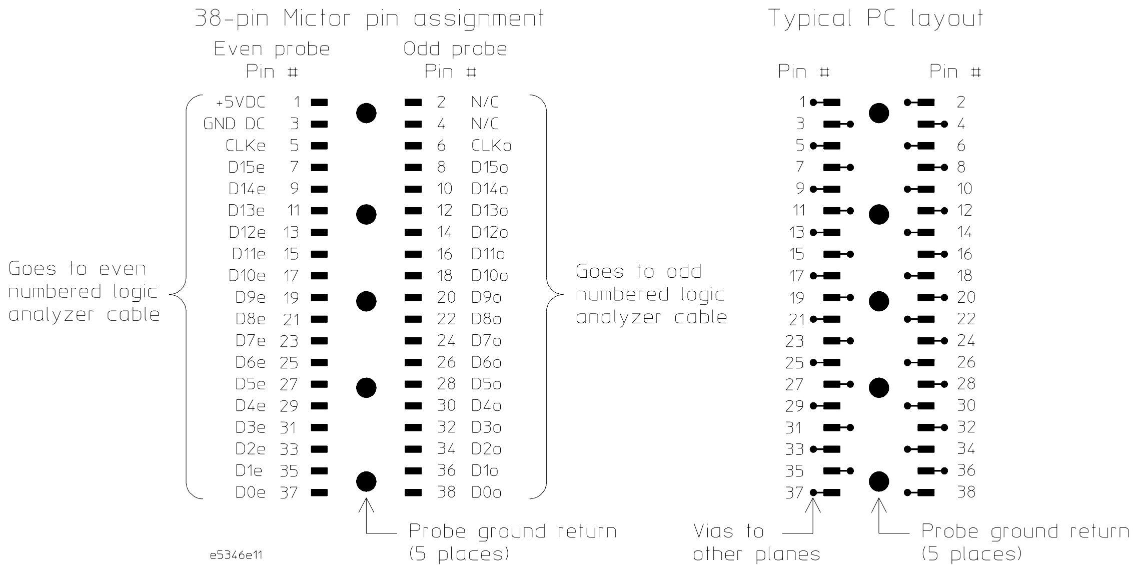
More information on this connector is available by searching for *High-Density Termination Adapter* at www.agilent.com.

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.


1.27 mm X 0.43 mm (0.050 in. X 0.017 in.) pad with 0.13 mm (0.005 in.) X 45° corner chamfers typical 38
30.23 mm / 1.190 in.
6.99 mm / 0.275 in. TYP.
11.43 mm / 0.450 in. 18 spaces at 0.64 mm (0.025 in.)
Pin 1 chamfer connector key
Ø 2.35 mm / 0.093 in. +0.08 mm (0.003 in.) or -0.02 mm (0.001 in.)
Ø 0.81 mm ± 0.05 / 0.032 in. ± .002 TYP. plated
1.00 mm / 0.039 in. TYP.
2.54 mm / 0.100 in. TYP.
6.35 mm / 0.250 in.
Ø 0.76 mm / 0.030 in. Mounting holes for support shroud
e5346e08

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.

MPC555 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**			
Odd		**Even**		**Odd**		**Even**		**Odd**		**Even**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
2	NC	1	NC	2	NC	1	NC	2	NC	1	NC
4	NC	3	NC	4	NC	3	NC	4	NC	3	NC
6	CLKOUT	5	$\overline{TS}$	6	$\overline{TA}$	5	$\overline{STS}$	6	TEA	5	RD/$\overline{WR}$
8	A[16]	7	PORESET	8	D[16]	7	D[0]	8	BURST	7	CR
10	A[17]	9	KAPWR	10	D[17]	9	D[1]	10	BDIP	9	$\overline{IRQ[1]}$/$\overline{RSV}$
12	A[18]	11	EXTCLK	12	D[18]	11	D[2]	12	OE	11	RSTCONF
14	A[19]	13		14	D[19]	13	D[3]	14	WE[0]	13	IWP[0]
16	A[20]	15	CS[0]	16	D[20]	15	D[4]	16	WE[1]	15	IWP[1]
18	A[21]	17	CS[1]	18	D[21]	17	D[5]	18	$\overline{WE[2]}$	17	IWP[2]
20	A[22]	19	CS[2]	20	D[22]	19	D[6]	20	$\overline{WE[3]}$	19	IWP[3]
22	A[23]	21	CS[3]	22	D[23]	21	D[7]	22	$\overline{AT[2]}$	21	LWP[0]
24	A[24]	23	A[8]	24	D[24]	23	D[8]	24	$\overline{TSIZ[0]}$	23	LWP[1]
26	A[25]	25	A[9]	26	D[25]	25	D[9]	26	$\overline{TSIZ[1]}$	25	DSCK (Optional)
28	A[26]	27	A[10]	28	D[26]	27	D[10]	28	$\overline{VFLS[0]}$	27	DSDO (Optional)
30	A[27]	29	A[11]	30	D[27]	29	D[11]	30	VFLS[1]	29	DSDI (Optional)
32	A[28]	31	A[12]	32	D[28]	31	D[12]	32	PTR	31	Future Compres-sion Pin
34	A[29]	33	A[13]	34	D[29]	33	D[13]	34	$\overline{IRQ[3]}$/$\overline{KR}$/RETR	33	VF[0]
36	A[30]	35	A[14]	36	D[30]	35	D[14]	36	SRESET	35	VF[1]
38	A[31]	37	A[15]	38	D[31]	37	D[15]	38	HRESET	37	VF[2]
Logic Analyzer Pod 1		Logic Analyzer Pod 2		Logic Analyzer Pod 3		Logic Analyzer Pod 4		Logic Analyzer Pod 5		Logic Analyzer Pod 6	

MICTOR Connector 4				**MICTOR Connector 5**			
Odd		**Even**		**Odd**		**Even**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
2	NC	1	NC	2	NC	1	NC
4	NC	3	NC	4	NC	3	NC
6	SCK/QGPIO[6]	5	$\overline{\text{ECK}}$	6	$\overline{\text{ENGCLK/BUCLK}}$	5	$\overline{\text{T2CLK, TPU\_A}}$
8	PCS[0]/$\overline{\text{SS}}$/QGPIO[0]	7	MDA[27]	8	MGPIO[0] *	7	TPUCH[0], TPU_A
10	PCS[1]/QGPIO[1]	9	MDA[28]	10	MGPIO[1] *	9	TPUCH[1], TPU_A
12	PCS[2]/QGPIO[2]	11	MDA[29]	12	MGPIO[2] *	11	TPUCH[2], TPU_A
14	PCS[3]/QGPIO[3]	13	MDA[30]	14	MGPIO[3] *	13	TPUCH[3], TPU_A
16	MISO/QGPIO[4]	15	MDA[31]	16	MGPIO[4] *	15	TPUCH[4], TPU_A
18	MOSI/QGPIO[5]	17	MPWM[0]	18	MGPIO[5]	17	TPUCH[5], TPU_A
20	TXD[1]/QGPO[1]	19	MPWM[1]	20	MGPIO[6]	19	TPUCH[6], TPU_A
22	TXD[2]/QGPO[2]	21	MPWM[2]	22	MGPIO[7]	21	TPUCH[7], TPU_A
24	RXD[1]/QGPI[1]	23	MPWM[3]	24	MGPIO[8]	23	TPUCH[8], TPU_A
26	RXD[2]/QGPI[2]	25	MPWM[16]	26	MGPIO[9]	25	TPUCH[9], TPU_A
28	MDA[11]	27	MPWM[17]	28	MGPIO[10]	27	TPUCH[10], TPU_A
30	MDA[12]	29	MPWM[18]	30	MGPIO[11]	29	TPUCH[11], TPU_A
32	MDA[13]	31	MPWM[19]	32	MGPIO[12]	31	TPUCH[12], TPU_A
34	MDA[14]	33		34	MGPIO[13]	33	TPUCH[13], TPU_A
36	MDA[15]	35		36	MGPIO[14]	35	TPUCH[14], TPU_A
38		37		38	MGPIO[15]	37	TPUCH[15], TPU_A
* If not used for VF and VFLS signals above.							
Logic Analyzer Pod 7		Logic Analyzer Pod 8		Logic Analyzer Pod 9		Logic Analyzer Pod 10	

MICTOR Connector 6				**MICTOR Connector 7**			
Odd		**Even**		**Odd**		**Even**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
2	NC	1	NC	2	NC	1	NC
4	NC	3	NC	4	NC	3	NC
6	$\overline{\text{T2CLK, TPU\_B}}$	5	$\overline{\text{CNTX0\_A}}$	6	$\overline{\text{ETRIG[1]}}$	5	$\overline{\text{ETRIG[2]}}$
8	TPUCH[0], TPU_B	7	CNRX0_A	8	AN[0]/ANW/PQB[0], QADC_A	7	AN[0]/ANW/PQB[0], QADC_B
10	TPUCH[1], TPU_B	9	CNTX0_B	10	AN[1]/ANW/PQB[1], QADC_A	9	AN[1]/ANW/PQB[1], QADC_B
12	TPUCH[2], TPU_B	11	CNRX0_B	12	AN[2]/ANW/PQB[2], QADC_A	11	AN[2]/ANW/PQB[2], QADC_B
14	TPUCH[3], TPU_B	13	EPEE	14	AN[3]/ANW/PQB[3], QADC_A	13	AN[3]/ANW/PQB[3], QADC_B
16	TPUCH[4], TPU_B	15	$\overline{\text{IRQ[0]}}$/SGPIOC[0]	16	AN[48]/PQB[4], QADC_A	15	AN[48]/PQB[4], QADC_B
18	TPUCH[5], TPU_B	17	$\overline{\text{IRQ[5]}}$/SGPIOC[5]/ MODCK[1]	18	AN[49]/PQB[5], QADC_A	17	AN[49]/PQB[5], QADC_B
20	TPUCH[6], TPU_B	19	$\overline{\text{IRQ[6]}}$/MODCK[2]	20	AN[50]/PQB[6], QADC_A	19	AN[50]/PQB[6], QADC_B
22	TPUCH[7], TPU_B	21	$\overline{\text{IRQ[7]}}$/MODCK[3]	22	AN[51]/PQB[7], QADC_A	21	AN[51]/PQB[7], QADC_B
24	TPUCH[8], TPU_B	23	TMS	24	AN[52]/MA[0]/ PQA[0], QADC_A	23	AN[52]/MA[0]/ PQA[0], QADC_B
26	TPUCH[9], TPU_B	25	TRST	26	AN[53]/MA[1]/ PQA[1], QADC_A	25	AN[53]/MA[1]/ PQA[1], QADC_B
28	TPUCH[10], TPU_B	27		28	AN[54]/MA[2]/ PQA[2], QADC_A	27	AN[54]/MA[2]/ PQA[2], QADC_B
30	TPUCH[11], TPU_B	29		30	AN[55]/PQA[3], QADC_A	29	AN[55]/PQA[3], QADC_B
32	TPUCH[12], TPU_B	31		32	AN[56]/PQA[4], QADC_A	31	AN[56]/PQA[4], QADC_B
34	TPUCH[13], TPU_B	33		34	AN[57]/PQA[5], QADC_A	33	AN[57]/PQA[5], QADC_B

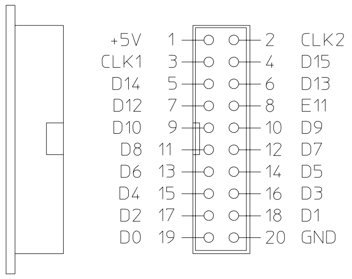
MICTOR Connector 6				**MICTOR Connector 7**			
Odd		**Even**		**Odd**		**Even**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
36	TPUCH[14], TPU_B	35		36	AN[58]/PQA[6], QADC_A	35	AN[58]/PQA[6], QADC_B
38	TPUCH[15], TPU_B	37		38	AN[59]/PQA[7], QADC_A	37	AN[59]/PQA[7], QADC_B
Logic Analyzer Pod 11		Logic Analyzer Pod 12		Logic Analyzer Pod 13		Logic Analyzer Pod 14	

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.

MPC555 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**		**Medium-Density Connector 6**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
1	NC	1	NC	1	NC	1	NC	1	NC	1	NC
2	NC	2	NC	2	NC	2	NC	2	NC	2	NC
3	CLKOUT	3	$\overline{\text{TS}}$	3	$\overline{\text{TA}}$	3	$\overline{\text{STS}}$	3	TEA	3	RD/$\overline{\text{WR}}$
4	A[16]	4	PORESET	4	D[16]	4	D[0]	4	BURST	4	CR
5	A[17]	5	KAPWR	5	D[17]	5	D[1]	5	BDIP	5	$\overline{\text{IRQ[1]}}$/$\overline{\text{RSV}}$
6	A[18]	6	EXTCLK	6	D[18]	6	D[2]	6	OE	6	RSTCONF
7	A[19]	7		7	D[19]	7	D[3]	7	WE[0]	7	IWP[0]
8	A[20]	8	CS[0]	8	D[20]	8	D[4]	8	WE[1]	8	IWP[1]
9	A[21]	9	CS[1]	9	D[21]	9	D[5]	9	$\overline{\text{WE[2]}}$	9	IWP[2]
10	A[22]	10	CS[2]	10	D[22]	10	D[6]	10	$\overline{\text{WE[3]}}$	10	IWP[3]
11	A[23]	11	CS[3]	11	D[23]	11	D[7]	11	$\overline{\text{AT[2]}}$	11	LWP[0]
12	A[24]	12	A[8]	12	D[24]	12	D[8]	12	$\overline{\text{TSIZ[0]}}$	12	LWP[1]
13	A[25]	13	A[9]	13	D[25]	13	D[9]	13	$\overline{\text{TSIZ[1]}}$	13	DSCK (Optional)
14	A[26]	14	A[10]	14	D[26]	14	D[10]	14	$\overline{\text{VFLS[0]}}$	14	DSDO (Optional)
15	A[27]	15	A[11]	15	D[27]	15	D[11]	15	VFLS[1]	15	DSDI (Optional)
16	A[28]	16	A[12]	16	D[28]	16	D[12]	16	PTR	16	Future Compression Pin
17	A[29]	17	A[13]	17	D[29]	17	D[13]	17	$\overline{\text{IRQ[3]}}$/$\overline{\text{KR}}$/RETR	17	VF[0]
18	A[30]	18	A[14]	18	D[30]	18	D[14]	18	SRESET	18	VF[1]
19	A[31]	19	A[15]	19	D[31]	19	D[15]	19	HRESET	19	VF[2]
20	GND	20	GND	20	GND	20	GND	20	GND	20	GND
Logic Analyzer Pod 1		Logic Analyzer Pod 2		Logic Analyzer Pod 3		Logic Analyzer Pod 4		Logic Analyzer Pod 5		Logic Analyzer Pod 6	

Medium-Density Connector 7		**Medium-Density Connector 8**		**Medium-Density Connector 9**		**Medium-Density Connector 10**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
1	NC	1	NC	1	NC	1	NC
2	NC	2	NC	2	NC	2	NC
3	SCK/QGPIO[6]	3	$\overline{\text{ECK}}$	3	$\overline{\text{ENGCLK/BUCLK}}$	3	$\overline{\text{T2CLK, TPU\_A}}$
4	PCS[0]/$\overline{\text{SS}}$/QGPIO[0]	4	MDA[27]	4	MGPIO[0] *	4	TPUCH[0], TPU_A
5	PCS[1]/QGPIO[1]	5	MDA[28]	5	MGPIO[1] *	5	TPUCH[1], TPU_A
6	PCS[2]/QGPIO[2]	6	MDA[29]	6	MGPIO[2] *	6	TPUCH[2], TPU_A
7	PCS[3]/QGPIO[3]	7	MDA[30]	7	MGPIO[3] *	7	TPUCH[3], TPU_A
8	MISO/QGPIO[4]	8	MDA[31]	8	MGPIO[4] *	8	TPUCH[4], TPU_A
9	MOSI/QGPIO[5]	9	MPWM[0]	9	MGPIO[5]	9	TPUCH[5], TPU_A
10	TXD[1]/QGPO[1]	10	MPWM[1]	10	MGPIO[6]	10	TPUCH[6], TPU_A
11	TXD[2]/QGPO[2]	11	MPWM[2]	11	MGPIO[7]	11	TPUCH[7], TPU_A
12	RXD[1]/QGPI[1]	12	MPWM[3]	12	MGPIO[8]	12	TPUCH[8], TPU_A
13	RXD[2]/QGPI[2]	13	MPWM[16]	13	MGPIO[9]	13	TPUCH[9], TPU_A
14	MDA[11]	14	MPWM[17]	14	MGPIO[10]	14	TPUCH[10], TPU_A
15	MDA[12]	15	MPWM[18]	15	MGPIO[11]	15	TPUCH[11], TPU_A
16	MDA[13]	16	MPWM[19]	16	MGPIO[12]	16	TPUCH[12], TPU_A
17	MDA[14]	17		17	MGPIO[13]	17	TPUCH[13], TPU_A
18	MDA[15]	18		18	MGPIO[14]	18	TPUCH[14], TPU_A
19		19		19	MGPIO[15]	19	TPUCH[15], TPU_A
20	GND	20	GND	20	GND	20	GND
* If not used for VF and VFLS signals above.							
Logic Analyzer Pod 7		Logic Analyzer Pod 8		Logic Analyzer Pod 9		Logic Analyzer Pod 10	

Medium-Density Connector 11		**Medium-Density Connector 12**		**Medium-Density Connector 13**		**Medium-Density Connector 14**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
1	NC	1	NC	1	NC	1	NC
2	NC	2	NC	2	NC	2	NC
3	$\overline{\text{T2CLK, TPU\_B}}$	3	$\overline{\text{CNTX0\_A}}$	3	$\overline{\text{ETRIG[1]}}$	3	$\overline{\text{ETRIG[2]}}$
4	TPUCH[0], TPU_B	4	CNRX0_A	4	AN[0]/ANW/PQB[0], QADC_A	4	AN[0]/ANW/PQB[0], QADC_B
5	TPUCH[1], TPU_B	5	CNTX0_B	5	AN[1]/ANW/PQB[1], QADC_A	5	AN[1]/ANW/PQB[1], QADC_B
6	TPUCH[2], TPU_B	6	CNRX0_B	6	AN[2]/ANW/PQB[2], QADC_A	6	AN[2]/ANW/PQB[2], QADC_B
7	TPUCH[3], TPU_B	7	EPEE	7	AN[3]/ANW/PQB[3], QADC_A	7	AN[3]/ANW/PQB[3], QADC_B
8	TPUCH[4], TPU_B	8	$\overline{\text{IRQ[0]}}$/SGPIOC[0]	8	AN[48]/PQB[4], QADC_A	8	AN[48]/PQB[4], QADC_B
9	TPUCH[5], TPU_B	9	$\overline{\text{IRQ[5]}}$/SGPIOC[5]/ MODCK[1]	9	AN[49]/PQB[5], QADC_A	9	AN[49]/PQB[5], QADC_B
10	TPUCH[6], TPU_B	10	$\overline{\text{IRQ[6]}}$/MODCK[2]	10	AN[50]/PQB[6], QADC_A	10	AN[50]/PQB[6], QADC_B
11	TPUCH[7], TPU_B	11	$\overline{\text{IRQ[7]}}$/MODCK[3]	11	AN[51]/PQB[7], QADC_A	11	AN[51]/PQB[7], QADC_B
12	TPUCH[8], TPU_B	12	TMS	12	AN[52]/MA[0]/ PQA[0], QADC_A	12	AN[52]/MA[0]/ PQA[0], QADC_B
13	TPUCH[9], TPU_B	13	TRST	13	AN[53]/MA[1]/ PQA[1], QADC_A	13	AN[53]/MA[1]/ PQA[1], QADC_B
14	TPUCH[10], TPU_B	14		14	AN[54]/MA[2]/ PQA[2], QADC_A	14	AN[54]/MA[2]/ PQA[2], QADC_B
15	TPUCH[11], TPU_B	15		15	AN[55]/PQA[3], QADC_A	15	AN[55]/PQA[3], QADC_B
16	TPUCH[12], TPU_B	16		16	AN[56]/PQA[4], QADC_A	16	AN[56]/PQA[4], QADC_B
17	TPUCH[13], TPU_B	17		17	AN[57]/PQA[5], QADC_A	17	AN[57]/PQA[5], QADC_B
18	TPUCH[14], TPU_B	18		18	AN[58]/PQA[6], QADC_A	18	AN[58]/PQA[6], QADC_B

Medium-Density Connector 11		**Medium-Density Connector 12**		**Medium-Density Connector 13**		**Medium-Density Connector 14**	
Pin	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**	**Pin**	**MPC555 Signal**
19	TPUCH[15], TPU_B	19		19	AN[59]/PQA[7], QADC_A	19	AN[59]/PQA[7], QADC_B
20	GND	20	GND	20	GND	20	GND
Logic Analyzer Pod 11		Logic Analyzer Pod 12		Logic Analyzer Pod 13		Logic Analyzer Pod 14	

Preparing for Emulation

When using the MPC5XX emulation module, you need to be aware of the requirements it makes of target systems, and you need to consider how and when the emulation module connects to the target system.

Target System Requirements

The DSDI and DSCK signals must not be actively driven by the target system when the debug port is being used.

The $\overline{\text{RESET}}$ and $\overline{\text{SRESET}}$ signals from the debug connector must be ORed with the respective $\overline{\text{RESET}}$ and $\overline{\text{SRESET}}$ signals that connect to the processor on the target system. They can be logically ORed or "wire-ORed" on the board. The emulation module drives $\overline{\text{RESET}}$ and $\overline{\text{SRESET}}$ through 100 Ohm resistors with an open-collector drivers. There is also 1.79 kOhm pullup to 3.3 volts on the $\overline{\text{RESET}}$ and $\overline{\text{SRESET}}$ lines.

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

Additional target requirements may be specified in the release notes in the "readme" file on the provided floppy disk.

Debug Port Connection

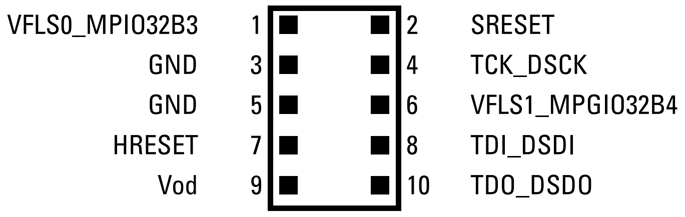
The emulation module requires a debug port (BDM) connector in the target system.

The connector should be a dual row header strip ("Berg connector"), 10 pins per inch, with 25 mil pins.

There are three possible pin outs of the BDM connector for the MPC555. While these can be chosen based on the application, there are preferred pin outs for specific applications.

MPC555 Debug Port Connector, Option 1

For maximum debug capability (access to BDM and program trace signals):



The E3497-66502 target interface module (TIM) requires 10 kOhm pull-up resistors on pins 1 and 6.

The E3497-66503 target interface module (TIM) has 10 kOhm pull-up resistors on pins 1 and 6.

Use the following commands to configure the emulation module for this configuration:

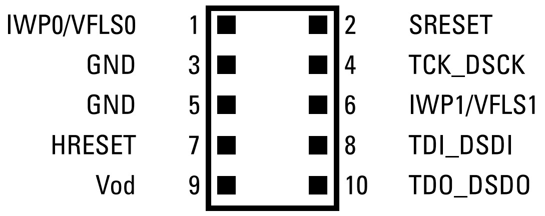
```
cf proc=MPC555
cf dbgconfig=1
rst -m
```

NOTE: The MIOS1TPCR (0x0030 6800) register is modified to configure J18 and K18 as VFLS0/1 pins. Software must not modify this register.

Option 1 is recommended because it leaves Program Trace pins for full analysis support. The other two options sacrifice watchpoint pins or program trace pins, thus forcing you to trade off analysis features.

MPC555 Debug Port Connector, Option 2

For maximum external bus capability:



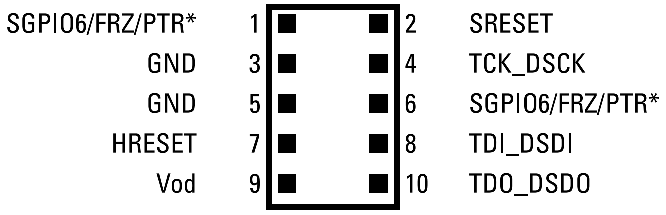
The E3497-66502 (as well as the E3497-66503) target interface module (TIM) is compatible with the MPC555 processor debug port.

Use the following commands to configure the emulation module for this configuration:

```
cf proc=MPC555
cf dbgconfig=2 (default setting)
rst -m
```

MPC555 Debug Port Connector, Option 3

For maximum I/O configuration:



The E3497-66502 target interface module (TIM) requires 10 kOhm pull-up resistors on pins 1 and 6.

The E3497-66503 target interface module (TIM) has 10 kOhm pull-up resistors on pins 1 and 6.

Bit 13 of the SIUMCR (0x002fc000) is set to enable FRZ. Software must not change this bit.

Use the following commands to configure the emulation module for this configuration:

```
cf proc=MPC555
cf dbgconfig=3
rst -m
```

On-Chip Flash Support

The emulation module will not directly support on-chip flash. Flash support should be provided by 3rd party debug vendors.

Fast Download

The Agilent Technologies emulation module will automatically use the chip's internally supported fast download mode.

The Software Watchdog Timer

The MPC555 includes a SWT (Software Watchdog Timer) resource. Each time the target processor is reset the SWT is returned to a default configuration which includes:

- The system clock is pre-scaled by 2048.
- The SWT counter decrements beginning at 0xFFFF.
- If the SWT counter reaches zero the SWT issues a system reset.

A software service routine that prevents the counter from reaching zero must be included in the target system initialization code. If the counter does reach zero, a system reset is issued by the SWT.

Target system initialization firmware (boot code) typically writes to the SYPCR (System Protection Control Register) immediately upon processor reset. This sets the SWT to a desired configuration. The SWT can be disabled, frozen, or pre-scaled; its time-out period can be changed, and its time-out behavior can be specified through the SYPCR.

When using the emulation probe, special consideration must be given to controlling the SWT in order to prevent repeated SWT resets. Typically, when a debugger connects to a target system through an emulation probe, the probe resets the processor and then places the processor in debug mode. If the SYPCR is not configured immediately after entering debug mode, the SWT will time out and reset the processor. In order to prevent this from occurring, the emulation probe always writes to the SYPCR when it goes from reset to debug mode. The default write value is 0xffffff88. This sets the SYPCR SWE (Software Watchdog Enable) bit to false, disabling the SWT. This prevents the SWT from counting down and resetting the processor.

NOTE: The SYPCR is a write-once register. Once written it cannot be changed until the processor is reset.

Because the SYPCR is a write-once register, and because the emulation probe writes to the register, some mechanism must be provided for the user to specify the SYPR value the probe will write.

The emulation probe's cf_sypcr register value can be changed by issuing the following command to the emulation probe:

```
reg cf_sypcr=0xXXXXXXXX
```

Any value chosen for the cf_sypcr register will be logically ORed with 0x00000008. This sets the SWF bit which stops the watchdog timer whenever the processor is in debug mode.

If you don't want the SWF bit set in the SYPCR register, the SYPCR must be written to first by system initialization firmware. The initialization code must be executed before entering debug mode.

There are two ways to do this:

- Manually reset the target processor
- Select Run from Reset using the debugger. (This presumes that the debugger issues the emulation probe commands **`rst -r`** or **`rst`** followed by **`r`**.) The emulation probe resets and runs the processor without entering debug mode and writing the SYPCR.

See Also Consult the *Motorola MPC555 / MPC556 User's Manual* for more information about the Software Watchdog Timer (SWT) and the System Protection Control Register (SYPCR).

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 16700-series logic analysis systems

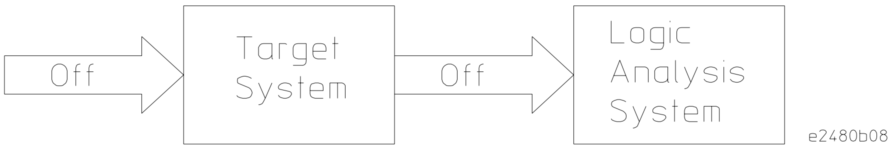
Ensure the target system is powered off.

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

To power-OFF

Turn off power to your system in the following order:

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



Installing Logic Analyzer Modules

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

Refer to the Agilent Technologies 16700-series logic analysis system's *Installation Guide*.

Installing the Emulation Module

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

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

To install in a 16700-series logic analysis system

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

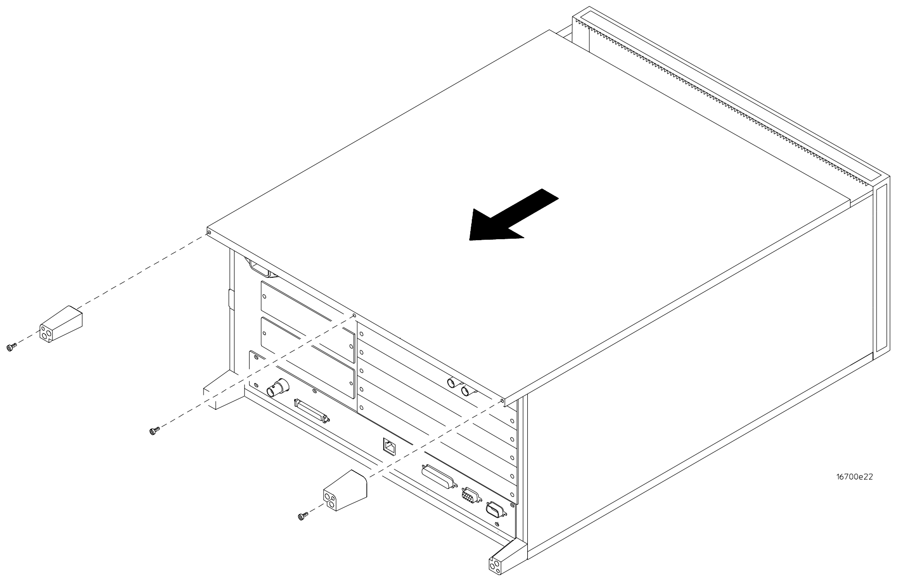
You will need T-10 and T-15 Torx screw drivers.

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

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

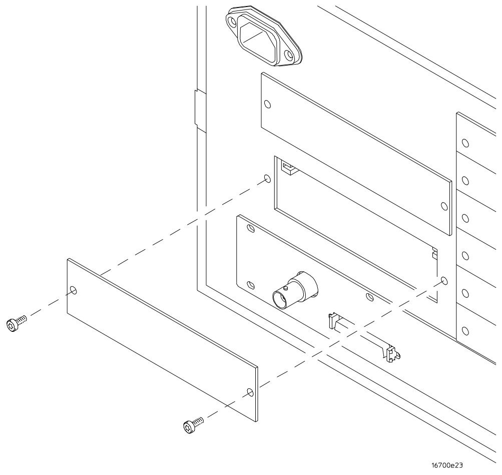
2 Turn the logic analysis system frame upside-down.

3 Remove the bottom cover.

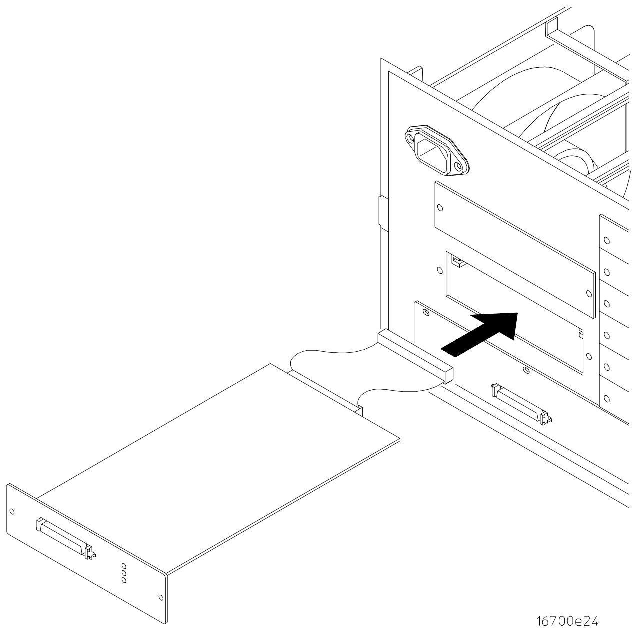


4 Remove the slot cover.

You may use either slot.

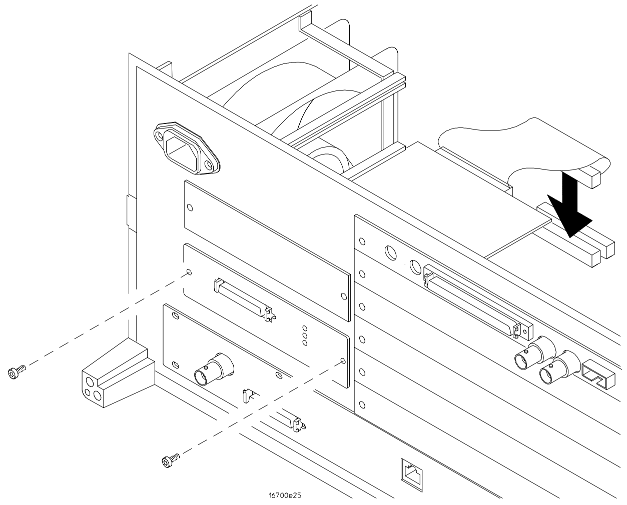


5 Install the emulation module.



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 76 for information on giving the emulation module a “personality” for your target processor.

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 Name ____
- Gateway Address ____
- Subnet Mask ____
- 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 16700-series logic analysis system)
6474	Slot 2 (Second emulation module in an Agilent Technologies 16700-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 16700-series logic analysis system)
6476	Slot 2 (Second emulation module in an Agilent Technologies 16700-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 16700-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 test the emulation module

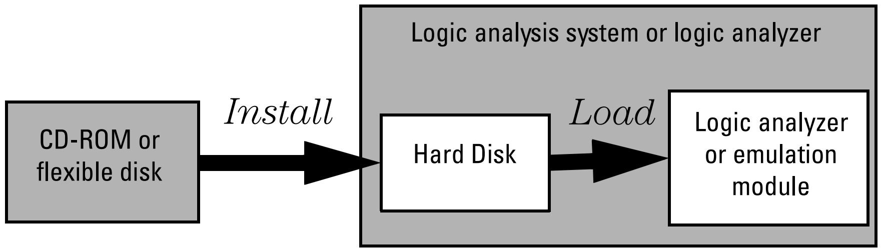
If this is the first time that you have used the emulation module, you should run the built-in performance verification tests before you connect to a target system. Refer to Chapter 12, “Troubleshooting the Emulation Module,” on page 175 for information on performance verification.

Installing Software

This section 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.


Logic analysis system or logic analyzer
CD-ROM or flexible disk
Install
Hard Disk
Load
Logic analyzer or emulation 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 the software from CD-ROM

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

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

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

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

2 Insert the CD-ROM in the drive.

3 Select the System Administration icon.

4 Select the Software Install tab.

5 Select Install....

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

6 Select Apply.

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

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

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

8 Select the "MPC5XX" package.

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

9 Select Install.

The Continue dialog box will appear.

10 Select Continue.

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

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

The configuration files are stored in /logic/configs/hp/mpc5xx. The inverse assemblers are stored in /logic/ia.

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

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

To list software packages which are installed

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

4

Probing the Target System

Using the Setup Assistant

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

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

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

Connecting the Logic Analyzer to the Target System

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

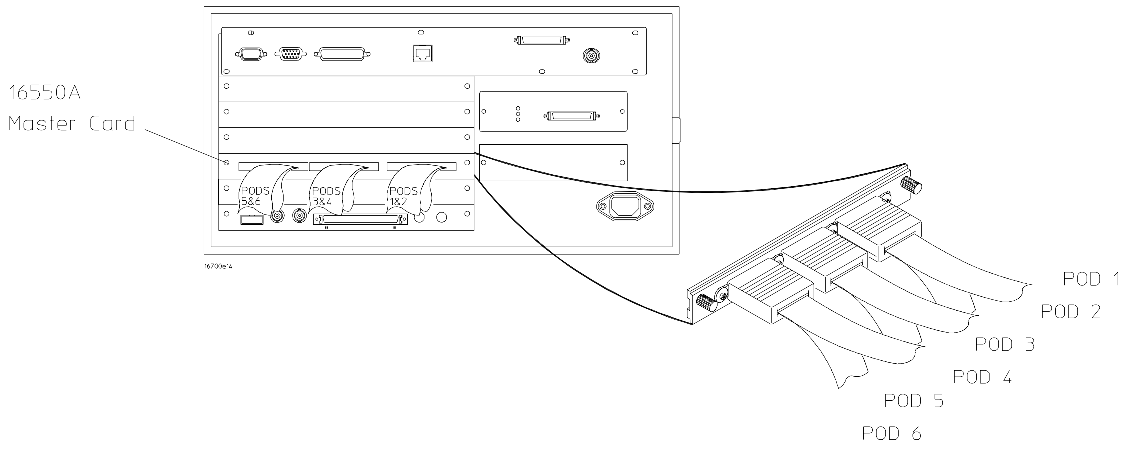
This section identifies connections to each Agilent Technologies logic analyzer supported by the inverse assembler. They are shown in the following order:

- 16550A logic analyzer (one or two cards).
- 16554/55 logic analyzers (two or three cards).
- 16556/57 logic analyzers (two, three, or four cards).
- 16710/11/12A logic analyzers (one or two cards).
- 16715/16/17/18/19A logic analyzers (two, three, or four cards)
- 16750/51/52A logic analyzers (two, three, or four cards)

Number of Pods Used/Required

The MPC555 inverse assembler requires a minimum of six pods. The logic analyzer configuration files assign signals for 14 pods. If fewer than 14 pods are used, only configuration file definitions for the available pods will be used.

To connect a one-card 16550A logic analyzer



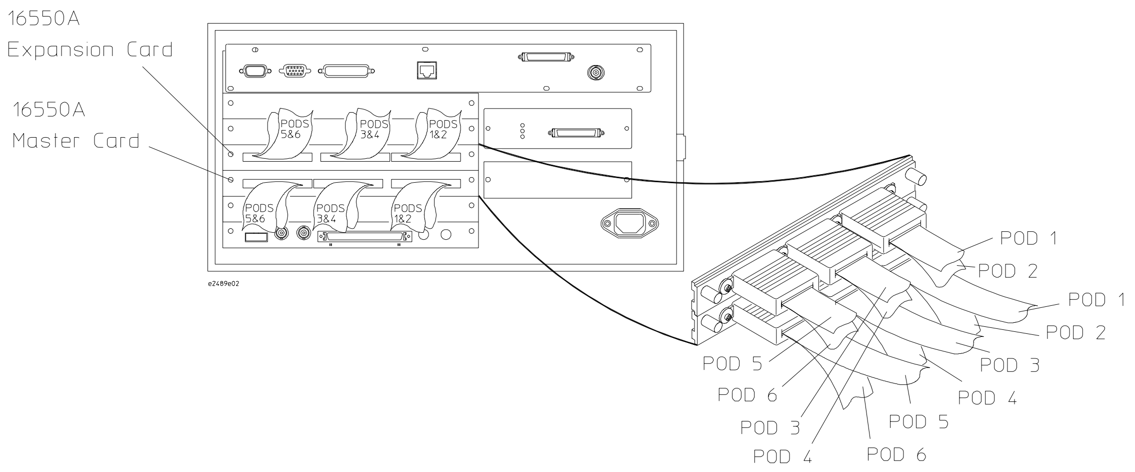
Use this table to connect cables from the Agilent Technologies 16550A logic analyzer to the connectors in the target system.

One-Card Agilent Technologies 16550A Logic Analyzer Pod to Connector Mapping

		Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
Master Card	**High-density**	3, Even	3, Odd	2, Even	2, Odd	1, Even	1, Odd
	Med-density	6	5	4	3	2	1

Configuration File for the One-Card Agilent Technologies 16550A Logic Analyzer: Use configuration file C555IA1.

To connect a two-card 16550A logic analyzer

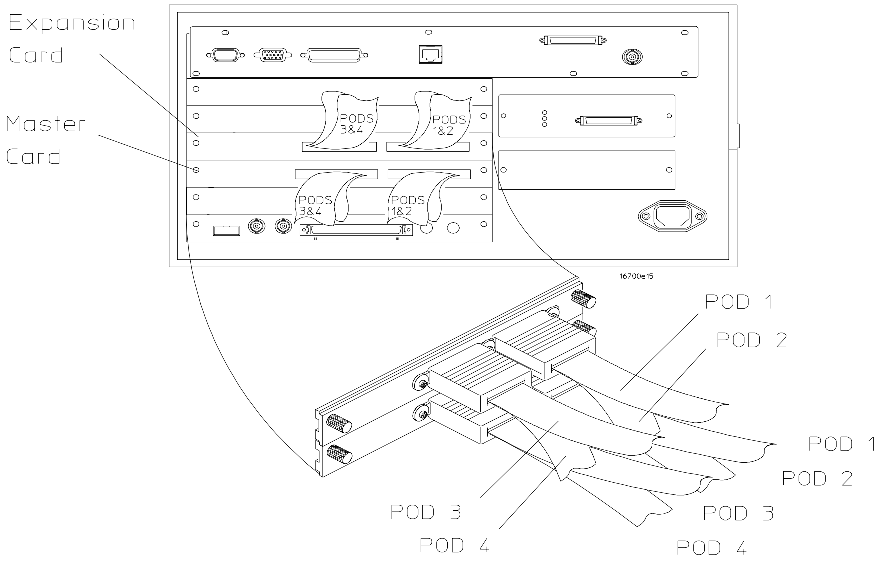


Use this table to connect cables from the Agilent Technologies 16550A logic analyzer to the connectors in the target system.

		Pod 6	**Pod 5**	**Pod 4**	**Pod 3**	**Pod 2**	**Pod 1**
Expansion Card 1	**High-density**	6, Even	6, Odd	5, Even	5, Odd	4, Even	4, Odd
	Med-density	12	11	10	9	8	7
Master Card	**High-density**	3, Even	3, Odd	2, Even	2, Odd	1, Even	1, Odd
	Med-density	6	5	4	3	2	1

Configuration File for the Two-Card Agilent Technologies 16550A Logic Analyzer: Use configuration file C555IA2.

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

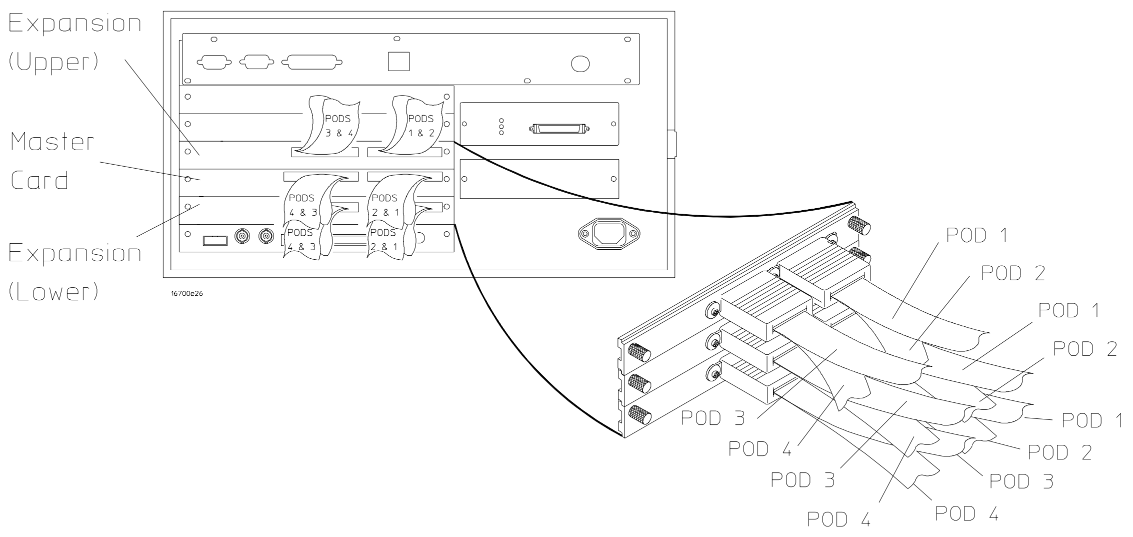


Use this table to connect cables from the two-card Agilent Technologies 16554/55/56/57 logic analyzer to the connectors in the target system.

		Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	**High-density**	4, Even	4, Odd	3, Even	3, Odd
	Med-density	8	7	6	5
Master Card	**High-density**	2, Even	2, Odd	1, Even	1, Odd
	Med-density	4	3	2	1

Configuration File for the Two-Card Agilent Technologies 1655455/56/57 Logic Analyzers:
Use configuration file C555IA3.

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

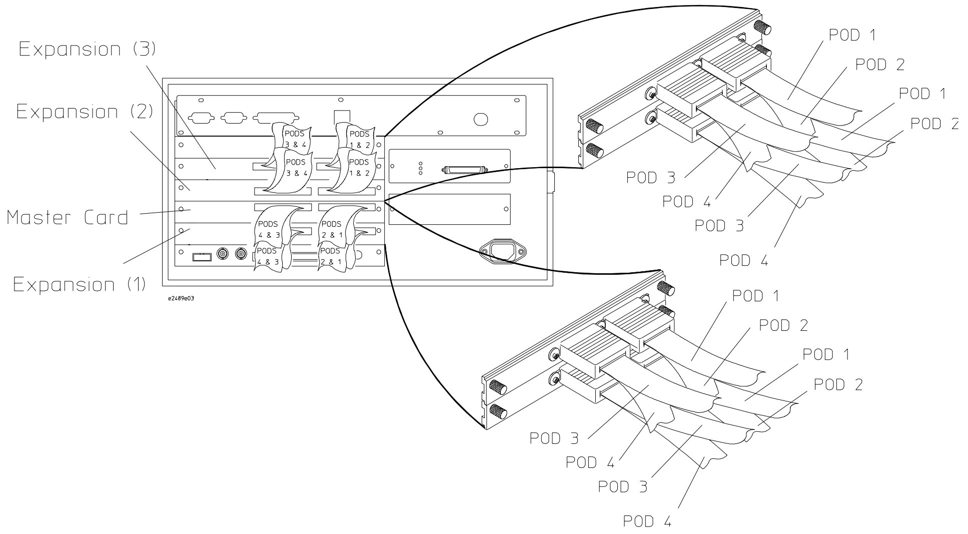


Use this table to connect cables from the three-card Agilent Technologies 16554/55/56/57 logic analyzer to the connectors in the target system.

		Pod 4	**Pod 3**	**Pod 2**	**Pod 1**
Expansion Card 2	**High-density**	6, Even	6, Odd	5, Even	5, Odd
	Med-density	12	11	10	9
Master Card	**High-density**	2, Even	2, Odd	1, Even	1, Odd
	Med-density	4	3	2	1
Expansion Card 1	**High-density**	4, Even	4, Odd	3, Even	3, Odd
	Med-density	8	7	6	5

Configuration File for the Three-Card Agilent Technologies 1655455/56/57 Logic Analyzers:
Use configuration file C555IA4.

To connect a four-card 16556/57 logic analyzer

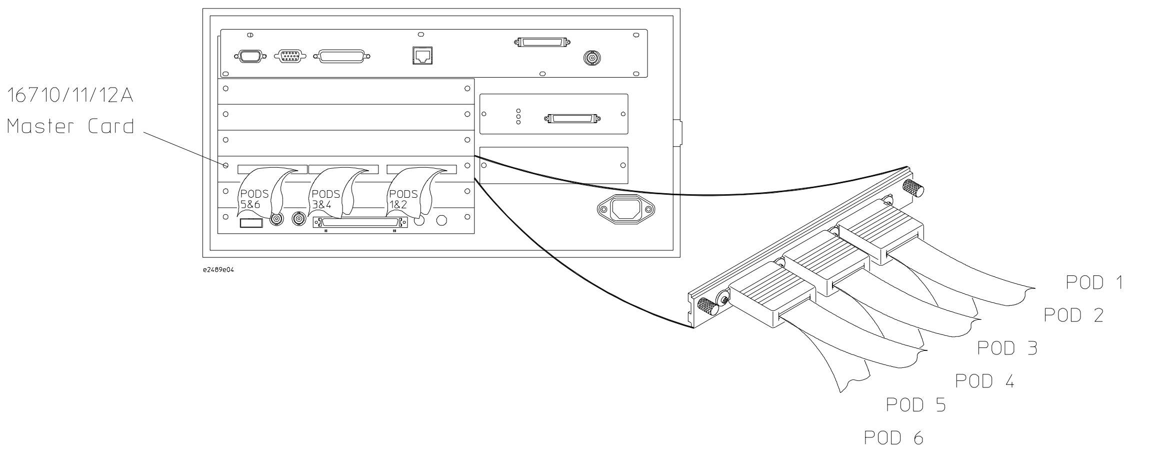


Use this table to connect cables from the four-card Agilent Technologies 16556/57 logic analyzer to the connectors in the target system.

		Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 3	**High-density**	Not Used	Not Used	7, Even	7, Odd
	Med-density	Not Used	Not Used	14	13
Expansion Card 2	**High-density**	6, Even	6, Odd	5, Even	5, Odd
	Med-density	12	11	10	9
Master Card	**High-density**	2, Even	2, Odd	1, Even	1, Odd
	Med-density	4	3	2	1
Expansion Card 1	**High-density**	4, Even	4, Odd	3, Even	3, Odd
	Med-density	8	7	6	5

Configuration File for the Four-Card Agilent Technologies 16556/57 Logic Analyzers:
Use configuration file C555IA4.

To connect a one-card 16710/11/12A logic analyzer

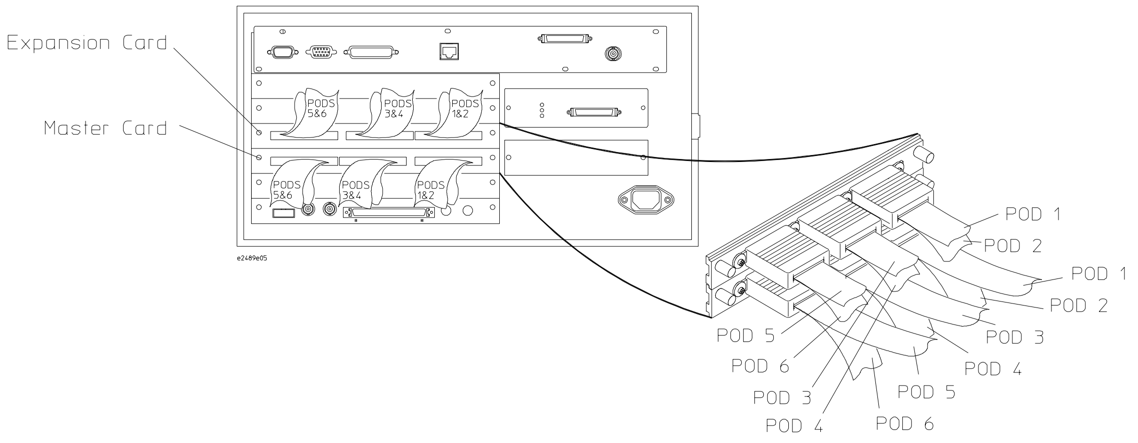


Use this table to connect cables from the Agilent Technologies 16710/11/12A logic analyzers to the connectors in the target system.

		Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
Master Card	**High-density**	3, Even	3, Odd	2, Even	2, Odd	1, Even	1, Odd
	Med-density	6	5	4	3	2	1

Configuration File for the One-Card Agilent Technologies 16710/11/12A Logic Analyzers:
Use configuration file C555IA1.

To connect a two-card 16710/11/12A logic analyzer

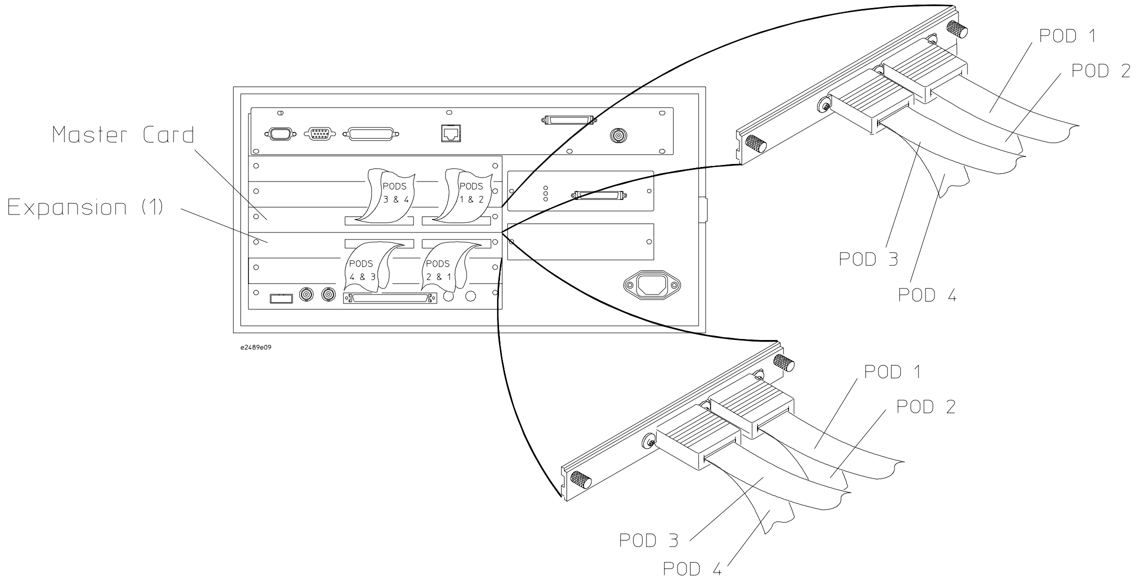


Use this table to connect cables from the Agilent Technologies 16710/11/12A logic analyzers to the connectors in the target system.

		Pod 6	Pod 5	Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 1	**High-density**			5, Even	5, Odd	4, Even	4, Odd
	Med-density	12	11	10	9	8	7
Master Card	**High-density**	3, Even	3, Odd	2, Even	2, Odd	1, Even	1, Odd
	Med-density	6	5	4	3	2	1

Configuration File for the Two-Card Agilent Technologies 16710/11/12A Logic Analyzers:
Use configuration file C555IA2.

To connect a two-card 16715/16/17/18/19/50/51/52A logic analyzer

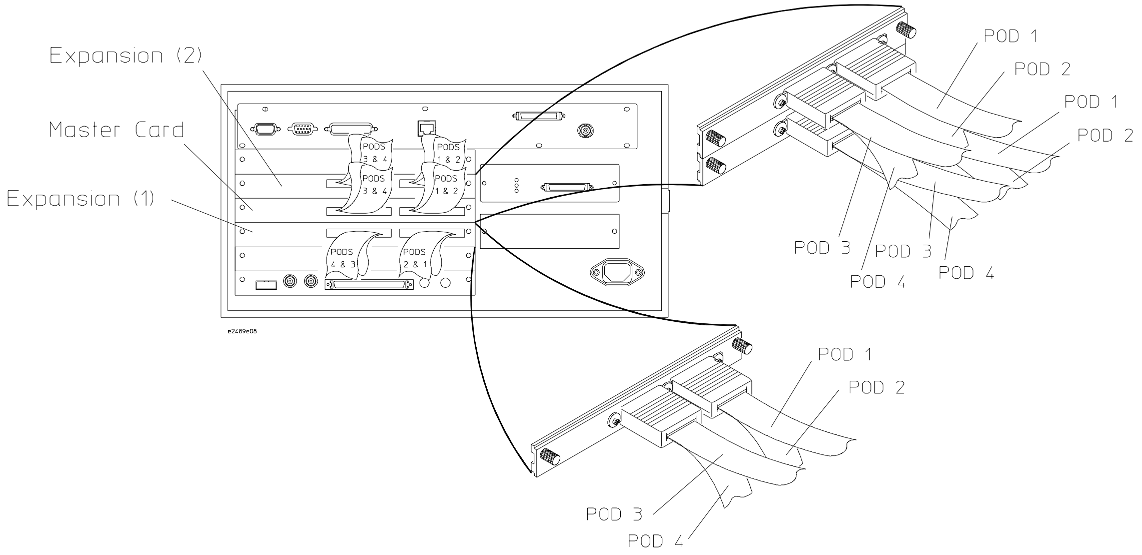


Use this table to connect cables from the Agilent Technologies 16715/16/17/18/19/50/51/52A logic analyzers to the connectors in the target system.

		Pod 4	**Pod 3**	**Pod 2**	**Pod 1**
Master Card	**High-density**	2, Even	2, Odd	1, Even	1, Odd
	Med-density	4	3	2	1
Expansion Card	**High-density**	4, Even	4, Odd	3, Even	3, Odd
	Med-density	8	7	6	5

Configuration File for the Two-Card Agilent Technologies 16715/16/17/18/19/50/51/52A Logic Analyzers:
Use configuration file C555IA3.

To connect a three-card 16715/16/17/18/19/50/51/52A logic analyzer

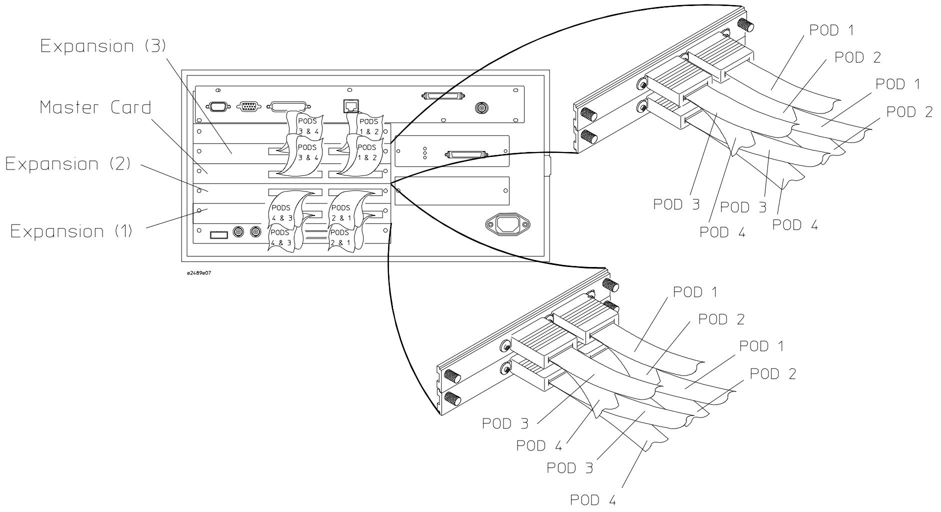


Use this table to connect cables from the Agilent Technologies 16715/16/17/18/19/50/51/52A logic analyzers to the connectors in the target system.

		Pod 4	**Pod 3**	**Pod 2**	**Pod 1**
Expansion Card 2	**High-density**	6, Even	6, Odd	5, Even	5, Odd
	Med-density	12	11	10	9
Master Card	**High-density**	2, Even	2, Odd	1, Even	1, Odd
	Med-density	4	3	2	1
Expansion Card 1	**High-density**	4, Even	4, Odd	3, Even	3, Odd
	Med-density	8	7	6	5

Configuration File for the Three-Card Agilent Technologies 16715/16/17/18/19/50/51/52A Logic Analyzers:
Use configuration file C555IA4.

To connect a four-card 16715/16/17/18/19/50/51/52A logic analyzer



Use this table to connect cables from the Agilent Technologies 16715/16/17/18/19/50/51/52A logic analyzers to the connectors in the target system.

		Pod 4	Pod 3	Pod 2	Pod 1
Expansion Card 3	**High-density**	Not Used	Not Used	7, Even	7, Odd
	Med-density	Not Used	Not Used	14	13
Master Card	**High-density**	2, Even	2, Odd	1, Even	1, Odd
	Med-density	4	3	2	1
Expansion Card 2	**High-density**	6, Even	6, Odd	5, Even	5, Odd
	Med-density	12	11	10	9
Expansion Card 1	**High-density**	4, Even	4, Odd	3, Even	3, Odd
	Med-density	8	7	6	5

Configuration File for the Four-Card Agilent Technologies 16715/16/17/18/19/50/51/52A Logic Analyzers:
Use configuration file C555IA4.

Connecting the Emulation Module to the Target System

This section shows you how to connect the emulation module to the debug port connector on the target board. 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 debug port on your target board, see "Preparing for Emulation" on page 42.

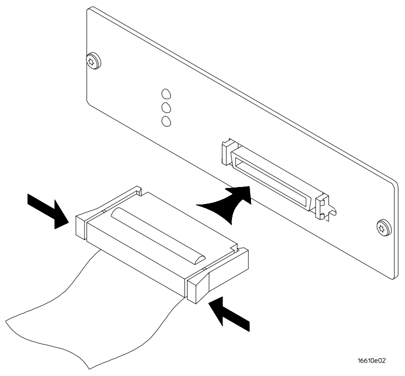
For a list of the parts supplied with the emulation module, see "Emulation Module" on page 20.

To connect to a target system debug port

The emulation module can be connected to a target system through a 10-pin debug port (BDM connector). The emulation module should be connected to a 10-pin male 2x5 header connector on the target system using the 10-conductor cable assembly provided.

1 Turn off the target system and disconnect it from all power sources.

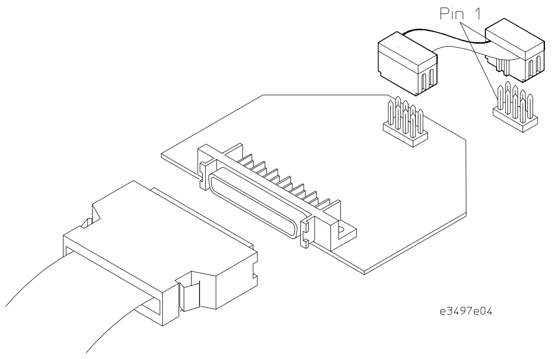
2 Plug one end of the 50-pin cable into the emulation module.

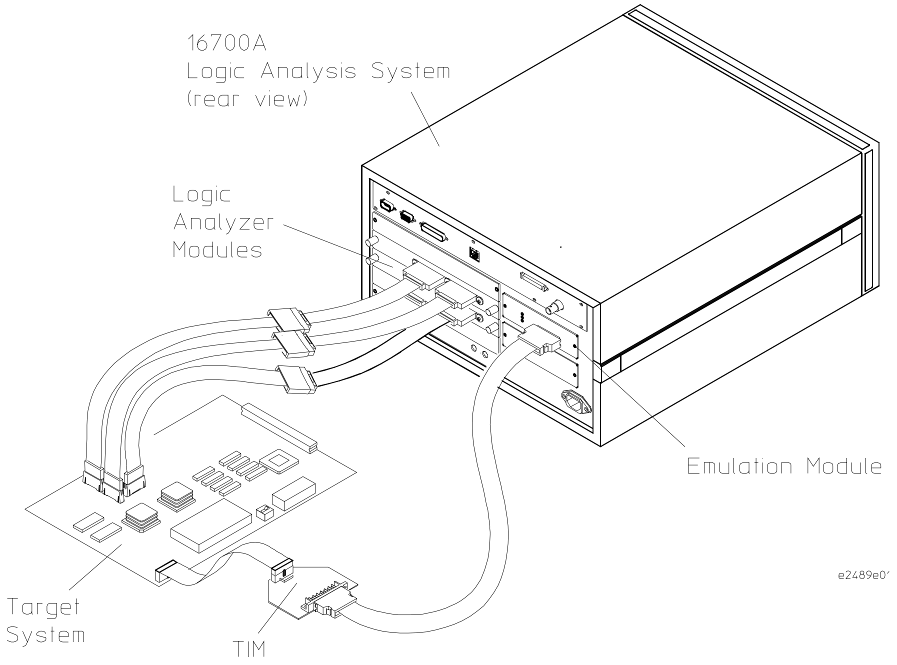


3 Plug the other end of the 50-pin cable into the target interface module.

4 Plug one end of the 10-pin cable into the target interface module.

5 Plug the other end of the 10-pin cable into the debug port on the target system.





6 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 42.

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.

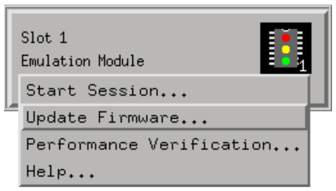
To update the firmware:

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 57.

To display the emulation module firmware version information

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

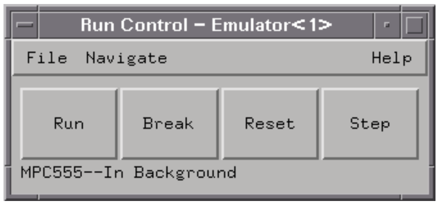
There are usually two firmware version numbers: one for "Generics" and one for the personality of your processor.

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:



```
MPC555--In Background
```

Part 2

Using the Logic Analyzer

5

Configuring the Logic Analyzer

Loading Configuration Files

If you used the Setup Assistant for connecting and configuring the logic analysis system, the proper configuration files and inverse assembler will already be loaded. You still have to set the inverse assembler preferences because their values are based on your target system design.

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.

There are several MPC555 configuration files. The configuration file you use depends on which logic analyzer model you are using and whether you are using state or timing analysis mode.

The MPC555 inverse assembler decodes data captured from external bus fetches into instruction pointer (IP) addresses (also known as software addresses) and assembly language mnemonics.

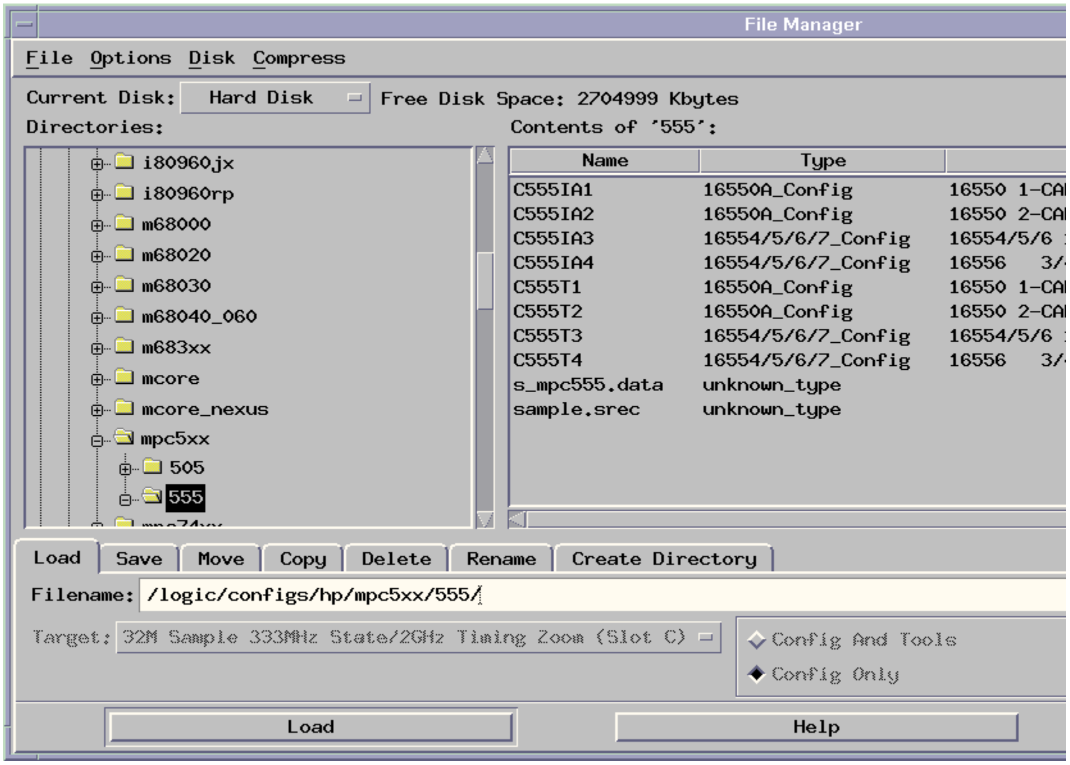
NOTE: The trace reconstruction tool can be used in conjunction with the inverse assembler to provide complete program execution trace, including trace of instructions which are executed out of internal memory (cache). This tool is not automatically loaded when you load the configuration file. For information on using the trace reconstruction tool see page 112.

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/mpc5xx/ exists.

> If the above directory does not exist, you need to install the MPC5XX Processor Support Package. Close File Manager, then use the procedure on the CD-ROM jacket to install the MPC5XX Processor Support Package before you continue.


File Manager
File Options Disk Compress
Load Save Move Copy Delete Rename Create Directory
Load
Help

Current Disk: Hard Disk Free Disk Space: 2704999 Kbytes

Directories:

```
i80960jx
i80960rp
m68000
m68020
m68030
m68040_060
m683xx
mcore
mcore_nexus
mpc5xx
  505
  555
```

Contents of '555':

Name	Type	
C555IA1	16550A_Config	16550 1-CA
C555IA2	16550A_Config	16550 2-CA
C555IA3	16554/5/6/7_Config	16554/5/6
C555IA4	16554/5/6/7_Config	16556 3/
C555T1	16550A_Config	16550 1-CA
C555T2	16550A_Config	16550 2-CA
C555T3	16554/5/6/7_Config	16554/5/6
C555T4	16554/5/6/7_Config	16556 3/
s_mpc555.data	unknown_type	
sample.srec	unknown_type	

Filename: /logic/configs/hp/mpc5xx/555/

Target: 32M Sample 333MHz State/2GHz Timing Zoom (Slot C)

◇ Config And Tools
◆ Config Only

2 Using File Manager, select the configuration file you want to load in the /logic/configs/hp/mpc555/ 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 MPC555 analysis by loading the appropriate MPC555 configuration file. Loading state 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)	State Configuration File	Timing Configuration File
16550A (one or two cards)	100 MHz state 250/500 MHz timing 4K/8K samples	C555IA1 (one card) C555IA2 (two cards)	C555T1 (one card) C555T2 (two cards)
16554A (two or three cards)	70 MHz state 125/250 MHz timing 0.5M/1M samples	C555IA3 (two cards) C555IA4 (three cards)	C555T3 (two cards) C555T4 (three cards)
16555A/D (two or three cards)	110 MHz state 250/500 MHz timing 2M/4M samples (D)	C555IA3 (two cards) C555IA4 (three cards)	C555T3 (two cards) C555T4 (three cards)
16556A/D (two, three, or four cards)	100 MHz state 200/400 MHz timing 2M/4M samples (D)	C555IA3 (two cards) C555IA4 (three cards)	C555T3 (two cards) C555T4 (three cards)
16557D (two, three, or four cards)	135 MHz state 250/500 MHz timing 2M/4M samples	C555IA3 (two cards) C555IA4 (three or four cards)	C555IA3 (two cards) C555T4 (three or four cards)
16710A (one or two cards)	100 MHz state 250/500 MHz timing 8K/16K samples	C555IA1 (one card) C555IA2 (two cards)	C555T1 (one card) C555T2 (two cards)
16711A (one or two cards)	100 MHz state 250/500 MHz timing 32K/64K samples	C555IA1 (one card) C555IA2 (two cards)	C555T1 (one card) C555T2 (two cards)
16712A (one or two cards)	100 MHz state 250/500 MHz timing 128K/256K samples	C555IA1 (one card) C555IA2 (two cards)	C555T1 (one card) C555T2 (two cards)

The following table describes the labels that are defined by the configuration files.

Configuration File Label Descriptions

Label	MPC555 Signals	Description
ADDR	A[31:8]	Address bus
AT2	$\overline{\text{IRQ[4]}}$/AT[2]/SGPIOC[4]	Address type
BDIP	BDIP	Burst data in progress
BI	$\overline{\text{BI}}$/$\overline{\text{STS}}$	Burst inhibit
BUCLK	$\overline{\text{ENGCLK}}$/$\overline{\text{BUCLK}}$	Backup clock
BURST	BURST	Burst indicator
CLKOUT	CLKOUT	Clock system frequency output
CNRX0A	CNRX0_A	TOUCAN receive data, first CAN
CNRX0B	CNRX0_B	TOUCAN receive data, second CAN
CNTX0A	CNTX0_A	TOUCAN transmit data 0, first CAN
CNTX0B	CNTX0_B	TOUCAN transmit data 0, second CAN
CR	$\overline{\text{IRQ[2]}}$/$\overline{\text{CR}}$/SGPIOC[2]/ $\overline{\text{MTS}}$	Cancel reservation
CS0-3	CS[3:0]	Chip selects
DATA	D[31:0]	Data bus
DSCK	TCK/DSCK	Development serial clock
DSDI	TDI/DSDI	Development serial data input
DSDO	TDO/DSDO	Development serial data output
ECK	ECK	External baud clock
ENGCLK	$\overline{\text{ENGCLK}}$/$\overline{\text{BUCLK}}$	Engineering clock output
EPEE	EPEE	Externally controls program/erase operations
EXTCLK	EXTCLK	External frequency source for the chip
FREEZE	SGPIOC[6]/FRZ/$\overline{\text{PTR}}$	Indicates that the RCPU is in debug mode
GPI	RXD[2:1]/QGPI[2:1]	QSCI general-purpose inputs
GPIO05	QGPIO[5:0]	QSPI general-purpose input/outputs
GPIO6	SCK/QGPIO[6]	QSPI general-purpose input/output
GPO	TXD[2:1]/QGPO[2:1]	QSCI general-purpose outputs
HRESET	HRESET	Hard reset

Configuration File Label Descriptions

Label	MPC555 Signals	Description
IRQ0	$\overline{\text{IRQ[[0]}}$/SGPIOC[0]	Interrupt request 0
IRQ1	$\overline{\text{IRQ[1]}}$/$\overline{\text{RSV}}$	Interrupt request 1
IRQ3	$\overline{\text{IRQ[3]}}$/$\overline{\text{KR}}$/RETR	Interrupt request 3
IRQ5	$\overline{\text{IRQ[[5]}}$/SGPIOC[5]/ MODCK[1]	Interrupt request 5
IRQ6	$\overline{\text{IRQ[[6]}}$/MODCK[2]	Interrupt request 6
IRQ7	$\overline{\text{IRQ[[7]}}$/MODCK[3]	Interrupt request 7
IWP0-3	IWP[3:0]	Instruction watchpoints
KAPWR	KAPWR	Keep alive power
KR	$\overline{\text{IRQ[3]}}$/$\overline{\text{KR}}$/RETR	Kill reservation
LWP0-1	LWP[1:0]	Load/store watchpoint
MDA	MDA[15:11], MDA[31:27]	Double action
MISO	MISO/QGPIO[4]	Master-in slave-out
MODCK1	$\overline{\text{IRQ[[5]}}$/SGPIOC[5]/ MODCK[1]	Mode clock 1
MODCK2	$\overline{\text{IRQ[[6]}}$/MODCK[2]	Mode clock 2
MODCK3	$\overline{\text{IRQ[[7]}}$/MODCK[3]	Mode clock 3
MOSI	MOSI/QGPIO[5]	Master-out slave-in
MPIO	MGPIO[2:0]	Mios general-purpose I/O
MPIO34	MGPIO[4:3]	Mios general-purpose I/O
OE	OE	Output enable
PCS0-3	PCS[3:0]/QGPIO[3:0]	QSPI peripheral chip selects
PORESE	PORESET	Power on reset
PTR	SGPIOC[6]/FRZ/$\overline{\text{PTR}}$	Program trace
PWM03	MPWM[3:0]	Pulse width modulation
PWM169	MPWM[19:16]	Pulse width modulation
RD/WR	RD/$\overline{\text{WR}}$	Read/write
RETRY	$\overline{\text{IRQ[3]}}$/$\overline{\text{KR}}$/RETR	Retry
RSTCON	RSTCONF	Reset configuration
RSV	$\overline{\text{IRQ[1]}}$/RSV_B/SGPIOC[1]	Reservation
RXD	RXD[2:1]/QGPI[2:1]	Receive data

Configuration File Label Descriptions

Label	MPC555 Signals	Description
SCK	SCK/QGPIO[6]	QSPI clock
SGPIO0	$\overline{\text{IRQ[[0]}}$/SGPIOC[0]	General-purpose input/outputs
SGPIO5	$\overline{\text{IRQ[[5]}}$/SGPIOC[5]/ MODCK[1]	General-purpose input/outputs
SRESET	SRESET	Soft reset
STAT		This label is a collection of processor control signals.
STS	STS	Special transfer start
T2CLKA	T2CLK, TPU_A	Clocks the timer count register, first TPU
T2CLKB	T2CLK, TPU_B	Clocks the timer count register, second TPU
TA	TA	Transfer acknowledge
TCK	TCK/DSCK	Test clock
TDI	TDI/DSDI	Test data in
TDO	TDO/DSDO	Test data out
TEA	TEA	Transfer error acknowledge
TMS	TMS	Test mode select
TPU		
TPU_A	TPUCH[15:0], TPU_A	First TPU
TPU_B	TPUCH[15:0], TPU_B	Second TPU
TRST	TRST	Test reset
TS	TS	Transfer start
TSIZ01	TSIZ[1:0]	Transfer size
VF0-2	VF[2:0]	Visible instruction queue flush status
VFLS	VFLS[1:0]	Visible history buffer flush status
WE	WE[3:0]	Write enable

To set the inverse assembler preferences

The Preferences dialog lets you specify an address range for memory banks 0 through 3, specify the bus size (for burst cycles), and specify how the IRQ4/AT2/SGPIOC4 pin is used.

Set Bus Size For Burst Cycles—Memory Bank Settings

Begin Address, Ending Address. The Begin Address and Ending Address fields specify the range of addresses to be designated as Memory Bank 0, Memory Bank 1, etc. The Memory Banks are used in conjunction with the Filter Dialog to apply different colors to different memory ranges in the listing window to make the listing easier to interpret. (See “To use the inverse assembler filters” on page 119.) You can designate memory ranges which overlap if desired.

Bus Size. The TSIZ pins are invalid during a burst cycle, and therefore are not looked at by the inverse assembler. Use the “Bus Size” setting to tell the inverse assembler the size of the target system’s bus so the inverse assembler can use this information during burst cycles.

If the inverse assembler encounters a burst cycle and the Bus Size is set to “Non-Burst,” an error message will be displayed in the Listing window asking the user to check the setting in the user’s preferences menu (Preferences dialog).

Data Type. Use the “Data Type” setting to select whether instructions or data will be in each memory bank.

Chip Select Information

The chip select information is only used when the MPC555 reconstruction tool is on the Workspace.

The MPC555 internal address bus is 32-bits wide, but only the lower 24-bits are routed to external pins. The chip select [0:3] information is used to recreate a software address from the 24 external address lines plus the four chip select lines.

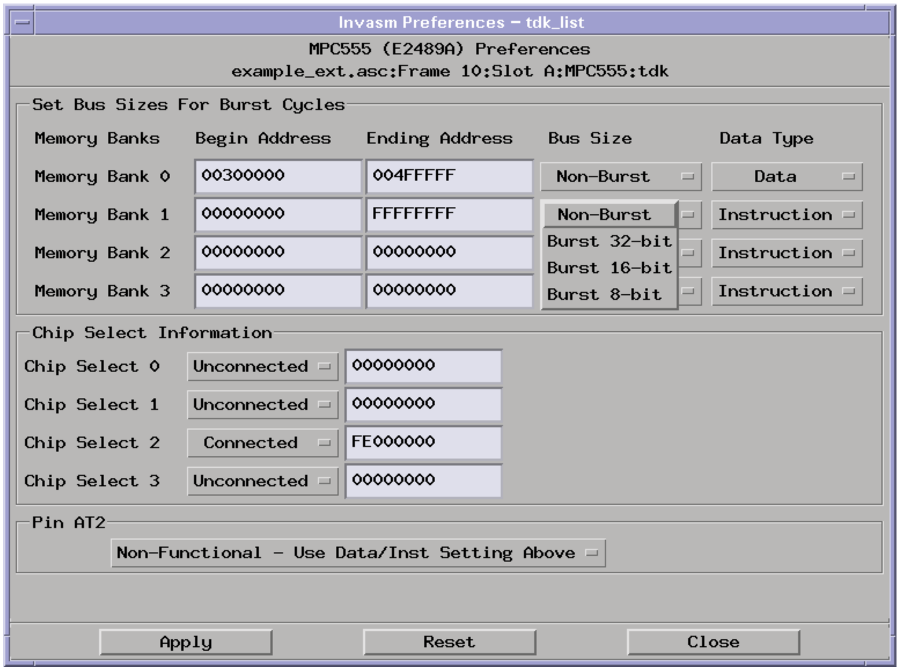
The ELF file, created by the compiler, uses software addresses. The software address must match the address created by the inverse assembler and displayed in the Listing window.

The MPC555 can drive up to four chip select lines to select external memory regions (memory banks). Use the Chip Select Information section to set the address offset associated with each memory region.

Unconnected (default). The inverse assembler will ignore the external memory chip select pins. The physical address is the same as the software address.

Connected. When a chip select is set to "Connected" and the corresponding chip select line is asserted low, the inverse assembler will create the 32-bit software address by adding the address presented on the external 24-bit address bus to the chip select information.

Invasm Preferences Dialog



Invasm Preferences – tdk_list

MPC555 (E2489A) Preferences
example_ext.asc:Frame 10:Slot A:MPC555:tdk

Set Bus Sizes For Burst Cycles

Memory Banks	Begin Address	Ending Address	Bus Size	Data Type
Memory Bank 0	00300000	004FFFFF	Non-Burst	Data
Memory Bank 1	00000000	FFFFFFFF	Non-Burst	Instruction
Memory Bank 2	00000000	00000000		Instruction
Memory Bank 3	00000000	00000000		Instruction

Non-Burst
Burst 32-bit
Burst 16-bit
Burst 8-bit

Chip Select Information

Chip Select 0	Unconnected	00000000
Chip Select 1	Unconnected	00000000
Chip Select 2	Connected	FE000000
Chip Select 3	Unconnected	00000000

Pin AT2

Non-Functional – Use Data/Inst Setting Above

Pin AT2 Setting

The Pin AT2 setting tells the inverse assembler how the IRQ4/AT2/SGPIOC4 pin is configured. The pin can be configured as "Functional" or "Non-Functional."

Functional - Use AT2 for Instruction/Data. If the pin is configured as "Functional," the IRQ4/AT2/SGPIOC4 pin is used for the AT2 function. The data type is ascertained from the pin. In this case, ignore the Preferences dialog setting called "Data Type."

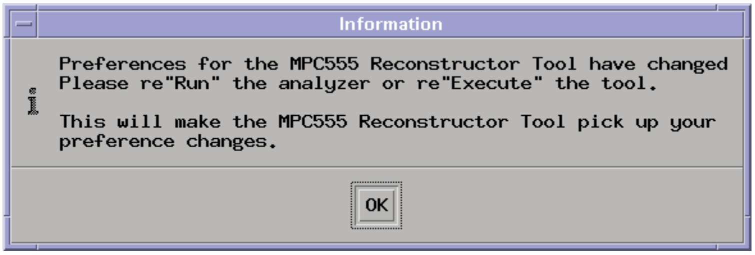
Non-Functional - Use Data/Inst Setting Above. If you are using a preliminary version of the MPC555 chip which does not have a functional AT2 pin, or if you have configured the pin as IRQ4 or SGPIOC4, you must set the mode to "Non-Functional" and set the "Data Type" for each memory bank so the inverse assembler can properly decode valid cycles as instructions or data.

Information Window

When you change the user's preferences settings and select the "Apply" button, an information window will be displayed. It will inform you that you must either double-click the MPC555 reconstructor tool or run the logic analyzer again so that the MPC555 reconstructor tool can apply the changes to the acquired data. If you are not using the MPC555 reconstructor tool, you can ignore this message.

The following figures show the Preferences dialog and the Information window.

Invasm Preferences Information Window



```
Preferences for the MPC555 Reconstructor Tool have changed
Please re"Run" the analyzer or re"Execute" the tool.

This will make the MPC555 Reconstructor Tool pick up your
preference changes.
```

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 ELF files into Agilent Technologies logic analyzers. The ELF file is different from the S-Record file used in show cycle reconstruction.

To view predefined symbols for the MPC555

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 MPC555 inverse assembler contain predefined symbols for logic analyzer labels.

To display the predefined symbols for the MPC555:

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
AT2	0	DATA
	1	INST
BDIP	0	BURST IN PROGRESS
	1	
BI	0	INHIBIT
	1	
BURST	0	BURST
	1	
CR	0	
	1	CANCEL RESERVATI
FREEZE	0	FREEZE
	1	
HRESET	0	HRESET
	1	
OE	0	ENABLED
	1	
RD/WR	0	WRITE
	1	READ
RETRY	0	RETRY
	1	
SRESET	0	SRESET
	1	
STS	0	SPECIAL XFER STA
	1	
TA	0	XFER ACKNOWLEDGE
	1	
TEA	0	XFER ERR ACK
	1	
TS	0	XFER START
	1	
TSIZ01	00	32-BIT
	01	8-BIT
	10	16-BIT
	11	

Predefined Symbols Description

Label	Encoding	Symbol
VF0-2	000	NONE
	001	SEQ INTR EXECUTE
	010	BR NOT TAKEN
	011	VSYNC ASSERTED
	100	EXCEPTION TAKEN
	101	BR INDIRECT TAKE
	110	BR DIRECT TAKEN
	111	BR NOT TAKEN
VFLS	00	0 INSTR FLUSHED
	01	1 INSTR FLUSHED
	10	2 INSTR FLUSHED
	11	FREEZE
WE	XXX0	WRITE EN
	XXX1	

To load object file symbols

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

If your compiler generates object files in a format that the logic analyzer doesn't understand, you can use a General Purpose ASCII (GPA) symbol file (see Chapter 14, “General-Purpose ASCII (GPA) Symbol File Format,” on page 201).

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

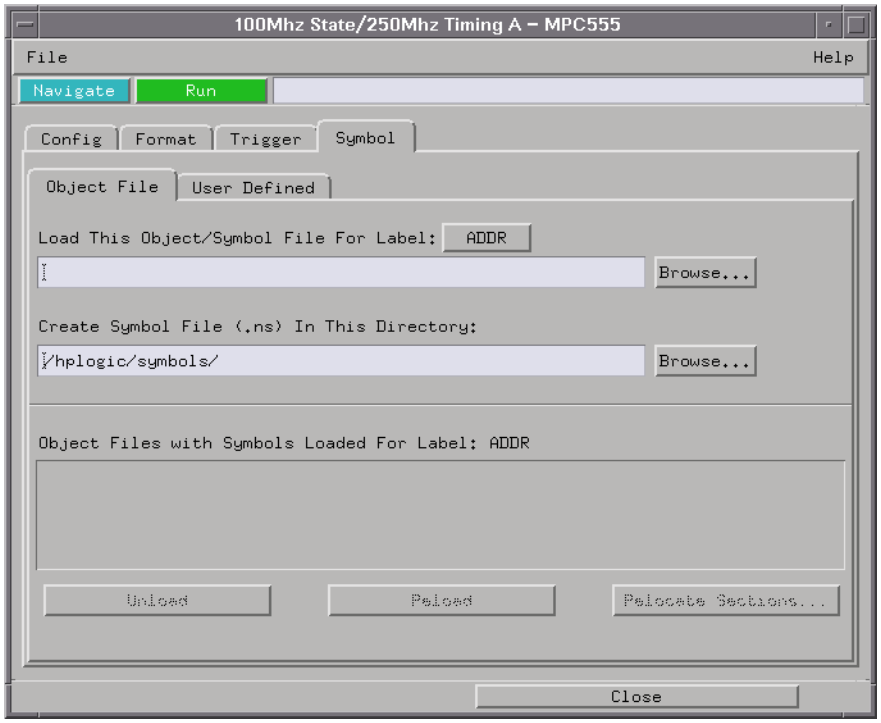
1 Open the logic analyzer module's Setup window.

2 Select the Symbol tab.

3 Select the Object File tab.

Make sure the label is ADDR.

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


100Mhz State/250Mhz Timing A – MPC555

File Help

Navigate Run

Config Format Trigger Symbol

Object File User Defined

Load This Object/Symbol File For Label: ADDR

Browse...

Create Symbol File (.ns) In This Directory:

/hplogic/symbols/

Browse...

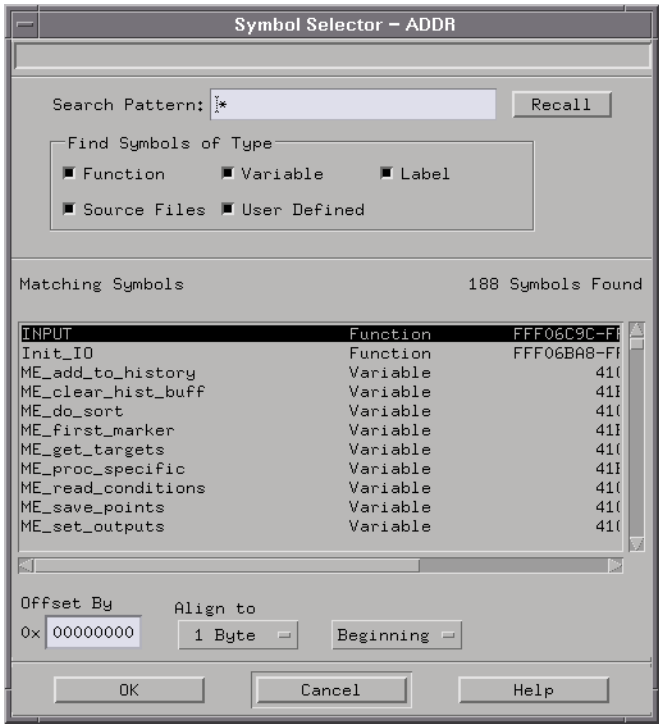
Object Files with Symbols Loaded For Label: ADDR

Unload Reload Relocate Sections...

Close

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 188 Symbols Found

```
INPUT                             Function        FFF06C9C-FF
Init_IO                           Function        FFF06BA8-FF
ME_add_to_history                 Variable                41(
ME_clear_hist_buff                Variable                41I
ME_do_sort                        Variable                41(
ME_first_marker                   Variable                41I
ME_get_targets                    Variable                41(
ME_proc_specific                  Variable                41I
ME_read_conditions                Variable                41(
ME_save_points                    Variable                41(
ME_set_outputs                    Variable                41(
```

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 analysis mode, every microprocessor clock cycle is captured by the logic analyzer, including idle and wait states. (This is the default mode set up by the configuration files.)

Because the VF/VFLS pins are valid on every rising edge of the clock, use the state analysis mode to trigger or to view activity on these pins.

To configure the logic analyzer for state-per-clock mode:

1 Load the appropriate logic analyzer configuration file (see “Loading Configuration Files” on page 82).

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

You can change the master clock setting by opening the logic analyzer’s Setup window, selecting the Format tab, and clicking the Mast Clk 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.

To configure the logic analyzer for timing analysis:

1. Load the appropriate logic analyzer configuration file (see “Loading Configuration Files” on page 82).
2. Open the logic analyzer’s Setup window.
3. Select the Config tab.
4. Change the type option from State to Timing.

6

Capturing MPC555 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.

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 82).

You can load program object file symbols into the logic analyzer when configuring it (see "Loading Symbol Information" on page 91).

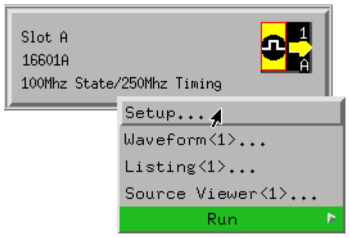
This chapter describes setting up logic analyzer triggers when using the Agilent Technologies E2489A inverse assembler and the Agilent Technologies B4620B source correlation tool set.

See Chapter 7, "Displaying Captured MPC555 Execution," on page 107 for information on displaying captured data.

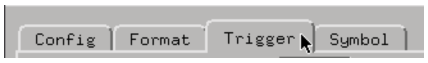
Setting Up Logic Analyzer Triggers

To set up logic analyzer triggers

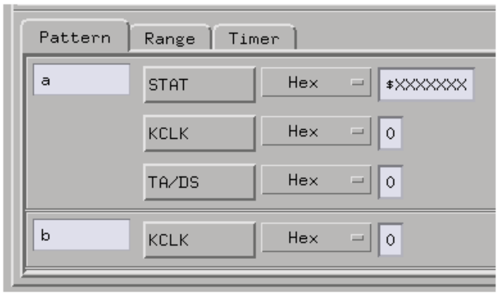
1 Open the logic analyzer's Setup window.



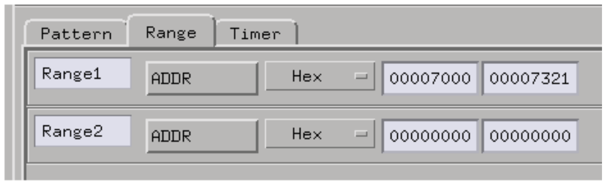
2 Select the Trigger tab.



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

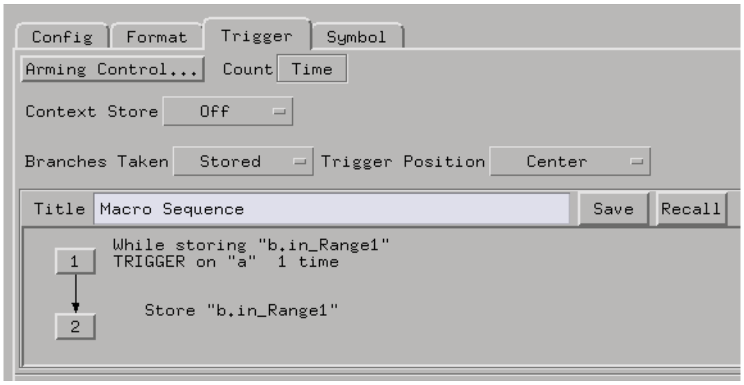


a	STAT	Hex	$XXXXXXX
	KCLK	Hex	0
	TA/DS	Hex	0
b	KCLK	Hex	0



Range1	ADDR	Hex	00007000	00007321
Range2	ADDR	Hex	00000000	00000000

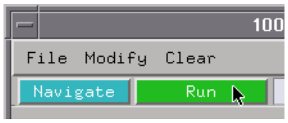
4 Set up the trigger sequence.



Context Store Off
Branches Taken Stored Trigger Position Center
Title Macro Sequence

```
1  While storing "b.in_Range1"
   TRIGGER on "a"  1 time

2     Store "b.in_Range1"
```

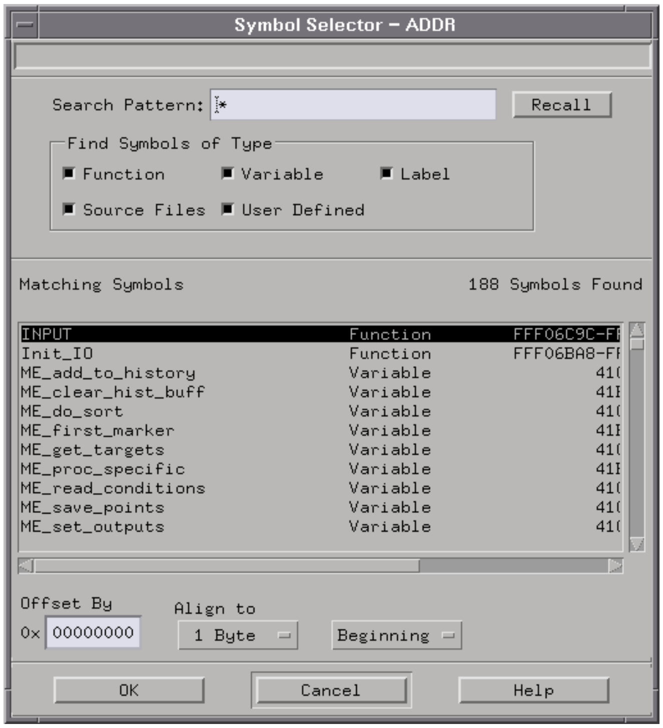
5 Run the measurement.



See Also The Agilent Technologies 16700-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.



Search Pattern: *

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

Matching Symbols 188 Symbols Found

```
INPUT                            Function        FFF06C9C-FF
Init_IO                          Function        FFF06BA8-FF
ME_add_to_history                Variable                41(
ME_clear_hist_buff               Variable                41I
ME_do_sort                       Variable                41(
ME_first_marker                  Variable                41I
ME_get_targets                   Variable                41(
ME_proc_specific                 Variable                41I
ME_read_conditions               Variable                41(
ME_save_points                   Variable                41(
ME_set_outputs                   Variable                41(
```

Offset By 0x 00000000   Align to 1 Byte   Beginning

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

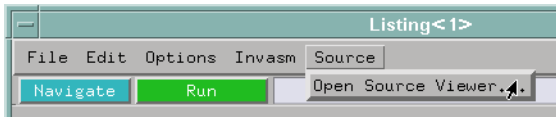
Triggering on Source Code

When setting up trigger specifications to capture MPC555 execution:

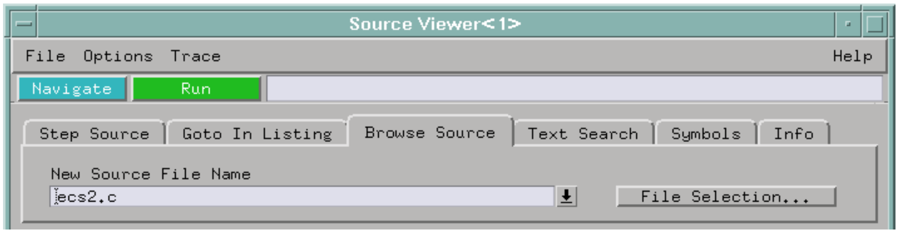
- Use the logic analyzer storage qualification to capture the software execution you're interested in and filter out library code execution (whose source file lookups can take a long time if the library source code is not available).

To set up triggers based on source code

1 Open the Source Viewer window.

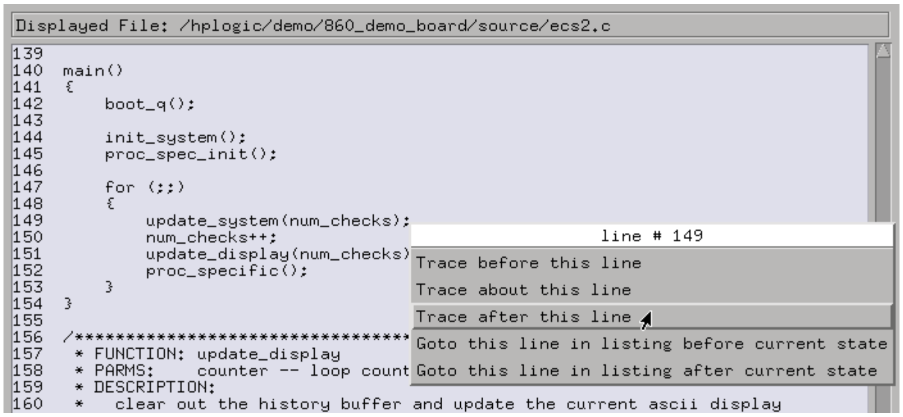


2 Browse the source file that contains the code you want to trigger on.



New Source File Name
ecs2.c

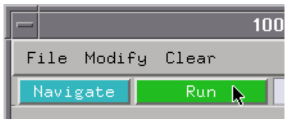
3 Click the source code line you want to trigger on and specify whether you want to trace before, about or after the line. Or, use the Source Viewer's Trace menu to trace about a variable, function, or line number.



```
Displayed File: /hplogic/demo/860_demo_board/source/ecs2.c

main()
{
    boot_q();

    init_system();
    proc_spec_init();

    for (;;)
    {
        update_system(num_checks);
        num_checks++;
        update_display(num_checks)
        proc_specific();
    }
}

/**********************************
 * FUNCTION: update_display
 * PARMS:    counter -- loop count
 * DESCRIPTION:
 *   clear out the history buffer and update the current ascii display
```

line # 149
Trace before this line
Trace about this line
Trace after this line
Goto this line in listing before current state
Goto this line in listing after current state

4 Run the measurement.



To avoid capturing library code execution

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

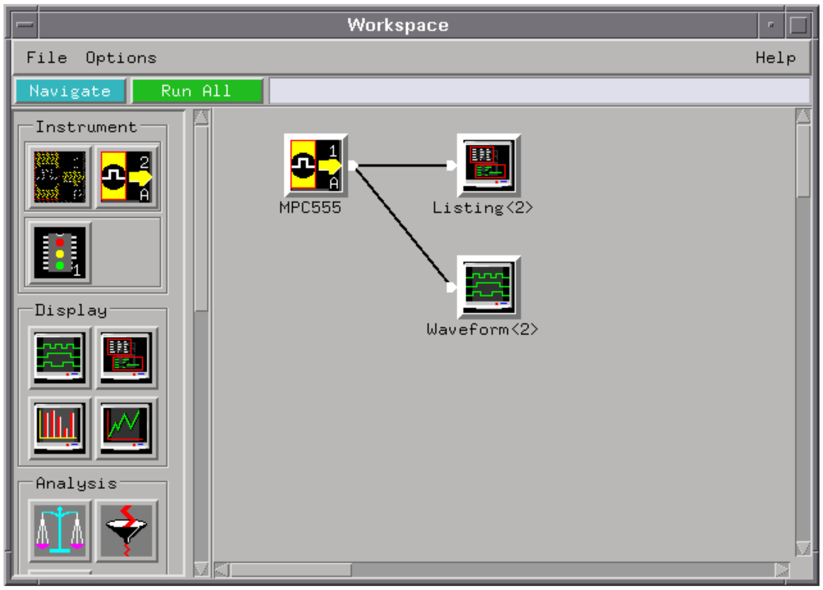
You should also configure the logic analyzer's storage qualification capabilities to store only those cycles that correspond to software execution (non-idle, etc.).

7

Displaying Captured MPC555 Execution

Displaying Data from the Logic Analyzer

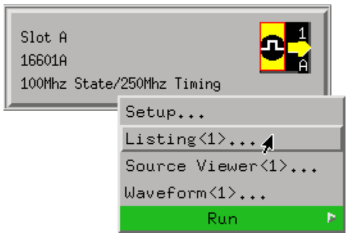
When you load the configuration files supplied with the inverse assembler, the Workspace window is set up as shown below.



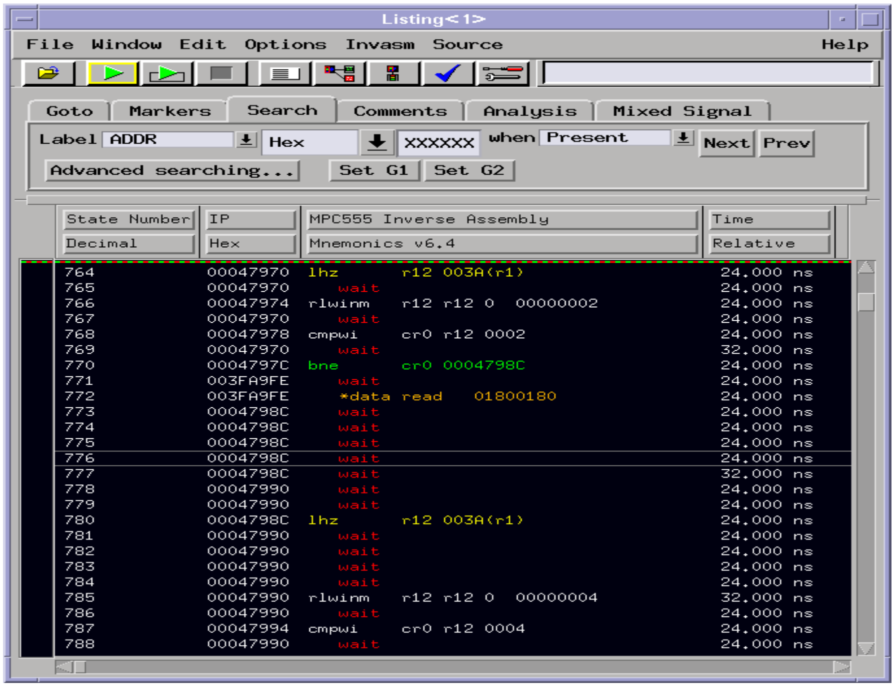
This setup is appropriate for debugging hardware. When you're debugging software, you'll probably want to use the trace reconstruction tool (see "Displaying Data from the Trace Reconstruction Tool" on page 112).

To display the captured state data

1 Open the Listing display window.



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


Listing<1>
File Window Edit Options Invasm Source Help
Goto Markers Search Comments Analysis Mixed Signal
Label ADDR Hex XXXXXX when Present Next Prev
Advanced searching... Set G1 Set G2

```
State Number  IP        MPC555 Inverse Assembly            Time
Decimal       Hex       Mnemonics v6.4                     Relative
764           00047970  lhz      r12 003A(r1)              24.000 ns
765           00047970     wait                            24.000 ns
766           00047974  rlwinm   r12 r12 0   00000002      24.000 ns
767           00047970     wait                            24.000 ns
768           00047978  cmpwi    cr0 r12 0002              24.000 ns
769           00047970     wait                            32.000 ns
770           0004797C  bne      cr0 0004798C              24.000 ns
771           003FA9FE     wait                            24.000 ns
772           003FA9FE     *data read    01800180          24.000 ns
773           0004798C     wait                            24.000 ns
774           0004798C     wait                            24.000 ns
775           0004798C     wait                            24.000 ns
776           0004798C     wait                            24.000 ns
777           0004798C     wait                            32.000 ns
778           00047990     wait                            24.000 ns
779           00047990     wait                            24.000 ns
780           0004798C  lhz      r12 003A(r1)              24.000 ns
781           00047990     wait                            24.000 ns
782           00047990     wait                            24.000 ns
783           00047990     wait                            24.000 ns
784           00047990     wait                            24.000 ns
785           00047990  rlwinm   r12 r12 0   00000004      32.000 ns
786           00047990     wait                            24.000 ns
787           00047994  cmpwi    cr0 r12 0004              24.000 ns
788           00047990     wait                            24.000 ns
```

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 I555E, and it is located in the /logic/ ia directory.

Because the PowerPC 555 presents one address and then reads 16 bytes during a burst, the least-significant digit is synthesized by the disassembler. The entire synthesized address appears under the “IP” label. The actual address bits presented by the PowerPC 555 may be observed under the ADDR label.

In the listing window shown here, an asterisk (“*”) is displayed on line 772. This indicates a data show cycle rather than a typical data read or write.

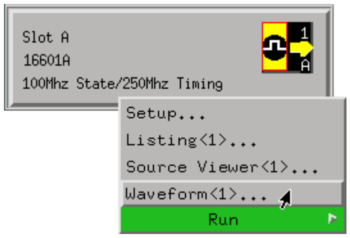
See Also

“To use the inverse assembler filters” on page 119 for information on displaying or hiding certain types of microprocessor bus cycles.

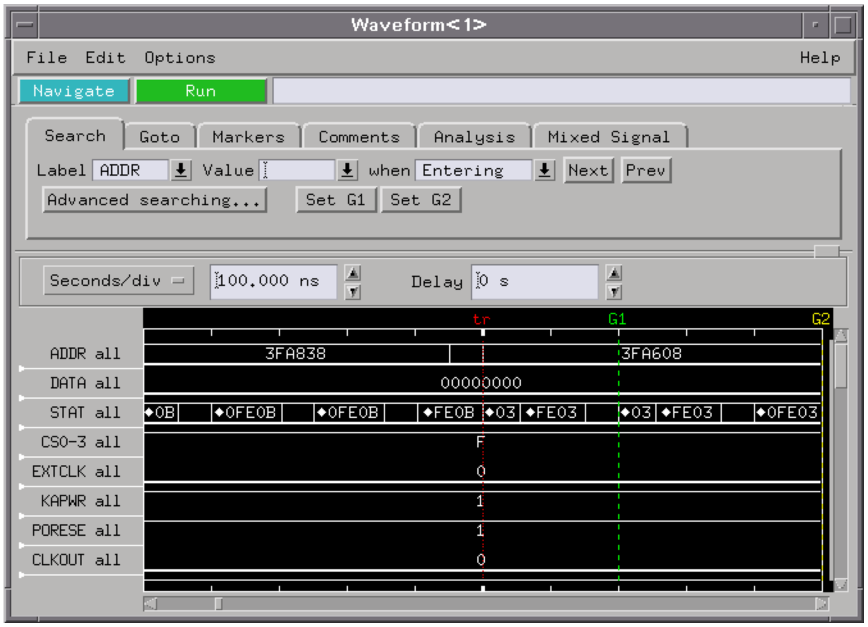
The Agilent Technologies 16700-series logic analysis system on-line help for information on using the Listing display.

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 ADDR Value when Entering Next Prev
Advanced searching... Set G1 Set G2
Seconds/div 100.000 ns Delay 0 s
tr G1 G2
ADDR all 3FA838 3FA608
DATA all 00000000
STAT all ◆0B ◆0FE0B ◆0FE0B ◆FE0B ◆03 ◆FE03 ◆03 ◆FE03 ◆0FE03
CS0-3 all F
EXTCLK all 0
KAPWR all 1
PORESE all 1
CLKOUT all 0

Displaying Data from the Trace Reconstruction Tool

The trace reconstruction tool uses show cycles, the VF/VFLS signals, and program image files (S-Records) to decode captured MPC555 execution into complete program traces. When compared to data that comes straight from the logic analyzer, the data from the trace reconstruction tool:

- Contains code that executes out of internal memory.
- Has unexecuted prefetches removed.
- Has external bus fetches removed and replaced by show cycles
- Shows the actual execution times of instructions.

The data from the trace reconstruction tool contains only code that has been executed by the microprocessor. Read and write cycles captured by the logic analyzer are unchanged.

The trace reconstruction tool is appropriate for debugging software. When you're debugging hardware, you'll probably want to look at the actual data captured by the logic analyzer (see "Displaying Data from the Logic Analyzer" on page 108).

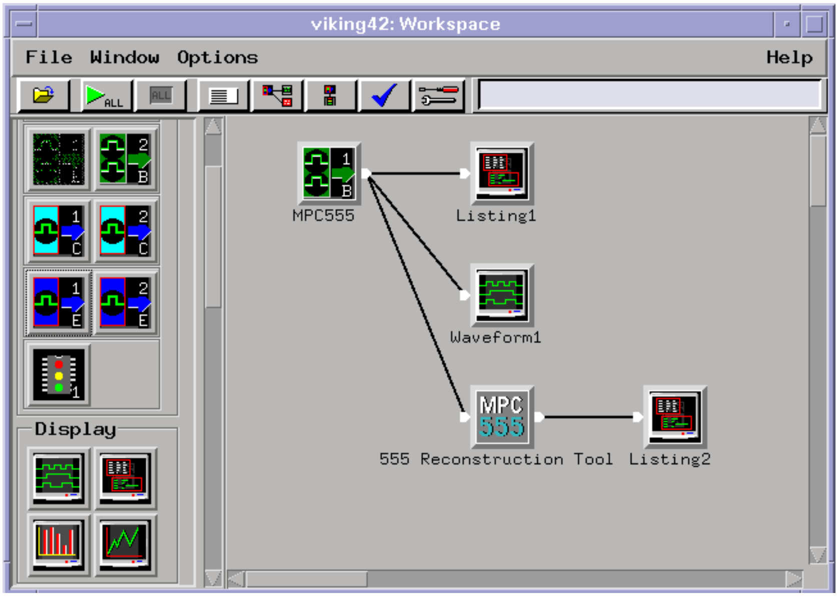
NOTE: Don't debug hardware based on the ADDR, DATA, and STAT values output by the trace reconstruction tool because they don't represent the actual signal values captured by the logic analyzer.

To add the trace reconstruction tool to the workspace

1 Open the Workspace window.

2 Drag the 555 Reconstruction Tool from the tool box and drop it on the logic analyzer instrument.

3 Drag a Listing display tool and drop it on the 555 Reconstruction Tool.

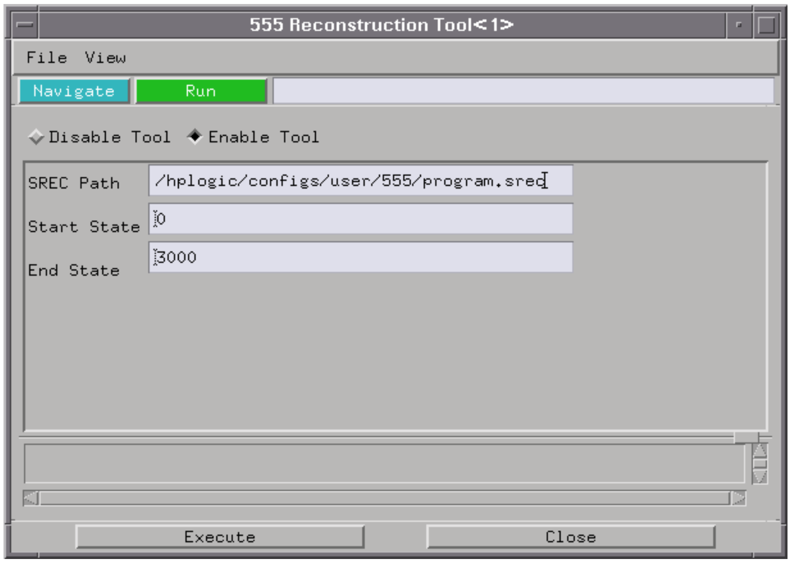


To load program image files into the trace reconstruction tool

Before the trace reconstruction tool can work, you must give it a program image file in S-Record format.

1 In the Workspace window, double-click the 555 Reconstruction Tool.

2 Enter the name of your program image S-Record file.



SREC Path /hplogic/configs/user/555/program.srec
Start State 0
End State 3000

3 Enter the starting and ending states of the range to be reconstructed.

Limiting the number of states to be reconstructed results in faster displays.

4 Click Execute.

To use multiple program image files

You can use multiple image files by concatenating the program image files for different parts of the target system code and loading the resulting combined image file.

If undefined opcodes appear in the output

If “Undefined Opcode”s appear in the output of the trace reconstruction tool, it means the opcode for that address could not be found in the program image file.

When this happens, the decoded instructions that follow, up to the next show cycle, may not be correct.

❑ Make sure that all program image files are loaded.

If the trace contains all wait states

The trace reconstruction tool needs a show cycle to sync up with before it can start decoding data. If there is no show cycle, all data, except reads and writes, are shown as wait states.

This situation can occur when indirect branches are enabled and the program has not yet encountered an indirect branch. One solution is to enable direct branches; the program will sync on the first direct branch in the listing.

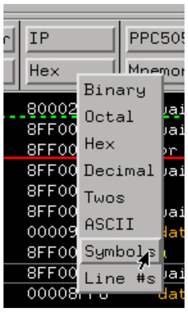
The second solution is to use one of these four sequential instructions in the program where there is lack of indirect branches: rfi, mtmsr, isync, and mtspr. These instructions will cause indirect branch show cycles and will provide the information the reconstruction tool needs to sync.

Using the Listing and Source Viewer Displays

The Listing and Source Viewer displays (as well as other display tools) can be used with either data straight from the logic analyzer or with data that comes from the trace reconstruction tool.

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 93.

To interpret the inverse assembled data

The next few paragraphs describe the general output format of the logic analyzer inverse assembler. For information on PowerPC 555 modes of operation, refer to "Unsupported Microprocessor Modes" on page 24.

Interpreting Data

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

Most numerical data is displayed in hexadecimal, for example, "lwz r28 0044(r1)."

Bit numbers and shift counts are displayed in decimal with a dot suffix, for example, "cror 31. 31. 31."

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

The disassembler decodes the full PowerPC instruction set architecture, including 64-bit mode instructions and optional instructions not implemented on the PowerPC 555. When these unimplemented opcodes are encountered, the instruction mnemonic has a "?" prefix. If a reserved bit is set in an instruction opcode field, a "?" is appended most often to the mnemonic, but in some cases to an operand.

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

Endian Mode

The inverse assembler only supports big-endian mode. Little-endian mode is not supported.

Burst Mode

The trace reconstruction tool reconstructs the least significant bits of burst accesses, so that every beat of the burst access has the correct address associated with it. Burst mode is supported for both 16-bit and 32-bit memory ports.

Pipelined Data

For pipelines accesses, the address phase and data phase of the pipelined data are aligned by the trace reconstruction tool.

Branch Instructions

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

Extended Mnemonics

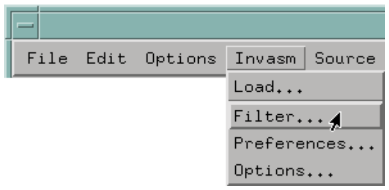
PowerPC assemblers support a number of extended mnemonics for some popular assembly language instructions as described in the *PowerPC 555 User's Manual*. The E2489A inverse assembler supports the following extensions:

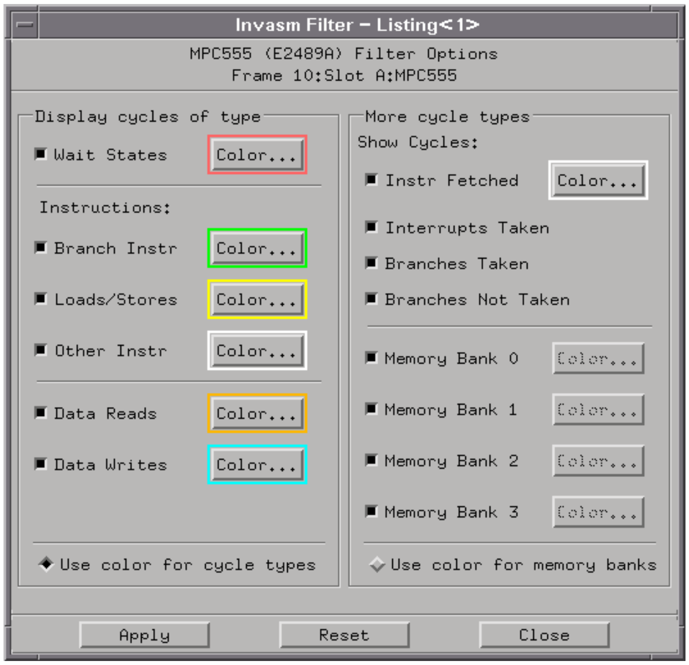
- Conditional traps and branches decode the condition mnemonically when possible. For some conditions which have no conventional mnemonics (for example, "signed less than or unsigned greater than"), the condition field is displayed in binary.
- The L bit is omitted as a compare operand. Instead, compares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

Mnemonic	Decoded As
rlwimi (rotate left word immediate then mask insert)	inslwi (insert from left immediate) insrwi (insert from right immediate)
rlwinm (rotate left word immediate then AND with mask)	rotlwi (rotate left immediate) rotrwi (rotate right immediate) slwi (shift left immediate) srwi (shift right immediate) extlwi (extract and left justify immediate) extrwi (extract and right justify immediate) clrlwi (clear left immediate) clrrwi (clear right immediate) clrlslwi (clear left and shift left immediate)
rlwnm (rotate left word then AND with mask)	rotlw (rotate left)

To use the inverse assembler filters

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




Invasm Filter – Listing<1>

MPC555 (E2489A) Filter Options
Frame 10:Slot A:MPC555

Display cycles of type

■ Wait States [Color...]

Instructions:

■ Branch Instr [Color...]

■ Loads/Stores [Color...]

■ Other Instr [Color...]

■ Data Reads [Color...]

■ Data Writes [Color...]

◆ Use color for cycle types

More cycle types

Show Cycles:

■ Instr Fetched [Color...]

■ Interrupts Taken

■ Branches Taken

■ Branches Not Taken

■ Memory Bank 0 [Color...]

■ Memory Bank 1 [Color...]

■ Memory Bank 2 [Color...]

■ Memory Bank 3 [Color...]

◇ Use color for memory banks

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

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

Filtering allows faster analysis in two ways:

- Unneeded information can be taken out of the display. For example, suppressing idle states will show only states in which a transaction was completed.
- Particular operations can be isolated by suppressing all other operations. For example, Branch Instr can be shown, with all other states suppressed, allowing quick analysis of branch instructions.

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

You can display or hide the following types of cycles:

- Idle/Wait States - A state in which there is no valid instruction, data read/

write, show cycle, or VF/VFLS activity.

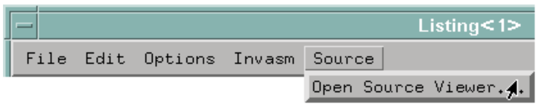
- Show Cycles - (For additional information on the type of information that is displayed for Show Cycles, see “To display the captured state data” on page 108).
- Branch Instructions.
- Loads and Stores.
- All Other Instructions - (Besides branches, loads, and stores).
- Data Reads.
- Data Writes.
- Instructions Fetched - These are VF/VFLS states that are the result of an instruction fetch. The instruction code executed will be hidden/displayed.
- Interrupts Taken - These are VF/VFLS states that are the result of an interrupt or an exception. The inverse assembler will also attempt to decode the type of interrupt based on the interrupt vector address. The interrupt code executed will be hidden/displayed.
- Branches Taken - These are VF/VFLS branches.
- Branches Not Taken - These are any VF/VFLS conditional branches NOT taken.
- Memory Bank 0-3 accesses.

See Also

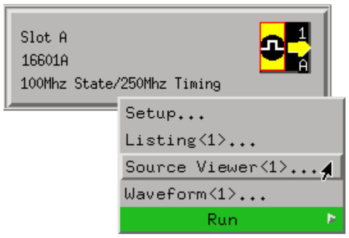
The address ranges for Memory Banks 0-3 are specified in the Preferences menu (see “To set the inverse assembler preferences” on page 88).

To view the source code associated with captured data

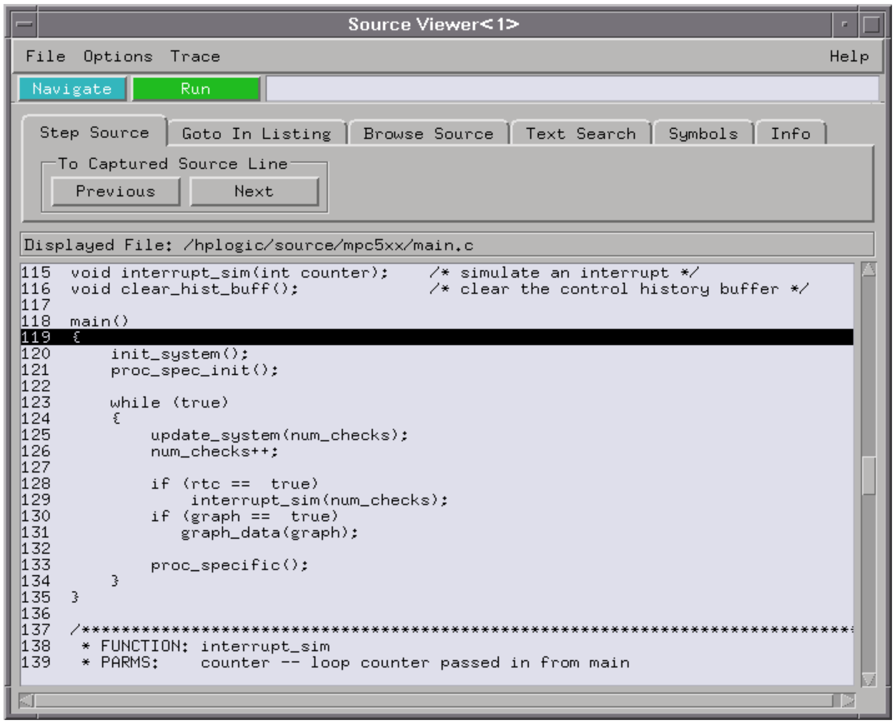
- In the Listing display window, select Source Viewer from the Source menu.



- 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 93).



Displayed File: /hplogic/source/mpc5xx/main.c

```
void interrupt_sim(int counter);     /* simulate an interrupt */
void clear_hist_buff();              /* clear the control history buffer */

main()
{
    init_system();
    proc_spec_init();

    while (true)
    {
        update_system(num_checks);
        num_checks++;

        if (rtc ==  true)
            interrupt_sim(num_checks);
        if (graph ==  true)
           graph_data(graph);

        proc_specific();
    }
}

/***************************************************************************
 * FUNCTION: interrupt_sim
 * PARMS:    counter -- loop counter passed in from main
```

Inverse Assembler Generated IP (Software Address) Label

In the Agilent Technologies 16700-series logic analysis system, the MPC5xx inverse assembler generates a “IP” label. The IP 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 16700-series logic analysis system perform a pattern search on the IP 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.

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. See also “Power-ON/Power-OFF Sequence” on page 48.

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 130 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 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 130. 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 63 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 Chapter 5, "Configuring the Logic Analyzer," on page 81 for more information.

❏ Verify that all microprocessor caches and memory managers (if present) 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 Chapter 5, “Configuring the Logic Analyzer,” on page 81 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.

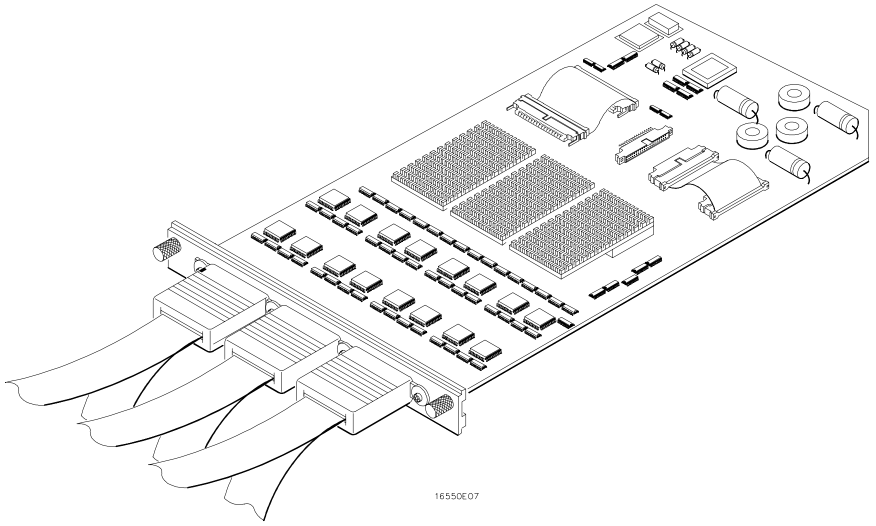
Ensure that the inverse assembler file is not renamed or deleted, and that it is located in the correct directory:

- For Agilent Technologies 16700-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.

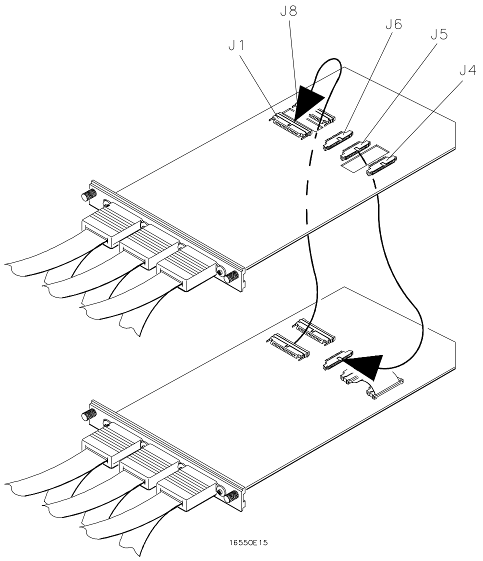
“Measurement Initialization Error”

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 Chapter 5, "Configuring the Logic Analyzer," on page 81 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 16700-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 63 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

The emulation module and firmware for the Motorola MPC555 is also used with the MPC505/509 processors.

9

Using the Emulation Control Interface

The Emulation Control Interface in your Agilent Technologies 16700-series logic analysis system allows you to control an emulator (an emulation module or an emulation probe).

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

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

The Emulation Control Interface allows you to:

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

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

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

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

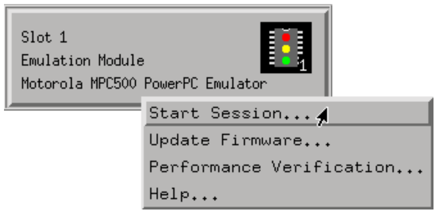
- The interfaces can get out of synchronization when commands are issued from both interfaces. This causes windows to be out-of-date and can cause confusion.
- Most debuggers cannot tolerate another interface issuing commands and may not start properly if another interface is running.

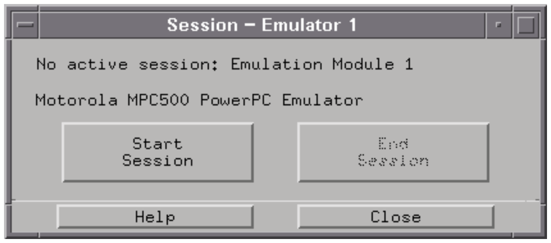
See Also All of the Emulation Control Interface windows provide on-line help with a Help button or a Help—>On this window menu selection. Refer to the on-line help for complete details about how to use a particular window.

To start from the main System window

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

2 Select Start Session....





No active session: Emulation Module 1

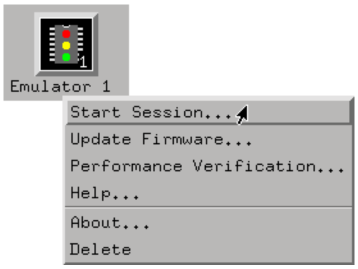
Motorola MPC500 PowerPC Emulator

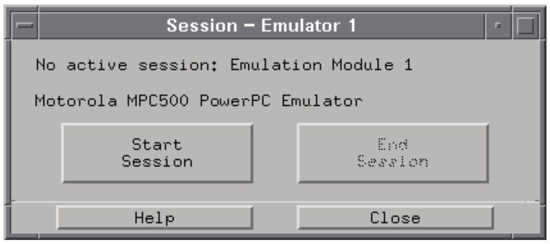
To start from the Workspace window (emulation module)

1 Open the Workspace window.

2 Drag the Emulator icon onto the workspace.

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




Session – Emulator 1

No active session: Emulation Module 1

Motorola MPC500 PowerPC Emulator

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 16700 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

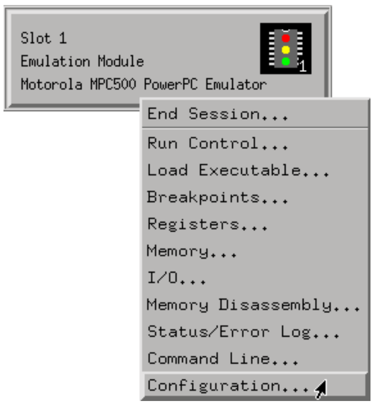
The easiest way to configure the emulation module is to use the Emulation Control Interface.

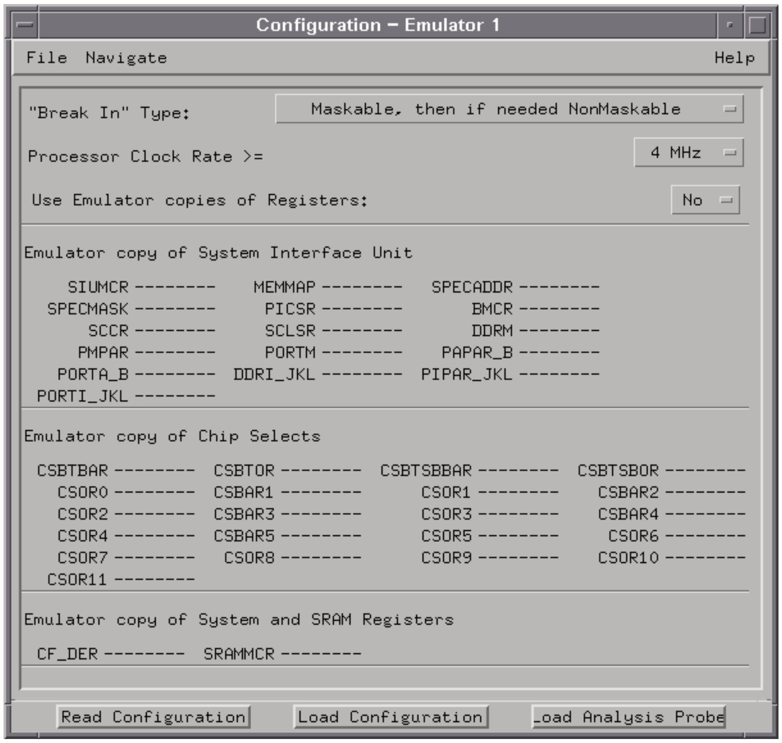
1 Start an Emulation Control Interface session.

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.


Slot 1
Emulation Module
Motorola MPC500 PowerPC Emulator


Configuration – Emulator 1

"Break In" Type: Maskable, then if needed NonMaskable

Processor Clock Rate >= 4 MHz

Use Emulator copies of Registers: No

Emulator copy of System Interface Unit

```
   SIUMCR --------    MEMMAP --------   SPECADDR --------
 SPECMASK --------     PICSR --------       BMCR --------
     SCCR --------     SCLSR --------       DDRM --------
    PMPAR --------     PORTM --------    PAPAR_B --------
  PORTA_B --------  DDRI_JKL --------   PIPAR_JKL --------
PORTI_JKL --------
```

Emulator copy of Chip Selects

```
CSBTBAR --------  CSBTOR --------  CSBTSBBAR --------  CSBTSBOR --------
  CSOR0 --------  CSBAR1 --------      CSOR1 --------    CSBAR2 --------
  CSOR2 --------  CSBAR3 --------      CSOR3 --------    CSBAR4 --------
  CSOR4 --------  CSBAR5 --------      CSOR5 --------     CSOR6 --------
  CSOR7 --------   CSOR8 --------      CSOR9 --------    CSOR10 --------
 CSOR11 --------
```

Emulator copy of System and SRAM Registers

```
CF_DER --------  SRAMMCR --------
```

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 179.

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
  procck    - Set clock speed of target processor
  dprocck   - Display default clock speed of target processor
  bnchardbrk - Set BNC break type
  breakin   - BNC break in control
  rrt       - Set restriction to real time runs
  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.

Setting the MPC5xx 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.
- Debug port connection type (MPC555 only).
- Reset configuration word source (MPC505/509 only)
- “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
MPC505	MPC505	cf proc=MPC505
MPC509	MPC509 (Default)	cf proc=MPC509
MPC555	MPC555	cf proc=MPC555

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

To configure the processor clock speed

The BDM communication speed will be 1/3 of the configured processor clock speed. You may set the processor clock speed to a speed lower than the actual clock speed of your target system.

Processor clock speed configuration

Value	Processor clock is at least	Built-in command
25	25 MHz	cf procck=25
20	20 MHz	cf procck=20
16	16 MHz	cf procck=16
8	8 MHz	cf procck=8
4	4 MHz (default)	cf procck=4
1	1 MHz	cf procck=1
512	512 kHz	cf procck=512
32	32 kHz	cf procck=32

You can also set the reset clock speed, which controls the BDM communication speed used after a reset, but before the Multiplication Factor in the SCCR is set up:

Reset processor clock speed configuration

Value	Processor clock is at least	Built-in command
25	25 MHz	cf dprocck=25
4	4 MHz	cf dprocck=4
32	32 kHz	cf dprocck=32

To configure the debug port connection type

This configuration option is valid when the processor type has been configured for MPC555.

Debug port connection type configuration

Value	Processor probe configured for	Built-in command
1	Maximum debug capability: Debug port pin 1 = MPIO3/VFLS0 Debug port pin 6 = MPIO4/VFLS1 The emulation probe sets MIOS1TPCR=0x0003 to enable VFLS[0:1] pins after a reset->break. Note: reset->run will not work. See also “MPC555 Debug Port Connector, Option 1” on page 43.	cf dbgconfig=1
2	Maximum external bus capability (Default): Debug port pin 1 = VFLS0/IWP0 Debug port pin 6 = VFLS1/IWP1 By default, the processor powers up with VFLS[0:1] pin function. See also “MPC555 Debug Port Connector, Option 2” on page 44.	cf dbgconfig=2
3	Maximum I/O configuration: Debug port pin 1 = SGPIO6/FRZ/PTR Debug port pin 6 = SGPIO6/FRZ/PTR The emulation probe sets bit 13 of SIUMCR to enable the FRZ pin after a reset->break. Note: reset->run will not work. See also “MPC555 Debug Port Connector, Option 3” on page 44.	cf dbgconfig=3

To configure the reset configuration word source

This configuration option is valid when the processor type has been configured for MPC505 or MPC509.

Reset configuration word source

Value	Which reset configuration word will be used?	Built-in command
int	Internal.	cf rstword=int
ext	External.	cf rstword=ext

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 break to the monitor. Examples include commands which display memory or registers. (Default)	cf rrt=no
yes	No commands are allowed which break to the monitor, except “break,” “reset,” “run,” or “step.”	cf rrt=yes

Testing the Emulation Module and Target System

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

See Also See Chapter 12, "Troubleshooting the Emulation Module," on page 175 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

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 16700-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

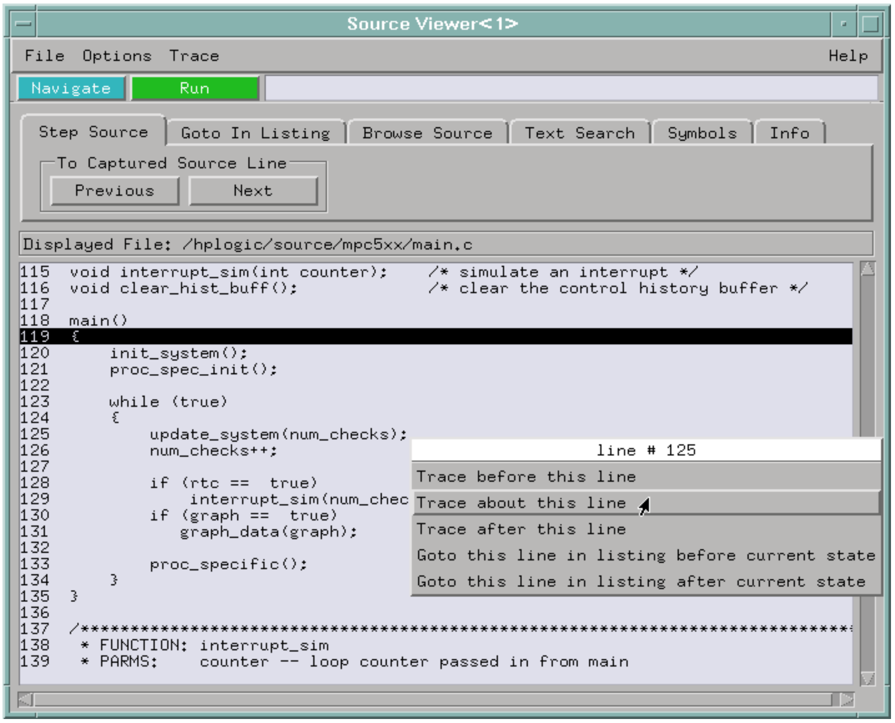
You can trigger the emulation module from the logic analyzer using either the Source Viewer window or the Intermodule window. If you are using the Agilent Technologies B4620B source correlation tool set, using the Source Viewer window is the easiest method.

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.



Displayed File: /hplogic/source/mpc5xx/main.c

```
void interrupt_sim(int counter);    /* simulate an interrupt */
void clear_hist_buff();             /* clear the control history buffer */

main()
{
    init_system();
    proc_spec_init();

    while (true)
    {
        update_system(num_checks);
        num_checks++;

        if (rtc ==  true)
            interrupt_sim(num_chec
        if (graph ==  true)
           graph_data(graph);

        proc_specific();
    }
}

/*************************************************************************
 * FUNCTION: interrupt_sim
 * PARMS:    counter -- loop counter passed in from main
```

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).

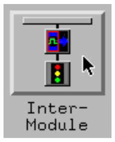
4 If your target system is not already running, click Run in the emulation Run Control window to start your target.

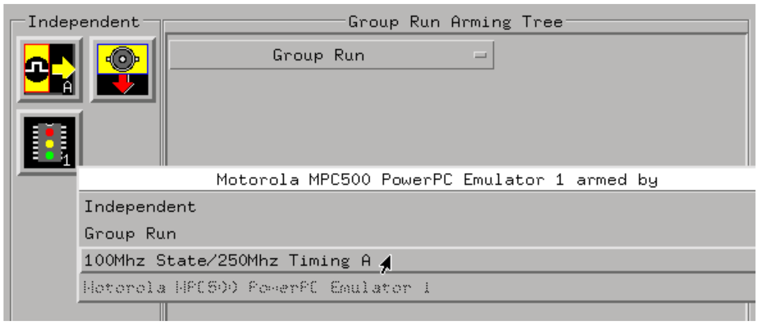
To stop on any trigger (Intermodule window)

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.

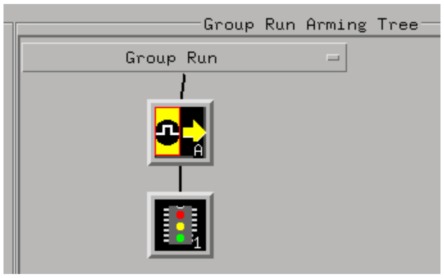
1 Create a logic analyzer trigger.

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





Motorola MPC500 PowerPC Emulator 1 armed by
Independent
Group Run
100Mhz State/250Mhz Timing A
Motorola MPC500 PowerPC Emulator 1



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 (if applicable).

See Also “To configure the processor clock speed” on page 151.

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 167).

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.

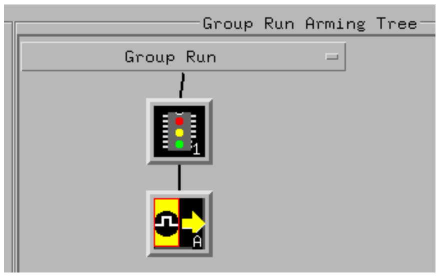
Here are some examples:

- 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 167).

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 167).

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.

12

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**
Target goes into reset unexpectedly	Ensure SYPCR register and watchdog timer settings are correct.	page 45
SYPCR value is different than expected value	Emulation probe may have written to the SYPCR prior to boot code execution.	page 45
Only wait states are shown in the listing window	The trace reconstruction tool needs a show cycle for synchronization.	page 115
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 179
Emulation module built-in commands do not work	**1** Check that the emulation module has been properly configured for your target system.	page 145
	2 Run the emulation module performance verification tests.	page 190
	3 If the performance verification tests pass, then there is an electrical problem with the connection to the target processor OR the target system may not have been designed according to “Designing a Target System.”	page 42, page 182
“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 151
Some commands fail	Check the “restrict to real-time runs” configuration.	page 155

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 16700-series logic analysis system is 6472. 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 r1 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 179 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 9 on header, 3.3V (V_{OD}).
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 SRESET and HRESET are working.

If interrupts are non-recoverable

❑ Check that interrupt service routines in the target code meet the requirements listed in the PowerPC documentation.

For proper debugging in interrupt service routines, the PowerPC documentation specifies that the exception handlers must do the following:

- As a prologue to the interrupt service routine:
  - Save the SRR0, SRR1, DAR, DSISR registers.
  - Set the RI bit in the MSR (Machine State Register.Recoverable Interrupt Bit).
- As an epilogue to the interrupt service routine:
  - Restore the SRR0, SRR1, DAR, DSISR registers.
  - Issue an RFI (Return from Interrupt) instruction.

Upon entering the interrupt service routine, the processor clears the MSR.RI bit, and copies the IP (Instruction Pointer)->SRR0 and the MSR->SRR1. The SRR0 and SRR1 are the save and restore registers. These contain the information needed to return to the state prior to the interrupt.

The RI bit will prevent the processor from breaking into debug mode with a maskable debug port breakpoint. A non-maskable breakpoint is required to break the processor when the RI bit is cleared, resulting in a possible non-recoverable state.

Software breakpoints place a "Trap" instruction into the breakpoint address. If the trap instruction is executed within an interrupt service routine, a break to background mode will occur. This causes the SRR0 ands SRR1 registers to be written over, causing a non-recoverable state. If the exception handler saves these registers, and sets the MSR.RI bit, the software breakpoint will always be recoverable.

If hardware breakpoints have no effect

Hardware breakpoints by default will not break the processor if they are set within an exception handler which has not saved the SRRs and set the MSR.RI bit. However, these can quite easily be reprogrammed to assert a non-maskable break. Note that the breakpoint will halt the processor, but will cause a non-recoverable state.

To reprogram the hardware breakpoint to assert a non-maskable break:

```
M>bc -e hwbp
M>reg lctrl2
  reg lctrl2=02018000
M>reg lctrl2=02018800 # OR in 0x00000800 with previous value
```

Hardware breakpoints will now cause a non-maskable break, which will halt the processor regardless of the status of the MSR.RI bit. Again, note that in this case the break will be non-recoverable if the exception handler has not saved the SRRs.

If the target resets itself

The most common plug-in issue is the target resetting itself. If the PC is set to some initial location, and then a short time later, the PC=100 or PC=fff00100, the target is resetting itself. In most cases, the chip is causing the reset, not the target hardware.

There are a number of possible causes of the reset. To determine the cause of reset, read the RSR (Reset Status Register):

```
M>m -a2 -d2 288@reg # telnet command which reads the RSR
```

The bits in this register show the cause of the reset:

RSR Bit Encoding

Bit	Cause of reset	Explanation
0 (MSB)	External Hard Reset	The emulation module actually uses an external reset when resetting the target.
1	External Soft Reset	
2	Loss of Lock	Caused when the PLL loses the phase lock on the external clock source.
3	SW Watchdog	Make sure the SYPCR register disables the watchdog timer. See page 45. R>reg cf_sypcr=ffffff88 or M>m -a4 -d4 4@reg=ffffff88
4	Checkstop	Occurs when the processor enters a checkstop state.
5	Debug Port Hard Reset	
6	Debug Port Soft Reset	
7	JTAG Reset	

To clear the RSR, execute the following:

```
M>m -a2 -d2 288@reg=ffff
```

If running from reset causes problems

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

If you see the “!ASYNC_STAT 173!” error message

If after a break, the following error arises:

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

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

If there are problems with the debug port signals

❏ Check for pull-down resistors on DSDI and DSCK.

Some target systems may have 220 ohm pull downs on these two signals. These signals are series terminated by the target interface module with a 46 ohm resistor. A 220 ohm pull-down would present a 20% drop in signal level when driven high, which could easily cause some malfunctions. There should be a very weak pull down on the target, if any at all. If you want to pull-down DSCK, use a value of 2.2K or greater.

To test the target system

The following program can be placed into memory. The opcode 0x4bfffff4 is a branch to a relative offset, so this program can be placed at any start address.

```
start: addi r1,1 - 0x38210001
nop - 0x60000000
nop - 0x60000000
bra start - 0x4bfffff4
```

Use 3fa004 instead of 10000 as the start address if you are using the Axiom CME-0555 Single Board Computer as your target system.

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

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

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.

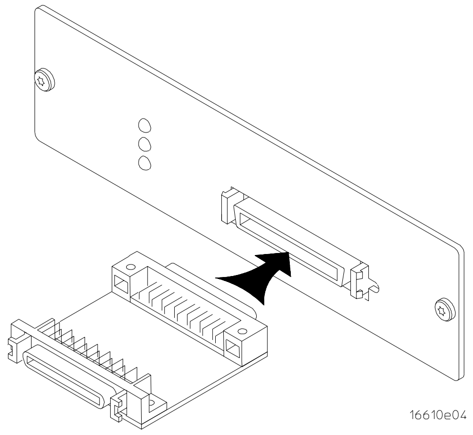
Solving Emulation Module Problems

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.07.51 17Dec97
    Location:  Generics

  E3497A Motorola MPC800 Embedded PowerPC Emulator
    Version:   A.01.02 18Dec97
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 PowerPC target system. You may also get a "?>" prompt for the same reason, and this is normal and expected. Any errors after the "PASSED Number of tests: 1 Number of failures: 0" line can be ignored.

If a performance verification test fails

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

If the particular failure you see is not listed below, contact 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

13

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 MPC555 target interface module.

Operating Characteristics	
Microprocessor Compatibility	Motorola MPC505, MPC509, and MPC555 Embedded PowerPC 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

14

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
E3456-69401	Programmed emulation probe

Emulation Module Replaceable Parts

Agilent Part Number	Description
0950-3043	Power Supply
E3496-61603	10-pin target cable
E3496-61601	50-pin cable
E3456-60001	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.

Glossary

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

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

Debug Mode See *Background Debug Monitor*.

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

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

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

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

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

Emulation Solution A set of tools for debugging your target system. A solution includes probing, inverse assembly, the B4620B Source Correlation Tool Set, and an emulation module.

Emulator An emulation module or an emulation probe.

Extender A part whose only function is to provide connections from one location to another. One or more extenders might be stacked to raise a probe above a target microprocessor to avoid mechanical contact with other components installed close to the target microprocessor. Sometimes called a "connector board."

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

Gateway Address An IP address entered in integer dot notation. The default gateway address is 0.0.0.0, which allows all connections on the local network or subnet. If connections are to be made across networks or subnets, this address must be set to the address of the gateway machine.

General-Purpose Flexible Adapter A cable assembly that connects the signals from an elastomeric probe adapter to an analysis probe. Normally, a male-to-male header or transition board makes the connections from the general-purpose flexible adapter to the analysis probe.

High-Density Adapter Cable A cable assembly that delivers signals from an analysis probe hardware interface to the logic analyzer pod cables. A high-density adapter cable has a single *MICTOR connector* that is installed into the analysis probe, and two cables that are connected to corresponding odd and even logic analyzer pod cables.

High-Density Termination Adapter Cable Same as a High-Density Adapter Cable, except it has a termination in the *MICTOR connector*.

In Background, In Monitor See *Background Debug Monitor*.

Inverse Assembler Software that displays captured bus activity as assembly language mnemonics. In addition, inverse assemblers may show execution history or decode control busses.

IP address Also called Internet Protocol address or Internet address. A 32-bit network address. It is usually represented as decimal numbers separated by periods; for example, 192.35.12.6.

Jumper Moveable direct electrical connection between two points.

JTAG (OnCE) port See *debug port*.

Label Labels are used to group and identify logic analyzer channels. A label consists of a name and an associated bit or group of bits.

Link-Level Address The unique address of the LAN interface. This value is set at the factory and cannot be changed. The link-level address of a particular piece of equipment is often printed on a label above the LAN connector. An example of a link-level address in hexadecimal: 0800090012AB. Also known as an LLA, Ethernet address, hardware address, physical address, or MAC address.

Mainframe Logic Analyzer A logic analyzer that resides on one or more board assemblies installed in a 16500, 1660-series, or 16600/700-series mainframe.

Male-to-male Header A board assembly that makes point-to-point connections between the female pins of a flexible adapter or transition board and the female pins of an analysis probe.

MICTOR Connector A high-density matched impedance connector manufactured by AMP Corporation. *High-density adapter cables* can be used to connect the logic analyzer to MICTOR connectors on the target system.

Monitor, In See *Background Debug Monitor*.

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

Preprocessor See *Analysis Probe*.

Preprocessor Interface See *Analysis Probe*.

Probe Adapter See *Elastomeric Probe Adapter*.

Processor Probe See *Emulation Probe*.

Run Control Probe See *Emulation Probe* and *Emulation Module*.

Setup Assistant Wizard software program which guides a user through the process of connecting and configuring a logic analyzer to make measurements on a specific microprocessor. The setup assistant icon is located in the main system window.

Shunt Connector. See *Jumper*.

Solution A set of tools for debugging your target system. A solution includes probing, inverse assembly, the B4620B Source Correlation Tool Set, and (optionally) an emulation module.

Stand-Alone Logic Analyzer A standalone logic analyzer has a predefined set of hardware components which provide a specific set of capabilities. A standalone logic analyzer differs from a mainframe logic analyzer in that it does not offer card slots for installation of additional capabilities, and its specifications are not modified based upon selection from a set of optional hardware boards that may be installed within its frame.

State Analysis A mode of logic analysis in which the logic analyzer is configured to capture data synchronously with a clock signal in the target system.

Subnet Mask A subnet mask blocks out part of an IP address so the networking software can determine whether the destination host is on a local or remote network. It is usually represented as decimal numbers separated by periods; for example, 255.255.255.0.

Symbol Symbols represent patterns and ranges of values found on labeled sets of bits. Two kinds of symbols are available:
1) Object file symbols — Symbols from your source code, and symbols generated by your compiler. Object file symbols may represent global variables, functions, labels, and source line numbers.
2) User-defined symbols — Symbols you create.

Target Board Adapter A daughter board inside the E5900B emulation probe which customizes the emulation probe for a particular microprocessor family. The target board adapter provides an interface to the ribbon cable which connects to the debug port on the target system.

Target Control Port An 8-bit, TTL port on a logic analysis system that you can use to send signals to your target system. It does not function like a pattern generator or emulation module, but more like a remote control for the target's switches.

Target Interface Module A small circuit board which connects the 50-pin cable from an E5901A emulation module or E5900A emulation probe to signals from the debug port on a target system. Not used with the E5900B emulation probe.

TIM See *Target Interface Module*.

Timing Analysis A mode of logic analysis in which the logic analyzer is configured to capture data at a rate determined by an internal sample rate clock, asynchronous to signals in the target system.

Transition Board A board assembly that obtains signals connected to one side and rearranges them in a different order for delivery at the other side of the board.

Trigger Specification A set of conditions that must be true before the instrument triggers. See the printed or online documentation of your logic analyzer for details.

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

Index

Index

Index

Restricted Rights Legend

Document Warranty

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.

About this edition

This is the *Solutions for the Motorola MPC555 User's Guide*.

Publication number
E2489-97004, July 2001
Printed in USA.

Print history is as follows:
E2489-97003, November 2000
E2489-97002, October 2000
E2489-97001, June 2000
E2489-97000, December 1998
E2489-97001, 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.