# Help Volume

 $\ensuremath{\mathbb{O}}$  1992-2001 Agilent Technologies. All rights reserved.

System: Measurement Examples

## **Measurement Examples**

These quick reminders on how to perform common measurements are grouped by the development phase in which the measurement typically occurs:

- "Hardware Turn-On" on page 10 Hardware designers take a loaded first cut printed-circuit board and verify its basic operation before delivering it to driver writers and software developers.
- "Firmware Development" on page 143 Given a printed-circuit board that has been turned-on, firmware developers create drivers and operating system calls that control and communicate with the hardware. They deliver stable hardware with a low-level software interface to application software developers.
- "Software Development" on page 198 Given stable hardware and lowlevel driver software, software developers verify real-time application software execution.
- "System Integration" on page 257 When system problems are discovered, system integrators determine whether the problem is being caused by hardware, software, or both. Also, they analyze system performance.

# See Also "Contents" on page 6 "Measurement Tips & Tricks" on page 300

Main System Help (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Glossary (see page 311)

### Contents

### **Measurement Examples**

#### **1** Measurement Examples

Contents 6

Hardware Turn-On 10 Looking at Signal Parameters 11 Looking at Signal Edges, Patterns, and Glitches 16 Measuring Conformance to Specifications 50 Looking at State Events 96 Exercising the Microprocessor (with the Emulation Probe) 136

Firmware Development 143 Testing Boot Code (with the Emulation Probe) 143 Making Driver Development Measurements 155 Making Interrupt Service Routine Measurements 183

Software Development 198 Analyzing Real-Time Software Execution 198 Analyzing Real-Time Variable Access 229 Analyzing Real-Time Memory Usage 250

System Integration257Making Cross-Domain Measurements257Making System Profile Measurements282Isolating Critical Defects288

### Contents

Measurement Tips & Tricks 300 Setting up 16715/16/17/18/19A triggers 300 Setting up triggers in other logic analyzers 302 Use trigger functions for easy measurement set up 305 Modify trigger functions to build new measurements 307 Know how processor execution affects measurements 308 Getting the most out of trace memory 309 If the trigger doesn't occur as expected 309

#### Glossary

Index

Measurement Examples

Chapter 1: Measurement Examples **Contents** 

## Contents

Hardware Turn-On	Looking at Signal Parameters
	• "To make basic oscilloscope measurements" on page 11
	Looking at Signal Edges, Patterns, and Glitches
	• "To trigger on a stable pattern" on page 16
	• "To find edges that are too close or too far" on page 20
	• "To find the Nth transition of a signal" on page 24
	• "To find when a signal or pattern stops" on page 28
	• "To delay capture after a pattern" on page 32
	• "To find an edge during a valid pattern" on page 36
	• "To find a pattern, an edge, and another pattern" on page 40
	• "To find signal glitches" on page 45
	Measuring Conformance to Specifications
	• "To measure conformance to specs (with the Compare tool)" on page 51
	• "To find setup and hold violations" on page 56
	• "To trigger if a pattern doesn't follow an edge" on page 59
	• "To verify pulse widths" on page 63
	• "To trigger on a violation of an edge sequence" on page 67
	• "To trigger when two edges are asserted simultaneously" on page 71
	• "To generate pattern stimulus on devices" on page 75
	• "To analyze jitter or time dispersion (with SPA)" on page 81
	• "To analyze bus stability (with SPA)" on page 88
	Looking at State Events
	• "To trigger on the Nth occurrence of an event" on page 96

• "To trigger on the Nth occurrence of an event" on page 96

# Chapter 1: Measurement Examples Contents

- "To store N samples of an event" on page 100
- "To trigger on a sequence of events" on page 105
- "To trigger when a program loop exits" on page 111
- "To find events that are too close or too far" on page 116
- "To count occurrences of an event between two events" on page 120
- "To trigger on a function call sequence" on page 125
- "To analyze bus occupation & bandwidth (with SPA)" on page 131

Exercising the Microprocessor (with the Emulation Probe)

- "To initialize registers, access memory" on page 137
- "To use the emulation probe as a test tool" on page 140

Testing Boot Code (with the Emulation Probe)

- "To download boot code" on page 144
- "To start or stop processor execution" on page 147
- "To stop processor execution using breakpoints" on page 149
- "To capture startup execution" on page 152

Making Driver Development Measurements

- "To trigger on an 8-bit serial pattern" on page 155
- "To view serial data in parallel" on page 160
- "To capture driver execution (& view HW and SW)" on page 165
- "To capture execution up to a failure or halt" on page 171
- "To view bus activity" on page 174
- "To capture simple program messages" on page 175
- "To trigger on packet data (with DataComm Analysis)" on page 177 Making Interrupt Service Routine Measurements
- "To capture interrupt frequency and type" on page 183

Firmware Development

# Chapter 1: Measurement Examples **Contents**

- "To measure interrupt latency and execution time" on page 186
- "To simulate particular interrupt sequences" on page 191
- "To view the occurrence rate of an event (with SPA)" on page 192

Software Development Analyzing Real-Time Software Execution

- "To trace about a source line" on page 199
- "To trace function flow" on page 203
- "To trace callers of a function" on page 206
- "To trace execution within a function" on page 210
- "To measure function execution time" on page 214
- "To measure function execution time (with SPA)" on page 218
- "To omit monitor cycles from the trace" on page 223
- "To stop execution at a source line (in ROM)" on page 226

Analyzing Real-Time Variable Access

- "To find NULL pointer de-references" on page 229
- "To trace a variable's values" on page 231
- "To find where variables are accessed from" on page 236
- "To trace before a variable value" on page 240
- "To stop execution on a corrupt variable" on page 245

Analyzing Real-Time Memory Usage

- "To monitor stack or heap usage" on page 251
- "To find stack overflow or guarded memory access" on page 255

**System Integration** 

- Making Cross-Domain Measurements
  - "To capture software execution when a scope triggers" on page 258
  - "To generate patterns when a source line executes" on page 262
  - "To arm one logic analyzer with another's trigger" on page 266

### Chapter 1: Measurement Examples Contents

- "To arm a state machine with a timing machine trigger" on page 271
- "To arm an oscilloscope when the analyzer triggers" on page 277

Making System Profile Measurements

- "To isolate the root cause of a performance bottleneck" on page 283
- "To simulate bus occupation and measure SW performance" on page 287

Isolating Critical Defects

- "To capture SW execution on a setup or hold violation" on page 289
- "To trigger an oscilloscope when a source line executes" on page 294

**Measurement Tips &** • "Setting up 16715/16/17/18/19A triggers" on page 300 Tricks

- "Setting up triggers in other logic analyzers" on page 302
- "Use trigger functions for easy measurement set up" on page 305
- "Modify trigger functions to build new measurements" on page 307
- "Know how processor execution affects measurements" on page 308
- "Getting the most out of trace memory" on page 309
- "If the trigger doesn't occur as expected" on page 309

# Hardware Turn-On

Looking at Signal Parameters

- "To make basic oscilloscope measurements" on page 11 Looking at Signal Edges, Patterns, and Glitches
- "To trigger on a stable pattern" on page 16
- "To find edges that are too close or too far" on page 20
- "To find the Nth transition of a signal" on page 24
- "To find when a signal or pattern stops" on page 28
- "To delay capture after a pattern" on page 32
- "To find an edge during a valid pattern" on page 36
- "To find a pattern, an edge, and another pattern" on page 40
- "To find signal glitches" on page 45

Measuring Conformance to Specifications

- "To measure conformance to specs (with the Compare tool)" on page 51
- "To find setup and hold violations" on page 56
- "To trigger if a pattern doesn't follow an edge" on page 59
- "To verify pulse widths" on page 63
- "To trigger on a violation of an edge sequence" on page 67
- "To trigger when two edges are asserted simultaneously" on page 71
- "To generate pattern stimulus on devices" on page 75
- "To analyze jitter or time dispersion (with SPA)" on page 81
- "To analyze bus stability (with SPA)" on page 88

Looking at State Events

• "To trigger on the Nth occurrence of an event" on page 96

- "To store N samples of an event" on page 100
- "To trigger on a sequence of events" on page 105
- "To trigger when a program loop exits" on page 111
- "To find events that are too close or too far" on page 116
- "To count occurrences of an event between two events" on page 120
- "To trigger on a function call sequence" on page 125
- "To analyze bus occupation & bandwidth (with SPA)" on page 131

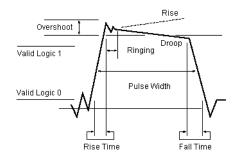
Exercising the Microprocessor (with the Emulation Probe)

- "To initialize registers, access memory" on page 137
- "To use the emulation probe as a test tool" on page 140

### Looking at Signal Parameters

• "To make basic oscilloscope measurements" on page 11

#### To make basic oscilloscope measurements



Possible uses:

- To measure the analog parameters of signals.
- To trace to the root cause of noise, crosstalk, or ground bounce problems when combined with a logic analyzer.

Probing the Target System

- 1. Connect the oscilloscope channel probes to signals of interest in the target system.
  - 2. Display the oscilloscope window.



3. Select the Channels tab, and set up the channels.

Channels	Memories	Measure   Markers
Setup.	<mark>Channel C1</mark> ✔ 0n	Channel C2 0ff
Scale:		Π 1.00 V/div Π
Offset:	2.50 V	2.50 V

Channel Setup – Scope <c></c>				
*				
Г Г	Channel C1	Channel C2		
	🔽 0n	📕 0ff		
Name:	Channel C1	Channel C2		
Probe:	10.00 :1 A	10.00 :1 A		
	🔶 1M Ohm / DC	◆ 1M Ohm / DC		
Input Z / Coupling:	💠 1M Ohm / AC	♦ 1M Ohm / AC		
Cooping.	♦ 50 Ohm / DC			
Skew:	0 s	◊s ▲		
Preset:	TTL =	TTL =		
Scale:	1.00 V/div <u>1</u>	1.00 V/div 1		
Offset:	2.50 V	2.50 V		

### **Capturing the Data** 1. Set up the trigger.

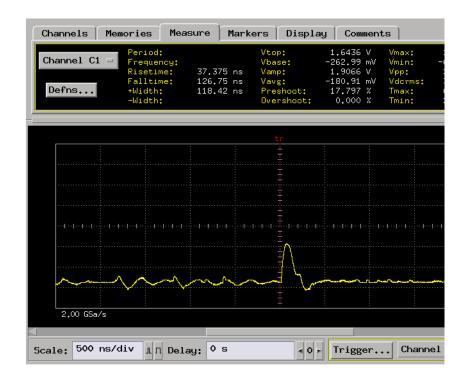
Trigger	: ••••• Channel		
-	Trigger Setu	p – Scope <c:< th=""><th>&gt;</th></c:<>	>
*			
Mode	Sweep		
🔷 Edge	🔶 Auto		
🔷 Pattern	$\diamond$ Triggered		
$\diamond$ Immediate			
Source:	Level:	Slope:	Occurrences:
Channel C1 -	1.62 V	× F	1 ×

2. Select the Run button to capture an oscilloscope trace.

You may want to change the time/div scale and select the Run button again to capture data with a different sample rate.



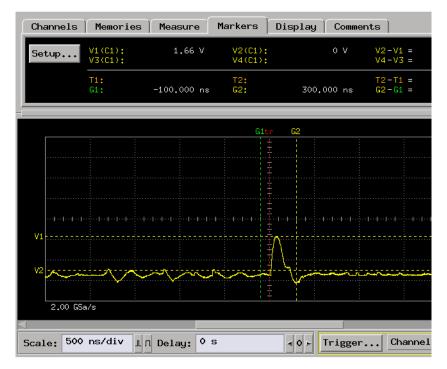
**Displaying the Data** 1. Select the Measure tab to view the data on the analog parameters of the captured signal.



2. Select the Markers tab to set up voltage and time markers on the display.

Channels	Memories	Measure	Markers
Setup. <sub>N</sub> .	V1(C1): V3(C1):		V2(C1): V4(C1):

Voltage	e Markers		-Global	Time Markers	
٧1	1.66 V	Channel C1 -	G1	-100,000 ns	<0 ► frc
₩2	0 V	Channel C1 -	G2	300.000 ns	∢0 ⊨ fro

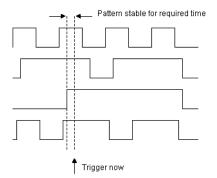


See Also"To capture software execution when a scope triggers" on page 258"To arm an oscilloscope when the analyzer triggers" on page 277"To trigger an oscilloscope when a source line executes" on page 294

## Looking at Signal Edges, Patterns, and Glitches

- "To trigger on a stable pattern" on page 16
- "To find edges that are too close or too far" on page 20
- "To find the Nth transition of a signal" on page 24
- "To find when a signal or pattern stops" on page 28
- "To delay capture after a pattern" on page 32
- "To find an edge during a valid pattern" on page 36
- "To find a pattern, an edge, and another pattern" on page 40
- "To find signal glitches" on page 45

### To trigger on a stable pattern



#### Possible uses:

- To wait for all status lines to finish transitioning before triggering.
- To filter out spurious triggers because of transitions that occur when the target system's state machine is indeterminate.

Probing the Target System

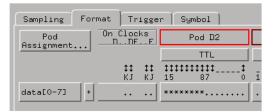
- Target1. Connect the logic analyzer probes to the signals on which you will look for<br/>the pattern.
  - 2. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling     Format     Trigger     Symbol       Analyzer Name:     Analyzer (E)     Image: On the symbol     Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
- Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

3. Assign pods if necessary.

Sampling Format Pod Assignment			
-		Pod Assignment	
Analyzer 1		Analyzer 2	
Name: Analyzer <e></e>	Name:	Analyzer <e2></e2>	
Type: Timing 😐	Type:	Off -	Unassigned Pods
E1: *************	E3:	‡‡ <sup>+</sup> ‡‡‡‡_‡ L ‡	D3:L _
<b>Е2:</b> ‡‡‡‡‡‡‡‡‡‡‡ К ‡	E4:	M _	D4:M _
<b>D1:‡‡‡‡‡‡‡‡‡‡‡‡‡</b>			
D2: *****************************			

4. Format a label for the signals on which you will look for a stable pattern.



**Capturing the Data** 1. Use the "Find pattern present for > duration" trigger function.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Status Save/Rec.	all]
General Timing	Trigger function libraries.
Find Nth occurrence of an edge Find pattern present for > duration Find pattern present for > duration Find pattern absent for > duration Find pattern absent for < duration	pattern
Replace Insert before Insert	after Delete

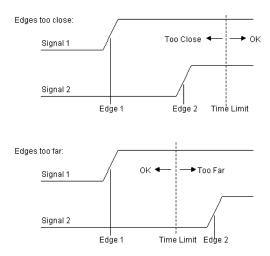
2. In the trigger definition, specify the pattern, and enter the time that pattern must be stable for.

Trigger definition
1 FIND PATTERN PRESENT FOR > DURATION
Find data[0-7] = 2D Hex
present for > 6 ns
then Trigger and fill memory

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. Use the Waveform display to verify that the pattern was stable for the specified time before the trigger.

Expander Card (Master: E)	data[0-7] all 74 2 Insert before		
Select ->	Insert after		
16717A Setup	TimingZoom		
333MHz St. Waveform<1>	Replace		
Timing Zoom<1>	Delete		
Listing(1)	Expand 🔺		
Source Viewer(1)	Overlag		
Run 🖻	Change attributes		
Search   Goto   Markers   Comments			
	Analysis   Mixed Signal		
G1: data[0-7]	from Trigger ± = −6.078 ns		
G2: data[0-7]	from Trigger <b>⊥</b> = 28.235 ns		
Seconds/div - 10.000 ns	Delay 🕅 s		
	G1 tr G2		
data[0-7] 0 0	1		
data[0-7] 1	0		
data[0-7] 2 0	1		
data[0-7] 3 0	1		
data[0-7] 4	9		
data[0-7] 5 0	1		
data[0-7] 6	0		
data[0-7] 7	0		

### To find edges that are too close or too far



Possible uses:

• To check DRAM row/column address strobe timing.

Probing the Target System

- 1. Connect logic analyzer probes to the signals whose edges you wish to look at.
  - 2. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run 🖻
Sampling Format Trigger Symbol Analyzer Name: Analyzer(E) 7 On 2012 Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

3. Assign pods if necessary.

Sampling Format

Assign	ment			
			Pod Assignment	
	-Analyzer 1		-Analyzer 2	
Name:	Analyzer <e></e>	Name:	Analyzer <e2></e2>	
Type:	Timing -	Type:	Off 🖃	Unassigned Pods
E1: ‡	: <b>************</b> ****** J 1	E3:	‡‡*‡‡‡_‡ L ‡	D3:L _
<b>E2:</b> ‡	атттттттттттттт	E4:	M _	D4:M _
D1: _	_ <b>*********</b> _ <b>***</b> J \$	:		
D2: ‡	***********************	¢		

4. Format labels for the signals of interest.

Sampling Format Trigger Symbol									
Pod Assignment		On Clocks DDFF		Pod D2		Pod D1		Pod E2	
Insergnmente.	••			TTL		TTL		TTL	
		‡‡ KJ	‡‡ KJ	<b>1</b> 5 87	‡	15 87	‡‡‡ 0	15 87	
addr[0-31]	+	••	•••				••••	******	******
*MEM_OE	+	•••		*			••••		•••••
*CS1	+	••				*	••••		•••••
*MEM_WE	+	••		*			••••		•••••
data[0-7]	+			*******					

### **Capturing the Data** 1. Use the

1. Use the "Find 2 edges too close together" or the "Find 2 edges too far apart" trigger function.

Sampling Format Trigger Symbol							
Trigger Functions Settings Overview Status Save/Recall							
General Timing	Trigger function libraries						
Find pattern absent for < duration Run until user stop Find 2 edges too close together Find 2 edges too far apart Find pattern occurring too soon after edge	edge 1 edge 2						
Replace 🕨 🛛 Insert before	Insert after Delete						

2. In the trigger definition, select edges and time.

Trigger definition						
1 FIND 2 EDGES TOO CLOSE TOGETHER						
Find <u>*CS1</u> Edge ↓						
followed by <u>*MEM_WE</u> Edge <u>†</u>						
occurring within 120 ns						
then Trigger and fill memory						

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. Open the Waveform display and use the global markers to show the time between the edges.

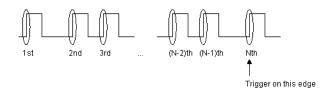
Expander Card (Master: E)						
Select						
16717A	Setup					
333MHz St	Waveform<1>					
	Timing Zoom<1>					
	Listing<1>					
	Source Viewer<1>					
	Run 🕨					

Search   Goto	Markers Co	omments   A	nalysis   Mixed Si	gnal ]	
G1: data[0-7]	<u>↓</u> = 18	Time 🛓 fi	rom Trigger 👤 =	-114.00	00 ns
G2: data[0-7]	<u>↓</u> = 05	Time 👤 fi	rom Trigger <u>↓</u> =	13.600	ns 🗸
Seconds/div 🗆	20,000 ns	Del.	ay 1-60,000 ns 🔺		
	L G	1 , ,		tı	G2
addr[0-31] all	G FFF02E73	1	00004018		G2 00004019
addr[0-31] all *MEM_OE all			00004018		
Þ			00004018		
*MEM_OE all	FFF02E73		1		00004019
*MEM_OE all *CS1 all	FFF02E73		1 0 0		00004019

#### See Also

"Use trigger functions for easy measurement set up" on page 305

### To find the Nth transition of a signal



Possible uses:

- To find the 3rd occurrence of the start of a data transfer.
- To find the 1000th occurrence of a chip select line being asserted.

Probing the Target System

- 1. Connect a logic analyzer probe to the signal of interest.
- 2. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run 🖻
Sampling Format Trigger Symbol Analyzer Name: Analyzer(E) 7 On ZGHz Parton Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
Acquisition Depth 2M 🛓
Sample Period 3.0ns

3. Assign pods if necessary.

Sampling Format

Assign	nment			
			Pod Assignment	
	-Analyzer 1		-Analyzer 2	
Name:	Analyzer <e></e>	Name:	Analyzer <e2></e2>	
Type:	Timing 🖃	Type:	Off -	Unassigned Pods
E1: ‡	t <b>ttttttttttt</b> t	E3:	*\$	D3:L _
E2: 1	<b>;;;;;;;;;;;;</b> ;;;	E4: _	M _	D4:M _
D1:	_ <b>_‡‡‡‡‡‡‡‡‡‡</b> _ <b>‡‡‡</b> J \$	:		
D2: ‡	**********************	:		

4. Format labels for the signals of interest.

Sampling Format Trigger Symbol									
Pod Assignment		On Clocks DDFF		Pod D2		Pod D1		Pod E2	
Insergnmente.	••			TTL		TTL		TTL	
		‡‡ KJ	‡‡ KJ	<b>1</b> 5 87	‡	15 87	‡‡‡ 0	15 87	
addr[0-31]	+	••	•••				••••	******	******
*MEM_OE	+	•••		*			••••		•••••
*CS1	+	••				*	••••		•••••
*MEM_WE	+	••		*			••••		•••••
data[0-7]	+			*******					

**Capturing the Data** 1. Use the "Find Nth occurrence of an edge" trigger function.

Sampling Format	Trigger Symbol							
Trigger Functions Set	Trigger Functions Settings Overview Status Save/Recall							
General Timing		Trigger	function libraries					
Find pattern Find edge Find edge AND pattern Find width violation of Find Nth occurrence of		Occurrence 1 edge	Occurrence					
Replace 📐	Insert before	Insert after	Delete					

2. In the trigger definition, select the edge label and number of occurrences.

Trigger definition						
1 FIND NTH OCCURRENCE OF AN EDGE						
Find 4						
<u>×MEM_OE</u> Edge ↑						
then Trigger and fill memory						

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. In the Waveform window (depending on the time between signal transitions) you may be able to see that you've triggered on the Nth transition of the signal.

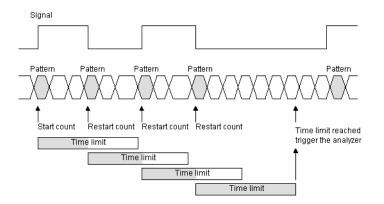
Expander Card (Master: E)							
Select							
16717A	Setup						
333MHz St	Waveform<1> 🖌						
	Timing Zoom<1>						
	Listing<1>						
	Source Viewer<1>						
	Run 🕨						

Search   Goto   Markers   Comments   Analysis   Mixed Signal				
61: ★CS1				
G2: ★CS1				
Seconds/div - 100.000 ns Delay 0 s				
C4 00				
G1 tr G2				
addr[0-31] all FFF02FBA •02FB0 •02FB1 •02FB2 FFF02FB3 •02FB5				
addr[0-31] all FFF02FAB •02FB0 •02FB1 •02FB2 FFF02FB3 •02FB4 •02FB5				
addr[0-31] all         FFF02FAB         •02FB0         •02FB1         •02FB2         FFF02FB3         •02FB4         •02FB5           *MEM_0E all         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0				
addr[0-31] all         FFF02FAB         •02FB0         •02FB1         •02FB2         FFF02FB3         •02FB4         •02FB5           *MEM_DE all         0         1         0         0         1         0         0         0         1         0         0         0         1         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         1         0         0         0         0         0         1         0				

#### See Also

"Use trigger functions for easy measurement set up" on page 305

### To find when a signal or pattern stops



You can count time by counting occurrences of sampled data or by using a timer.

Possible uses:

- To find when signals are inactive for too long a time.
- To check when execution leaves an address range.
- To check when expected variable values stop being written.

• To capture what leads up to an unexpected condition.

Probing the Target1. Configure a timing machine to look at signal edges or patterns.System

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform (1)
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling     Format     Trigger     Symbol       Analyzer Name:     Analyzer     Analyzer     Image: Compare the symbol
$\blacklozenge$ Timing Mode – Asynchronous sampling clocked internally by analyzer $\diamondsuit$ State Mode – Synchronous sampling clocked by the Device Under Test
-Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M <u>+</u>
Sample Period 3.0ns

2. Assign pods if necessary.

Sampling Format

Pod Assignment.					
	Pod Assignment				
Analyzer 1	Analyzer 2				
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>				
Type: Timing -	Type: Off 🗆	Unassigned Pods			
E1: **************	E3:‡‡-‡‡‡‡L ‡	D3:L _			
E2: <sup>‡‡‡‡‡‡‡‡‡‡‡</sup> _‡ K ‡	E4: M _	D4:M _			
D1:**********					
D2: *****************					

3. Format labels for the signals or patterns of interest.

Sampling Format Trigger Symbol						
Pod		Cloc D DF		Pod D2	Pod D1	Pod E2
Assignment		<u>.</u>		TTL	TTL	TTL
		‡‡ KJ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡</b> _ <b>‡‡‡</b> 15 87 0	<b>***********</b> 15 87 0 1
addr[0-31]	+	•••				******
*MEM_OE	+	••	•••	*		
*CS1	+	•••			*	
*MEM_WE	+	••		·····		
data[0-7]	+	•••		*****		
*TS	+	••			*	
*TA	+	••	*.			
AT2	+	•••			*	

**Capturing the Data** 1. Use the "Find pattern absent for > duration" trigger function.

Sampling Format Trigger Symbol					
Trigger Functions Settings Overview Status Save/Recall					
General Timing	Trigger function libraries				
Find Nth occurrence of an edge Find pattern present for > duration Find pattern present for < duration Find pattern absent for > duration Find pattern absent for < duration	NOT pattern				
Replace 📐 Insert before 🛛 Insert	after Delete				

2. In the trigger definition, specify the pattern and the amount of time absent.

Trigger definition					
FIND PATTERN ABSENT FOR > DURATION					
Find *TA = 0 Hex					
absent for > 4.002 us					
then Trigger and fill memory					

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. In the Waveform display (depending on the time between the edge/pattern and the trigger) you may be able to see the last time the edge/pattern occurred.

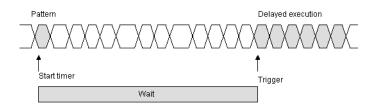
Expand	ler Card (Master: E)				
Select	->				
16717A	Setup				
333MHz St.	Waveform<1>4				
	Timing Zoom<1>				
	Listing<1>				
	Source Viewer<1>				
	Run Þ				

Search   Goto	Markers	Comments A	Analysis   Mixed Signal
G1: *CS1		Time 🛓 fr	rom Trigger
G2: *CS1	<u>↓</u> = 1	Time 🛓 fr	rom Trigger 🛓 = 5,320 us
Seconds/div -		s 🖌 Del	lay 10 s
	G1		tr G2
addr[0-31] all		FFF00507	
*MEM_OE all		1	
*CS1 all			1
*MEM_WE all			1
data[0-7] all		FF	
*TS all		1	
*TA all		1	
AT2 all	0	1	•

#### See Also

"Use trigger functions for easy measurement set up" on page 305

### To delay capture after a pattern



#### Possible uses:

- To hold off the trigger and look at control signals later than when the address bus pattern becomes invalid.
- To look for a receiver's response which is supposed to occur 3 milliseconds after a transmission.

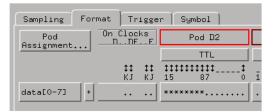
Probing the Target System 1. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling     Format     Trigger     Symbol       Analyzer Name:     Analyzer (E)     Image: On the symbol     Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
- Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

2. Assign pods if necessary.

Sampling Format Pod Assignment						
-	Pod Assignment					
Analyzer 1		Analyzer 2				
Name: Analyzer <e></e>	Name:	Analyzer <e2></e2>				
Type: Timing -	Type:	Off -	Unassigned Pods			
E1: **************	E3:	‡‡~‡‡‡‡_‡ L ‡	D3:L _			
E2: <sup>‡‡‡‡‡‡‡‡‡‡‡</sup> _‡ K ‡	E4:	M _	D4:M _			
D1:**********						
D2: ****************************						

3. Format a label for the signals on which you will look for a stable pattern.



**Capturing the Data** 1. Build a two level trigger setup using the "Find pattern present for > duration" and "Wait t seconds" trigger functions.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Status Sa	ve/Recall]
General Timing	Trigger function libraries
Find pattern occurring too soon after edge Find pattern occurring too late after edge Mait t seconds Wait for arm in	t sec
Replace Insert before	Insert after 🛛 🔹 Delete

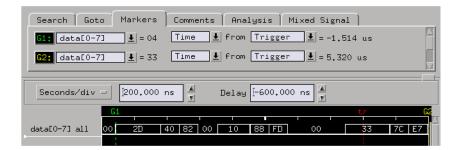
Sampling Format Trigger Symbol					
Trigger Functions Settings Overview Status Save/Recall					
General Timing	Trigger function libraries				
Find Nth occurrence of an edge Find pattern present for > duration Find pattern present for < duration Find pattern absent for > duration Find pattern absent for < duration	pattern				
Replace Insert before 🔊 Inser	t after Delete				

2. In the trigger definition, specify the pattern and the amount of time present. Then, enter the amount of time to wait.

Trigger definition	
I       FIND PATTERN PRESENT FOR > DURATION         Find data[0-7]       = 2D         present for > 6       ns         then       Goto         Next	
2 WAIT T SECONDS	
Wait 1.5 us	
then Goto Next Insert ACTION >	
Goto	
Trigger 🗈 Trigger a	nd fill memory
Trigger a	nd goto

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. In the Waveform display (depending on the time between the pattern and the trigger) you may be able to see the last time the pattern occurred.

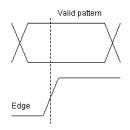
Expander Card (Master: E)		
Select	->	
16717A	Setup	
333MHz St	Waveform<1> 🖌	
	Timing Zoom<1>	
	Listing<1>	
	Source Viewer<1>	
	Run 🕨	



#### See Also

"Use trigger functions for easy measurement set up" on page 305

### To find an edge during a valid pattern



Possible uses:

- To capture a memory chip's select line at a given address.
- To view the timing of a write signal to a peripheral.

# Probing the Target1. Configure a timing analysis machine.System

n

Expander Card (Master: E)
Select $\rightarrow$
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run 🖻
Sampling     Format     Trigger     Symbol       Analyzer Name:     Analyzer (E)     Image: On Control of Control
Timing Mode – Asynchronous sampling clocked internally by analyzer
$\diamond$ State Mode – Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M <b>±</b>
Sample Period 3.0ns

Sampling Format Pod Assignment.			
		Pod Assignment	
Analyzer 1		Analyzer 2	
Name: Analyzer <e></e>	Name:	Analyzer <e2></e2>	
Type: Timing -	Type:	Off -	Unassigned Pods
E1: **************	E3:	‡‡~‡‡‡‡_‡ L ‡	D3:L _
E2: ++++++++++++++++++++++++++++++++++++	E4:	M _	D4:M _
D1:**********			
<b>D2: #########</b>			

3. Format one label for the signals on which you will look for a pattern and another label for the signal on which you will look for the edge.

Sampling	For	mat )	Tri	gger	Symbol	1	
Pod Assignment.		Clocks DDFF		Pod D2			
					TTL		
		<b>‡‡</b> КЈ	<b>‡‡</b> КЈ	<b>‡‡‡‡</b> 15	<b>*****</b> 87	‡	‡ 15
data[0-7]	+	••	•••	****	****	••••	•••
*MEM_WE	+	•••		••••	*	••••	

**Capturing the Data** 1. Use the "Find edge AND pattern" trigger function.

Sampling Format	Trigger   Symbol						
Trigger Functions Settings Overview Status Save/Recall							
General Timing			Trigger	function	libraries		
Find pattern Find edge Find edge AND pattern Find width violation of Find Nth occurrence of				pattern edge			
Replace 📐	Insert before	Insert	after		Delete		

2. In the trigger definition, specify the edge and the pattern.

Trigger definition
1 FIND EDGE AND PATTERN
Find <u>★MEM_WE</u> Edge ↓
and data[0-7] = 2D Hex
then Trigger and fill memory

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Waveform display to show the edge in relation to the pattern.

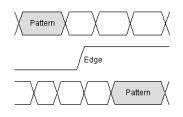
Expand	ler Card (Master: E)
Select	->
16717A	Setup
333MHz St	Waveform<1>≰
	Timing Zoom<1>
	Listing<1>
	Source Viewer<1>
	Run 🕨

Search Goto	Markers Comments Analysis Mixed Signal					
Goto Time 🛓	<u>∑0 s</u> <b>⊥</b> Goto					
Trigger B	eginning End G1 G2					
Seconds/div - 5.000 ns J Delay 0 s						
	1 tr G2					
data[0-7] all	2D 28 08					
*MEM_WE all	1 0					

If the analyzer never triggers, it could mean the pattern was never found or the edge never occurs when the pattern is valid.

See Also "Use trigger functions for easy measurement set up" on page 305 "If the trigger doesn't occur as expected" on page 309

### To find a pattern, an edge, and another pattern



Possible uses:

- To view a correct address bus, control signal, data bus sequence.
- To check whether a data packet was sent, a handshake signal followed, and an acknowledgement was returned.

Probing the Target System 1. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling     Format     Trigger     Symbol       Analyzer Name:     Analyzer (E)     Image: On the symbol     Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
- Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

Sampling Format

Assignment		
	Pod Assignment	
Analyzer 1 Name: Analyzer (E) Type: Timing = E1: ttttttttttttttt J t E2: ttttttttttttt_L K t D1:ttttttttttt J t I2: ttttttttttttt J t I2: tttttttttttttt J t	Analyzer 2 Name: Analyzer <e2> Type: Off E3:tt~ttt.t_t_ L t E4: M _</e2>	Unassigned Pods D3: L - D4: M -

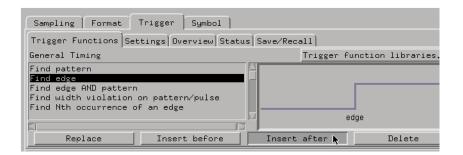
3. Format labels for the signals on which you will look for the edge and patterns.

Sampling Format Trigger Symbol									
Pod Assignment.,		On Clocks		On Clocks D. DF. F				Pod E2	
HSSIgnmenc.,	••			TTL	TTL	TTL			
		‡‡ KJ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	<b>***********</b> _*0			
addr[0-31]	+	••	••	•••••		****			
*MEM_OE	+	•••	••	*					
*CS1	+	••	••		**				
*MEM_WE	+	• •	••	·····					
data[0-7]	+	••	•••	******					

**Capturing the Data** 

1. Build a three level trigger setup using the "Find pattern", "Find edge", and "Find pattern" trigger functions.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Status Save/Recall	
General Timing Trigger f	function libraries.
Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge	urrence 1
Replace 🖌 Insert before 🛛 Insert after	Delete



Sampling Format Trigger Symbol		
Trigger Functions Settings Overview Status Save/Rec	all	
General Timing	Trigger f	unction libraries.
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge		irrence 1
Replace Insert before Insert	after 📐	Delete

2. In the trigger definition, specify the first pattern and the amount of time present. Then, specify the edge. Finally, specify the second pattern and the amount of time present.

Trigger definition	
FIND PATTERN Find addr[0-31] = 20000001 Hex then Goto Next	
2 FIND EDGE Find *HEM_WE Edge ↓ then Goto Next	
3 FIND PATTERN	
Find data[0-7] = 45 Hex	
then Goto Next Insert ACTION > Goto	
Trigger 🕑 Trigger	and fill memory
Trigger	and goto

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Waveform display to show the proper sequence was captured.

Expand	ler Card (Master: E)
Select	
16717A	Setup
333MHz St	Waveform<1>
	Timing Zoom<1>
	Listing<1>
	Source Viewer(1)
	Run 🕨

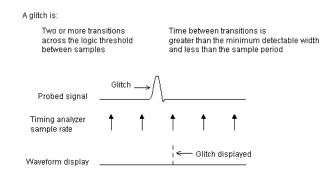
Search   Goto	Markers Comme	ents   Analysis   Mixed Signal
G1: *CS1	± = 1 Time	± from Trigger ± = -42.000 ns
G2: *CS1	L = 1 Time	. I from Trigger I = -3.200 ns
Seconds/div =	j20.000 ns ▲	Delay 🛛 s
	G	1 G2r
addr[0-31] all	FFF030E3	20000001
*MEM_OE all		1
*CS1 all		
*MEM_WE all	1	0
data[0-7] all	01	45

If the analyzer never triggers, the proper sequence does not occur. Depending on the level that was reached in the sequence above, you will need to set up a different trigger to see what actually occurs.

### See Also "Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

### To find signal glitches



Possible uses:

• To look for pulses more narrow than the minimum pulse width.

Probing the Target System

- 1. Connect the logic analyzer probes to the signals on which you will look for glitches.
  - 2. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run 🖻
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E)
♦ Timing Mode - Asynchronous sampling clocked internally by analyzer ♦ State Mode - Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

The sample period will specify what is interpreted as a glitch.

If the logic analyzer has a special acquisition mode for capturing glitches, select that mode.

**NOTE:** You must select the glitch capture mode in order to see the glitch symbol in the Waveform display.

Sampling Format Pod Assignment.		
-	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: Timing =	Type: Off =	Unassigned Pods
E1: **************	E3:‡‡-‡‡‡‡_‡ L ‡	D3:L _
E2: <sup>‡‡‡‡‡‡‡‡‡‡‡</sup> _ <sup>‡</sup> K ‡	E4:M _	D4:M _
<b>D1:‡‡‡‡‡‡‡‡‡‡‡‡‡</b>		
<b>D2: #########</b> ##########################		

4. Format labels for the signals of interest.

Sampling	Format	Tr	igge	r   S	ymbol ]		
Pod Assignment	- T	Cloc 1DF			Pod D2		
-HSSIgnmente	•••				TTL		
		‡‡ KJ	‡‡ KJ	<b>‡‡‡‡</b> 15	<b>*****</b>	‡	ī
data[0-7]	+	••		****	****		•

**Capturing the Data** You can trigger on anything, on any glitch, or on a particular glitch.

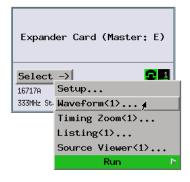
1. Use the "Find edge" trigger function.

Sampling Format Trigger Symbol			
Trigger Functions Settings Overview Status Save/Rec	all		
General Timing	Trigger	function	libraries
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge			
		edge	
Replace 📐 Insert before 🛛 Insert	after		Delete

2. In the trigger definition, specify the signal on which you will look for glitches.

Trigger	definition
1 FIND E	DGE
Find _	data[0-7] Edge •••••
then	Trigger and fill memory

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. Use the Waveform display to show the captured glitches.



G1:       data[0-7]       ■ = 43       Time       ■ from       Trigger       ■ = -15.000 ns         G2:       data[0-7]       ■ = 43       Time       ■ from       Trigger       ■ = 6.062 ns         Seconds/div       =       [3.000 ns       ■       ■       ■       ■         data[0-7]       0       1       1       ■       ■         data[0-7]       1       1       ■       ■
Seconds/div         3.000 ns         A         Delay         F6.000 ns         A           G1         tr         G2           data[0-7]         1         1           data[0-7]         1         1
G1         tr         G2           data[0-7]         0         1         1           data[0-7]         1         1         1
data[0-7] 0 1 1 1
data[0-7] 1 1
data[0-7] 2 0 1 0
data[0-7] 3 0
data[0-7] 4 0
data[0-7] 5 0
data[0-7] 6 1
data[0-7] 7 0 1 0

2. Use the TimingZoom display to get a better picture of the glitch.

Expand	er Card (Master: E)
Select	
16717A	Setup
333MHz Sta	Waveform<1>
	Timing Zoom<1>
	Listing<1>
	Source Viewer<1>
	Run 🗈

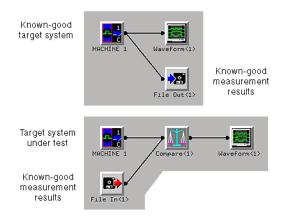
G1: data[0-7]	<b>±</b> = 43				
		Time 👤 from Tr	rigger 👤 = −1	15.000 ns	
G2: data[0-7]	<u>↓</u> = 43	Time 👤 from Tr	rigger <u>↓</u> = 6.	.062 ns	
Seconds/div =	]]. ]3,000 ns	🔺 Delay 🎽	5.000 ns ▲		
	(	G1	tı	-	G2
data[0-7]_TZ 0		1		0	
data[0-7]_TZ 1		1		0	
data[0-7]_TZ 2	0 1		0		
data[0-7]_TZ 3	0		0		
data[0-7]_TZ 4	0		0		
data[0-7]_TZ 5	0	0 1 0 1		)	
data[0-7]_TZ 6		1		0	
data[0-7]_TZ 7	0	1	0		

#### See Also

"To arm an oscilloscope when the analyzer triggers" on page  $277\,$ 

## Measuring Conformance to Specifications

- "To measure conformance to specs (with the Compare tool)" on page 51
- "To find setup and hold violations" on page 56
- "To trigger if a pattern doesn't follow an edge" on page 59
- "To verify pulse widths" on page 63
- "To trigger on a violation of an edge sequence" on page 67
- "To trigger when two edges are asserted simultaneously" on page 71
- "To generate pattern stimulus on devices" on page 75
- "To analyze jitter or time dispersion (with SPA)" on page 81
- "To analyze bus stability (with SPA)" on page 88



# To measure conformance to specs (with the Compare tool)

Possible uses:

- To measure specifications conformance against known-good circuitry.
- To measure specifications conformance under component stress conditions.

Probing the Target 1. Configure a timing or state machine. System

Expander Card (Master: E)
Select $\rightarrow$
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run
Sampling Format Trigger Symbol Analyzer Name: Analyzer(E) 7 On 2GHz Zoom Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

Pod Assignment.		
	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: Timing -	Type: Off 🖃	Unassigned Pods
E1: **************	E3:+++++++++++++++++++++++++++++++++	D3:L _
E2: <sup>‡‡‡‡‡‡‡‡‡‡‡</sup> _‡ K ‡	E4: M _	D4:M _
D1:**********		
<b>D2: ‡‡‡‡‡‡‡‡‡‡</b>		

3. Format labels for the signals of interest.

Sampling Format Trigger Symbol						
Pod Assignment		Clock N NF	ks F	Pod D2	Pod D1	Pod E2
Hastgrindite	•			TTL	TTL	TTL
		‡‡ KJ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡‡</b>	<b>‡‡‡‡‡‡‡‡±‡±‡</b>
addr[0-31]	+	••	••		•••••	*****
*MEM_OE	+	••	•••	*		
*CS0	+	••			·····*····	
*CS1	+	••	••		*	
*MEM_WE	+	••	•••	*		
data[0-7]	+	••	•••	*****		

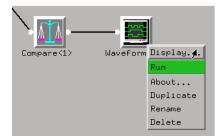
**Capturing the Data** 1. Set up a trigger specification.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Status Sa	we/Recall
General Timing	Trigger function libraries.
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge	Occurrence 1
Replace Insert before	Insert after Delete
Trigger definition	
1 FIND PATTERN	
Find addr[0-31] = FFF034D8 Hex	
then Trigger and fill memory	

- 2. Select the Run button to start the measurement.
- 3. In the Workspace window, add the Compare tool to your analyzer configuration and copy the known-good dataset to the Compare tool's *reference buffer*.

Analyzer (E) Waveform (1) Compare Setup Javeform (2) Run About Rename Delete	
♦ Disable Compare ◆ Enable Compare	
Compare Frame 10:Slot E:Analyzer <e> to Refere</e>	ence Buffer
Lopy Dataset to Reference Buffer	all matching labels label pairs

- 4. If you want to turn OFF the analyzer and probe a different target system, save the Compare tool configuration. This will save the contents of the Reference Buffer to the logic analyzer's disk.
- 5. Probe a different target system or add the component stress conditions.
- 6. If you saved a Compare tool configuration, load it.
- 7. Select the Run button to repeat the measurement.
- **Displaying the Data** 1. In the display window, differences in the measurement results will be highlighted with gray.



Seconds/div =	10,000 ns 🖌 Delay 10 s	
	1 tr	G
addr[0-31] all	FFF03187 FFF034	D8
*MEM_OE all	1 0	
*CSO all	1 0	
*CS1 all	1	
*MEM_WE all	1	
data[0-7] all	55	
addr[0-31]_ref all	FFF03187 FFF034D	8
*MEM_OE_ref all	1 0	
*CS0_ref all	1 0	
*CS1_ref all	1	
*MEM_WE_ref all		
data[0-7]_ref all	55	70
DiffFlag all	0 1 0	

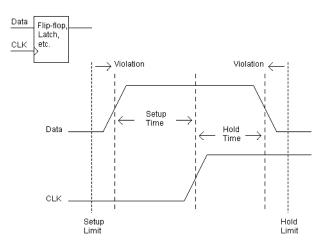
Note that a *difference flag* label is generated so you can search for differences.

See Also

The Compare tool online help (see the Compare Tool help volume) for

more information.

### To find setup and hold violations



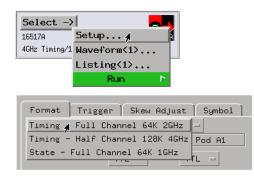
Possible uses:

• Verifying that design timing meets setup and hold specifications of flipflops, latches, and other memory element circuitry.

Requirements:

• The Agilent Technologies 16517A 4GHz Timing/1GHz State Logic Analyzer can look for setup and hold violations on multiple channels (for example, a data bus).

Probing the Target 1. Configure a timing analysis machine. System



2. Format labels for the signals of interest.



Capturing the Data1. In the Trigger window, replace level 1 of the trigger specification with the<br/>"Find setup or hold violation" macro. In the macro dialog, select the label,<br/>edge, setup time limit, and hold time limit.

Format Trigger Skew A Sample Period 500 ps	
Title Macro Sequence	
Edit Insert before Insert after Copy Delete Replace	Macro selection Replace level 1 with: Macros Find pattern occurring too soon Find pattern occurring too late Find setup or hold violation Find setup/hold violation clk'd
- Trig	ger sequence step #1

ļ	Trigger sequence step # 1
	Select New Macro
I	Find setup or hold violation
I	Find setup or hold violation on label(s) data[0-3]
I	clocked by "edge1"
I	setup time = 80ns 🔺 hold time = 50.00ns 🔺
	Setup violation Hold violation
I	
I	K−− 80 ns −− ★−− 50.000ns −− →
I	
I	"edge1"

2. Set up the edge resource.

Format Trigger Skew Adjust Symbol					
Sample Period 500 ps Trigger Position Center					
Title Macro Sequence					
Find setup/hold, "data[0-3]" then TRIGGER clock: "edge1", Tsetup: 80 ns, Thold: 50.00ns					
Pattern Edge					
edge1 <u>*MEM_WE</u> 1					

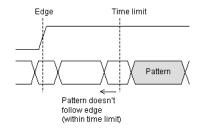
- 3. Select the Run button to start the measurement.
- Displaying the Data1. If the analyzer triggers, a setup or hold violation occurs. Open the<br/>Waveform display and use the global markers to see the actual setup or<br/>hold time.

Select ->   16517A   Setup   4GHz Timing/1   Waveform<1>   Listing<1>   Run	
Search       Goto       Markers       Comments       Analysis       Mixed Signal         G1:       *MEM_WE       ± = 0       Time       ± from       Trigger       ± = -70.588 ns         G2:       *MEM_WE       ± = 0       Time       ± from       Trigger       ± = 43.137 ns	
Seconds/div 20.000 ns 📕 Delay 10 s	E
G1         tr         G2           *MEM_WE all         1         0         1         0           data[0-3] all         1         0         0         0	

#### See Also

To see how setup and hold violations affect software execution, (see "To capture SW execution on a setup or hold violation" on page 289).

### To trigger if a pattern doesn't follow an edge



Possible uses:

• To measure interrupt response time.

• To trigger when expected data does not appear on the data bus from a remote device when requested.

\_

Probing the Target System 1. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing<1>
Source Viewer<1>
Run 🖻
Sampling Format Trigger Symbol Analyzer Name: Analyzer (E) I On ZGHz Common Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer
$\sim$ State Mode – Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
333 MHz Full Channel 2M Sample 🛃 Trigger Position Center
Acquisition Depth 2M
Sample Period 3.0ns

Pod Assignment.		
	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: Timing -	Type: Off —	Unassigned Pods
E1: ********************		D3: L .
E2: <sup>‡‡‡‡‡‡‡‡‡‡‡</sup> _‡ K ‡		D4:M .
E3:+++++++++++++++++++++++++++++++++		
E4: M _		
D1: _++++++++++++++++ J +		
D2: ************		

3. Format one label for the signals on which you will look for a pattern and another label for the signal on which you will look for the edge.

Sampling Format Trigger Symbol				
Pod Assignment	D1	Pod E4	Pod E3	Pod E2
HOST SHINDHOT ***	Έ	TTL	TTL	TTL
		ī5 <sup></sup> 87 <sup></sup> 0	15 87 0	<b>*********************</b>
ADDR +				******
*IRQ3 +			*	

**Capturing the Data** 1. Use the "Find pattern occurring too late after edge" trigger function.

Sampling Format

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Status	Save/Recall
General Timing	Trigger function libraries.
Find pattern occurring too soon after edge Find pattern occurring too late after edge Find glitch Wait t seconds Wait for arm in	edge duration
Replace Nort before	Insert after Delete

2. In the trigger definition, enter the time period and specify the edge and the pattern.

Trigger definition
1 FIND PATTERN OCCURRING TOO LATE AFTER EDGE
Find a time period of 20.004 us 🛓
<u>*IRQ3</u> Edge ↓ in which
ADDR = FFF02EA3 Hex does not occur
then Trigger and fill memory

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Waveform display to show that the pattern didn't occur after the edge. Use the Search dialog to find the edge and the pattern.

Expand	ler Card (Master: E)
Select	
16717A	Setup
333MHz St	Waveform<1>∳
	Timing Zoom<1>
	Listing<1>
	Source Viewer<1>
	Run 🕨

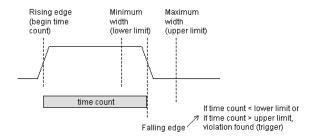
Search G	oto   Markers	Comments	Analysis	Mixed Signa	1	
Label ADDR	📕 Value [f	ff02ea3 🛓	when Entering	Next	Prev	
Advanced s	earching	Set G1 Se	et G2			
Seconds/di	v = [10.000 u	is 🔺	Delay 🚺 s	<u>×</u>		
		G1	tr		G2	
ADDR all		🖬 🗖				
*IRQ3 all	1	1		1		

If the analyzer never triggers, the pattern always occurs after the edge within the time specified.

**See Also** "Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

### To verify pulse widths



Possible uses:

- To test minimum and maximum pulse limits.
- To verify that all pulses controlling a mechanical device fall within specifications.

Probing the Target System 1. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom(1)
Listing(1)
Source Viewer(1)
Run 🖻
Sampling Format Trigger Symbol Analyzer Name: Analyzer(E) 7 On 2GHz 2001
<ul> <li>Timing Mode - Asynchronous sampling clocked internally by analyzer</li> <li>State Mode - Synchronous sampling clocked by the Device Under Test</li> </ul>
Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

Sampling Format Pod Assignment		
-	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: Timing 🖃	Type: Off - Unassigned Pods	\$
E1: **************	E3:++-+++++_+_ L + D3:	_ L _
E2: ++++++++++++++++++++++++++++++++++++	E4: M _ D4:	_м_
<b>D1:‡‡‡‡‡‡‡‡‡‡‡‡‡</b>		
D2: ++++++++++++++++++++++++++++++++++++		

3. Format a label for the signal whose pulses you will be looking at.

Sampling Format Trigger Symbol							
Pod Assignment	Т	On Clocks D. DE. F		Pod D2	Pod D1	Pod E2	
HSSIgnment	•			TTL	TTL	TTL	
		‡‡ KJ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	<b>************</b> _* 15 87 0	
addr[0-31]	+	••				****	
*MEM_OE	+			*			
*CS1	+	•••			*		
*MEM_WE	+	••		······*			
data[0-7]	+	••		******			

**Capturing the Data** 1. Use the "Find width violation on pattern/pulse" trigger function.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Statu	s Save/Recall
General Timing	Trigger function libraries
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge	Pulse too narrow Pulse too wide
Replace 📐 🛛 Insert before	Insert after Delete

2. In the trigger definition, specify the pattern and enter the minimum and maximum widths.

Trigger definition
1 FIND WIDTH VIOLATION ON PATTERN/PULSE
Find maximum or minimum width violation
on <u>*MEM_WE</u> = 0 Hex
min width 12 ns 🔺 max width 72 ns 🛓
then Trigger and fill memory

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Waveform display to show that the trailing edge of the pulse didn't occur within the defined interval.

Expand	ler Card (Master: E)	
Select	->	
16717A	Setup	
333MHz St	Waveform<1> 🖌	
	Timing Zoom<1>	
	Listing<1>	
	Source Viewer(1)	
	Run	>

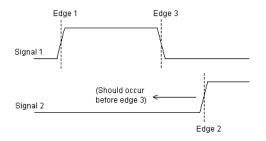
Search   Goto	Marker	s Comme	nts   Analı	lysis   Mixed Signal	
G1: *CS1	<u>+</u> =	0 Time	上 from [	Trigger 🛓 = -72.000 ns	
G2: *CS1	<u>+</u> =	1 Time	from [	Trigger 🛓 = 5,320 us	
Seconds/div =	20,000	) ns 📕	Delay	j0 s	E
	G	1	1	tr	G2
addr[0-31] all			00	000041CA	
*MEM_OE all				1	
*CS1 all		0		1	
*MEM_WE all	1		0	1	
data[0-7] all	CA			01	
•					

If the analyzer never triggers, the falling edge occurs within the defined interval.

**See Also** "Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

### To trigger on a violation of an edge sequence



Possible uses:

- To detect a handshake violation.
- To trigger on incorrect control signal generation from a Programmable Logic Device (PLD).

#### Probing the Target System

1. Configure a timing analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run M
Sampling Format Trigger Symbol Analyzer Name: Analyzer (E) I On Color Timing Zoom Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test Timing Mode Controls 333 MHz Full Channel 2M Sample I Trigger Position Center
Acquisition Depth 2M
Sample Period 3.0ns

Sampling Format Pod Assignment.		
	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: Timing 🗆	Type: Off =	Unassigned Pods
E1: ++++++++++++++++++++++++++++++++++++	E3:+++++++++++++++++++++++++++++++++	D3:L _
E2: ++++++++++++++++++++++++++++++++++++	E4:M _	D4:M _
D1:**********		
<b>D2: #########</b> ##########################		

3. Format labels for the signals whose edges you will be looking at.

Sampling Format Trigger Symbol							
Pod Assignment			Pod D2	Pod D1	Pod E2		
HSSIgnment	·			TTL	TTL	TTL	
		<b>‡‡</b> КЈ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	<b>************</b> _* 15 87 0	
addr[0-31]	+	• •	•••			****	
*MEM_OE	+			*			
*CS1	+	••			*		
*MEM_WE	+	••		·····			
data[0-7]	+			******			

Capturing the Data1. Build a two level trigger setup using the "Find pattern" and "Advanced 2-<br/>way branch" trigger functions. The pattern will identify when the edge<br/>sequence is about to occur and the 2-way branch will check for the<br/>sequence of edges.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Status Save/Rec	all
General Timing	Trigger function libraries.
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge	Occurrence 1
Replace N Insert before Insert	after Delete
Sampling Format Trigger Symbol ] Trigger Functions Settings Overview Status Save/Rec	all
General Timing	Trigger function libraries.
Advanced - If/then Advanced - 2-way branch Advanced - 3-way branch Advanced - 4-way branch Advanced - Find pattern1 AND pattern2	

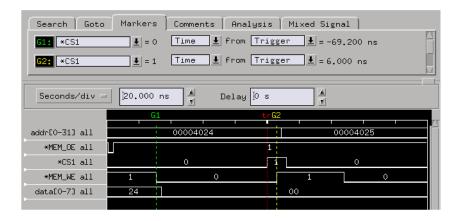
2. In the trigger definition, specify the pattern and the edges. If edges occur in the wrong sequence, trigger the logic analyzer; otherwise, go back and look for the next occurrence of the edges.

1 FIND PATTERN Find *CS1 = 0 Hex And *MEM_WE = 0 Hex					
Find *CS1 = 0 Hex And *MEM_WE = 0 Hex	Trigger definition				
*MEM_WE = 0 Hex	1 FIND PATTERN				
	Find *CS1 = 0 Hex And				
them Coto Next	*MEM_WE = 0 Hex				
	then Goto Next				
2 If *CS1 Edge ↑ occurs 1 time then Trigger and fill memory Else if *MEM_WE Edge ↑ then Goto 1	occurs 1 time then Trigger and fill memory Else if *MEM_WE Edge ↑				

3. Select the Run button to start the measurement.

**Displaying the Data** 1. When the analyzer triggers, use the Waveform display to show the edge sequence violation.



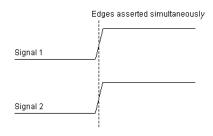


If the analyzer never triggers, the edges occur in the proper sequence.

#### See Also

"If the trigger doesn't occur as expected" on page 309

### To trigger when two edges are asserted simultaneously



Possible uses:

Probing the Target

System

- To detect bus contention.
- To view system activity when two entities are trying to seize a digital communications channel at once.

1. Configure a timing analysis machine.

Expander Card (Master: E) Select -> E1 16717A Setup 3330Hz Sta Waveform<1> Timing Zoom<1> Listing<1>
Source Viewer<1>
Run N
Sampling Format Trigger Symbol Analyzer Name: Analyzer(E) I On ZGHz Coom Timing Zoom
Timing Mode – Asynchronous sampling clocked internally by analyzer State Mode – Synchronous sampling clocked by the Device Under Test
-Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

Sampling Format Pod Assignment.			
-	Pod Assig	nment	
Analyzer 1	Analyzer	2	
Name: Analyzer <e></e>	Name: Analyzer	.E2>	
Type: Timing =	Type: Off —		Unassigned Pods
E1: ***************	E3:‡‡-‡‡‡	‡_‡ L ‡ 🛛 🗖	3 <mark>:</mark> L _
E2: <b>##########</b> ##K #	E4:	M _ 🛛 🗖 🗖	4 <mark>:</mark> M _
D1:\$\$\$\$\$\$\$\$\$			
<b>D2: #########</b> ##########################			

3. Format labels for the signals whose edges you will be looking at.

Sampling Format Trigger Symbol								
Pod	Clocks D. DF. F	Pod D2	Pod D1	Pod E2				
Assignment		TTL	TTL	TTL				
	<b>‡‡ ‡‡</b> КЈ КЈ	<b>‡‡‡‡‡‡‡‡‡</b> 15 87 0	$\begin{array}{c} \ddagger \ddagger$	<b>‡‡‡‡‡‡‡‡‡‡ ‡</b> 15 87 0 1				
addr[0-31] +		•••••	•••••					
*MEM_OE +		*						
*CS1 +			*					
*MEM_WE +		*						
data[0-7] +		*****						
*TS +			*					
*TA +	*.			·····				
AT2 +			*					

**Capturing the Data** 1. Use the "Find edge" trigger function.

Sampling Format Trigger Symbol							
Trigger Functions Settings Overview Status Sa	ave/Recall]						
General Timing	Trigger function libraries						
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge	edge						
Replace N Insert before	Insert after Delete						

2. In the trigger definition, select the button after Find, and choose the "Insert LABEL after" item. Specify the two edges.

Trigger definition	
1 FIND EDGE	
Find <u>*CS1</u> Edge ↓	And
×TS Edge ↓	
then Trigger and fill memory	

3. Select the Run button to start the measurement.

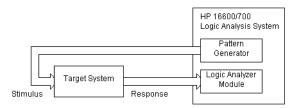
## **Displaying the Data** 1. When the analyzer triggers, use the Waveform display to show the two edges that occur at the same time.

Expander Card (Master: E)						
Select						
16717A	Setup					
333MHz St.	Waveform<1> 🖌					
	Timing Zoom<1>					
	Listing<1>					
	Source Viewer<1>					
	Run 🕨					

Search   Goto	Markers Comments Analysis Mixed Signal
G1: *CS1	➡ = 1 Time ➡ from Trigger ➡ = -4.020 us
G2: *CS1	
Seconds/div _	100.000 ns 🖌 Delay 10 s
addr[0-31] all	◆03959 ◆0395A FFF0395B 00004349 ◆0395C ◆0395D
*MEM_OE all	
*CS1 all	1 0 1
*MEM_WE all	1
data[0-7] all	89 4C 43 49 01 7D 4A
*TS all	
*TA all	
AT2 all	0 1 0

If the analyzer never triggers, the two edges don't occur within the same sample period.

### To generate pattern stimulus on devices



**Requirements:** 

• This measurement requires a pattern generator module (Agilent Technologies 16522A).

Possible uses:

- To simulate digital circuitry that isn't available.
- To generate signals for functionally testing prototype hardware.

#### Connecting Pattern Generator Outputs to the Target System

1. Connect the pattern generator lines to your target system using the appropriate TTL, CMOS, or ECL data or clock pods (see the pattern generator documentation (see the *Agilent Technologies 16522A 200M Vectors/s Pattern Generator* help volume) for more information).

#### Configuring the Pattern Generator & Labeling Outputs

1. In the Format tab, select the output mode and the clock source, and label the output signals.

Select	->	4	
16522A	Setup		<u> </u>
200Mvector	Run 🕨	ator	

Format Sequence Macro								
Output Mode Fu	Output Mode Full Channel 100Mbit/s = Clock Source Internal							
Clock Out Dela	Clock Out Delay Clock Period 100.0us							
	Pod B5	Pod B4	Pod B3	Pod B2	Pod B			
	70	70	70	70	7			
Data	+				*****			
Dreg/*Ireg	+			.*				
R∕*W	+			*				
Enable^	+			*				
*PGenEnabl	+			*				

#### Building the Test Vectors

1. In the Sequence tab, insert vectors. Select data values, and enter the new values. A double-quote character means the same value as above.

Format Sequence Macro Pattern Fills							
xed Count	Rotate Toggle Random						
INIT START							
INIT END MAIN STAR MAIN END MAIN END	Vector A Loop 0 User Hacro Hait External Event Hait II-B: Event If External Event If II-B: Event Signal II-B:						
	Eneak						
	rn Fills xed Count Instruction Hex Binar INIT START INIT START MAIN STAR Goto Line						

During a repetitive run, the vectors in the INIT section are only executed once.

You can use macros to insert vectors that need to be repeated.

INIT END     Insert After      Vector       MAIN STAR     Goto Line     Loop       WAIN END     User Mac     Wait Ext       MAIN END     User Mac     User Mac       Signal If     Signal If       Break	ernal Event 5. Macro 3 Event 6. Macro 4 nal Event 7. Macro 6 al Event 8. Macro 7 event 9. Macro 8
Format Sequence Macro Macro Name Macro O Pattern Fills Fixed	Parameters Copy To Recall
Line Instruction Hex Hex	able Dreg/*Ireg R/*W Enable^ Hex Hex Hex
0   MACRO START  1   MACRO EN  Insert After Set Farameter Clear Parameter Goto Line	Vector Loop User Hacro Hait External Event Ereak

You can pass parameters to macros.

	F	Parameter Mo	dification	
	1. Instruct	ion:	💠 Off 🔶	On 🛛
Macro	2. Paramete	er 1	🗢 Off 💠	0n
Parameters	3. Paramete	er 2	🗢 Off 💠	On 🚽
		Close	<b>N</b>	
0 MACRD START			Set Paramet	ter
1		Instruction		
2 MACRO END Insert Af	fter 🕨			
	meter 🖌			
Clear Par	rameter			
Goto Line	-			
GOCO LING	C+++	ок 📐	Apply	Cancel

Running the Pattern<br/>Generator1.When you've finished building the test vectors, select the Run button to<br/>cause the pattern generator to output the vectors.

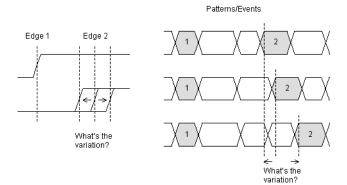
ſ	Format	Sequence	Macr	• ]				
L C	Patter	n Fills						
	Fi>	(ed C	ount	. Ro	tate	Togg	le	Random
			Data	Dreg/*Ir	eg R/*W	Enable	*PGenE	Inable
	Line	Instruction	Hex	Hex	Hex	Hex	Hex	
ſ	0	INIT START						
	1		ΰo	0	0	0	1	
	2						0	
	3	MACRO	Write	Instructi	on(00000	038)		
	4	MACRO	Write:	Instructi	on(00000	DOE)		
	5	MACRO	Write	Instructi	on(00000	006)		
	6	MACRO	Write:	Instructi	on(00000	001)		
	7	INIT END						
	8	MAIN START						
	9		00				0	
	10							
	11	MACRO		Instructi		(030		
	12	MACRO		Data(0000				
	13	MACRO	Write	Data(0000	0065)			
	14	MACRO		Data(0000				
	15	MACRO		Data(0000				
	16	MACRO		Data(0000				
	17							
	18	MAIN END						

<u> </u>	ormat	Sequence	Macro					
Ma	Macro Name WriteInstruction Parameters Copy To Recall							
F		n Fills	ount	Ro	tate	Toggl	e	Random
L	ine	Instruction		Dreg∕*Ir Hex	eg R/*W Hex	Enable^ Hex	*PGenEna Hex	able
0 1 2 3		MACRO START START LOOP	[1]	O " REPEAT 1	0 " 1 TIMES	1	0 "	
4 5 6		END LOOP MACRO END	[1]			1		

Macro M	Name WriteDat	a	Para	ameters	s Cor	⊳y To	Recall
Pattern Fills Fixed Count Rotate Toggle Random							
Line	Instruction		Dreg/*Ireg Hex	R∕*W Hex	Enable^ Hex	*PGenEna Hex	ble
0	MACRO START						
1		*Dat	1	0	1	0	
1 2 3					0		
3					1		

See Also"To simulate particular interrupt sequences" on page 191"To generate patterns when a source line executes" on page 262

### To analyze jitter or time dispersion (with SPA)



#### **Requirements:**

• This measurement requires the system performance analyzer (SPA) tool set.

#### Possible uses:

- To measure the jitter between two edges.
- To measure the variation between two bus states.
- To measure setup and hold times.

#### Probing the Target System

- **jet** 1. Probe the signals of interest.
  - 2. Configure a timing or state analyzer, depending on whether you want to look at signal edges, patterns, or events.

Expander Card (Master: E)
Select $\rightarrow$ $\Box$ 1
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run 🖻
Sampling Format Trigger Symbol Analyzer Name: Analyzer(E) 7 On 2GHz Parager Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M
Sample Period 3.0ns

3. Assign pods if necessary.

Sampling Format				
		Pod Assignment		
Analyzer 1		Analyzer 2		
Name: Analyzer <e></e>	Name:	Analyzer <e2></e2>		
Type: Timing -	Type:	Off 📼	Unassigned	d Pods
E1: ++++++++++++++++++++++++++++++++++++	E3: -	‡‡*‡‡‡_‡ L ‡	D3:	L .
E2: ++++++++++++++++++++++++++++++++++++	E4: -	M _	D4:	M _
D1:\$\$\$\$\$\$\$\$				
<b>D2: #########</b> K <b>#</b>				

4. Label the logic analyzer channels. (If you're using an analysis probe, you can configure the analyzer and set up labels by loading the included configuration files.)

Sampling Format Trigger Symbol						
Pod Assignment		Clocks DDFF		Pod D2	Pod D1	Pod E2
				TTL	TTL	TTL
		‡‡ KJ	‡‡ KJ	$\begin{array}{c} \ddagger \ddagger$	$\begin{array}{c} \ddagger \ddagger$	<b>‡‡‡‡‡‡‡‡‡‡‡</b> 0
addr[0-31]	+	••				*****
*MEM_OE	+	••		*		
*CS1	+	•••			·····*····	
*MEM_WE	+	•••		*		••••••
data[0-7]	+	••		*****		
*TS	+	••	•••		*	
*TA	+	••	*.			
AT2	+	••			*	

#### Capturing the Data

1. Set up a trigger specification to capture the signal edges, patterns, or events you're interested in.

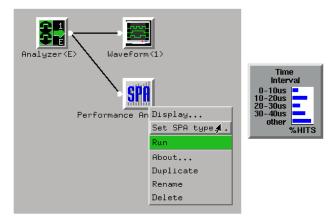
Sampling Format Trigger Symbol		
Trigger Functions Settings Overview Status	Save/Recall	
General Timing	Trigger	function libraries.
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge		currence 1
Replace Insert before	Insert after	Delete
Trigger definition		
I FIND PATTERN		
Find addr[0-31] = XXXXXXXX Hex		
then Trigger and fill memory		

- 2. Select the Run button to start the measurement.
- 3. Display the captured waveforms.



Search Goto	Markers Comments Analysis Mixed Signal
Goto Time 🛓	ios 🛓 Goto
Trigger Be G1 & G2 Fente	ginning End G1 G2
Seconds/div =	1.000 us 📕 Delay 🔯 s
	1 tr G2
addr[0-31] all	
*MEM_OE all	
*CS1 all	
*MEM_WE all	1 0 1
data[0-7] all	
*TS all	אריינטער בי אנעט אינעט אינער אינער אינער אינער אינער אינעט אינעט אינעט אינעט אינער אינער אינער אינער אינער אינע
*TA all	ממר דרמב ב הממור מממר ממור מרמר מממר מממר ממחר המחר הממור הממור הממר ממור ה
AT2 all	

**Displaying the Data** 1. Use the system performance analyzer's Time Interval display to view the captured data.



2. Define the start and end of the event whose time variations you wish to measure.

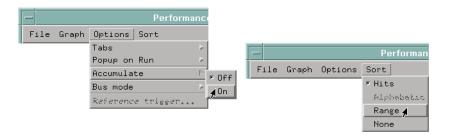
Performance Analysis<1>					
File Window Graph Options Sort					
Define ranges Qualify data Include other					
Time Interval					
Qualify Data - Performance Analysis<1>					
Define the start and end of event. The time duration between the start of event and end of event is graphed.					
Start of event					
The start of the duration occurs when this event is: Entered					
Event:					
*CS1 Binary - Pattern - 0					
End of event					
The end of the duration occurs when this event is: Entered =					
Event:					
Event:					
AT2 Binary - Pattern - 0					
U					

3. Define *buckets* for expected time ranges.

Ti Ti	– Auto Range – Performance Ana
Time Ranges Hits(%)	-Range limits
100 ms Undo 20	Start time: 268,000 ns 🔺
300 ms     X)       400 ms     Delete	End time: 1632,001 ns
500 ms Delete thru end X) 600 ms View event times X)	◆ Linear
800 ms 900 ms	Number of buckets: 10
	↓ Logarithmic
	Log base 10 🔺
	OK Apply Cancel

lys

4. Set the appropriate data gathering and display options.

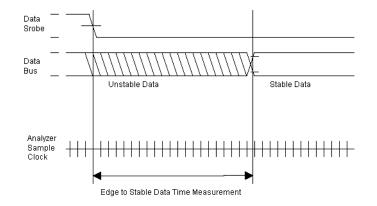


Use Accumulate Mode to analyze the behavior of your system over a long period of time (and, perhaps, run the measurement repetitively).

5. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.

	Time	Interval		
Time Ranges	Hits(%)			
268 ns 304,4 ns	1,855(41,1%)			
304.4 ns 340.8 ns	0(0,0%)			
340.8 ns 377.2 ns	0(0,0%)			
377.2 ns 413.6 ns	140(3,1%)			
413.6 ns 450 ns	0(0,0%)			
450 ns 486,4 ns	0(0,0%)			
486.4 ns 522.8 ns	0(0,0%)			
522.8 ns 559.2 ns	0(0,0%)			
559.2 ns 595.6 ns	0(0,0%)			
595.6 ns 632.001 ns	2,521(55.8%)			
	0	x 20x	40% 60% 80%	100%
Display Information			nformation	
Qualified Events:	4.517		1,855(41.07%)	
Total Acquisitions:	4		268 ns - 304,4 ns	
Total Display %:	99.98%		268 ns/276 ns	
Total Buckets:	10	Mean:	271.163 ns	

Statistics such as the maximum time, minimum time, standard deviation, and mean help you document system behavior.



### To analyze bus stability (with SPA)

The stability of a bus is defined by two or more consecutive acquisitions of the same data value on the bus.

For example, if you analyze a microprocessor's access to a RAM, you want to be sure that the data is stable when it is strobed.

In this context, the system performance analyzer helps you characterize areas of stability or instability for this bus.

Requirements:

• This measurement requires the system performance analyzer (SPA) tool set.

Possible uses:

- To measure the correlation between a signal (such as a strobe or an edge) and the presence of valid, stable information on a bus (or a label with one or more channels).
- To search on the entry or exit of a stable or unstable bus condition.
- To focus on bus transactions.
- To search for stability within a defined time range or outside a defined time range.

Probing the Target 1. Probe the bus and strobe signals of interest. System

2. Configure a timing analyzer.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run 🕨
Sampling     Format     Trigger     Symbol       Analyzer Name:     Analyzer (E)     Image: Contemport     Timing Zoom
Timing Mode - Asynchronous sampling clocked internally by analyzer State Mode - Synchronous sampling clocked by the Device Under Test
-Timing Mode Controls-
333 MHz Full Channel 2M Sample 🛓 Trigger Position Center 🛓
Acquisition Depth 2M 🛓
Sample Period 3.0ns

3. Assign pods if necessary.

Sampling Format

Assignment.						
-	Pod Assignment					
Analyzer 1	Analyzer 2					
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>					
Type: Timing =	Type: Off 📼	Unassigned Pods				
E1: *************	E3:+++++++++++++++++++++++++++++++++	D3: L .				
E2: ++++++++++++++++++++++++++++++++++++	E4:M _	D4:M .				
D1:\$\$\$\$\$\$\$\$						
<b>D2: \$\$\$\$\$\$</b>						

4. Label the logic analyzer channels.

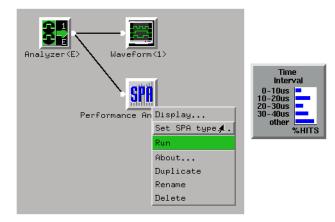
Sampling Format Trigger Symbol							
Pod	Pod Assignment		Pod D2	Pod D1	Pod E2		
HSSIgnment.			TTL	TTL	TTL		
<b>‡‡ ‡‡</b> KJ KJ				<b>‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡</b> _ <b>‡‡‡</b> 15 87 0	<b>‡‡‡‡‡‡‡‡‡‡‡‡15</b> 87 0 1	
addr[0-31]	+	••	•••			<u>*************************************</u>	
*MEM_OE	+	••		*			
*CS1	+				*		
*MEM_WE	+	•••		····· *·····			
data[0-7]	+			*****			
*TS	+				*		
*TA	+	••	*.				
AT2	+	••			*		

#### **Capturing the Data**

1. Set up a trigger specification to capture the strobe signal edge and the bus signals.

Sampling Format Trigger Symbol								
Trigger Functions Settings Overview Status Save/Recall								
General Timing	Trigger function libraries							
Find pattern Find edge Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge	Occurrence 1							
Replace Insert before	Insert after Delete							
Trigger definition								
1 FIND PATTERN Find addr[0-31] = XXXXXXX Hex then Trigger and fill memory								

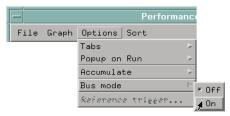
**Displaying the Data** 1. Use the system performance analyzer's Time Interval display to view the captured data.



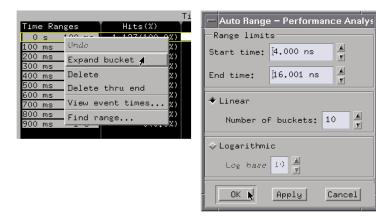
2. Define the start of the event as the data strobe signal going active, and define the end of event as the bus being stable (that is, a "don't care" pattern *Not Present*).

Performance Analysis<1>					
File Window Graph Options Sort					
Define ranges Qualify data Include other					
Time Interval					
Qualify Data - Performance Analysis<1>					
Define the start and end of event. The time duration between the start of event and end of event is graphed.					
Start of event					
The start of the duration occurs when this event is: Entered =					
Event:					
TS Binary - Pattern - 1					
End of event					
The end of the duration occurs when this event is: Not Present $\Rightarrow$					
Event:					
data[0-7] Hex - Pattern - XX					

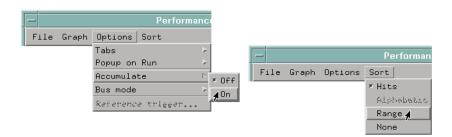
3. Turn ON the system performance analyzer's bus mode.



4. Define *buckets* for expected time ranges.

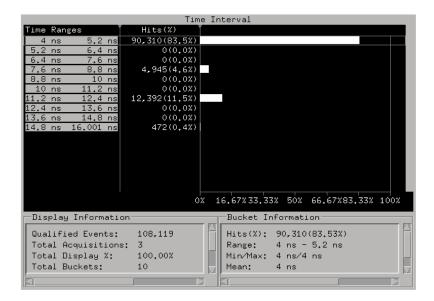


5. Set the appropriate data gathering and display options.



Use Accumulate Mode to analyze the behavior of your system over a long period of time (and, perhaps, run the measurement repetitively).

6. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.



You can use recorded event times to view an event in the Waveform display.

14.8 ns 14.8 ns 14.8 ns 16.001 ns Undo Expand bucket Delete Delete thru end View event times Find range	C       Start / End         142.352 us / 142.368 us       186.728 us / Send start time to marker         194.112 us / Send end time to marker       61         Start / End       186.728 us / 142.368 us         186.728 us / Send start time to marker       186.728 us / Send start time to marker         186.728 us / Send start time to marker       61         194.112 us / Send end time to marker       61
Expander Card (Master: E	
HP 16555D	
2M Sample 110 MHz State/500 MHz T	THITHA
Setup Waveform<1>	
Listing<1>	
Source Viewer	
	Analysis<1>
	un P
Search Goto Markers	Comments   Analysis   Mixed Signal
Goto Time 🛓 🛛 s	
Trigger Beginning	End G1 G2
G1 & G2 Centered	
Seconds/div 10.000 ns	Delay 142.352 us
r	G1 G2
addr[0-31] all ◆386F	FFF03870
*MEM_OE all 1	
*MEM_WE all	1
*CS1 all	1
data[0-7] all	50 3D
*TS all 1	0 1
*TA all	1 0
AT2 all	
· · · · · · · · · · · · · · · · · · ·	

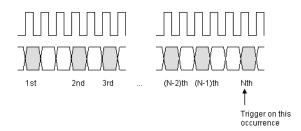
#### See Also

Bus Mode Search Criteria (see the *System Performance Analyzer* help volume) in the system performance analyzer help volume.

### Looking at State Events

- "To trigger on the Nth occurrence of an event" on page 96
- "To store N samples of an event" on page 100
- "To trigger on a sequence of events" on page 105
- "To trigger when a program loop exits" on page 111
- "To find events that are too close or too far" on page 116
- "To count occurrences of an event between two events" on page 120
- "To trigger on a function call sequence" on page 125
- "To analyze bus occupation & bandwidth (with SPA)" on page 131

#### To trigger on the Nth occurrence of an event



Possible uses:

- To find the 50th occurrence of a digital signal processing (DSP) subroutine.
- To trigger on the 3rd write to a specific memory address.

Probing the Target System 1. Configure a state analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E)
$\diamond$ Timing Mode – Asynchronous sampling clocked internally by analyzer
State Mode - Synchronous sampling clocked by the Device Under Test

2. Select the state analyzer's clock input.

State Mode Controls								
167 MHz /	167 MHz / 2M State 🛓 Trigger Position Center 🛓							
Acquisitio	Acquisition depth 2M							
Clock Set	up—							
Mode: Ma	ster	onl	<u>ا</u> ا		Advanced Clocking			
Pod	E4	E3	E2	E1				
Clock	М	L	К	J				
Activity	_	\$	\$	\$				
Master	Off	Off	0	F	=> [(J†) •(K=0)]			
					·			

3. Format labels for the signals on which you will look for the event.

Sampling Format Trigger Symbol								
Pod Assignment	Clocks		Pod D2	Pod D1	Pod E2			
			TTL	TTL	TTL			
Setup/Hold KJ KJ		<b>**********</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	<b>‡‡‡‡‡‡‡‡‡‡‡</b> 0	‡: 15			
ADDR +		•••	•••••	•••••	****	**		
DATA +		••	*****	****	•••••			
STAT +	**	**	*	*****	•••••			
data[0-7] +			******	•••••				

**Capturing the Data** 1. In the Trigger tab, use the "Find pattern n times" trigger function.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Default Storing Status Sa	ve/Recall
General State Trigger f	unction libraries.
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop	Occurrence n pattern
Replace Insert before Insert after	Delete

2. In the trigger definition, enter the number of occurrences and specify the pattern.

Trigger definition					
1 FIND PATTERN N TIMES					
Find 10					
ADDR = FFF034D8 Hex					
then Trigger and fill memory					

Use the default storage qualifier that is initially on and stores all states.

3. Select the Run button to start the measurement.

**Displaying the Data** 1. When the analyzer triggers, use the Listing display to show that the event occurred the number of times you specified.

Expander Card (Master: E)					
Select					
16717A	Setup				
333MHz St.	Waveform<1>				
	Timing Zoom<1>				
	Listing<1> 🖌				
Source Viewer<1>					
Run 🕨					

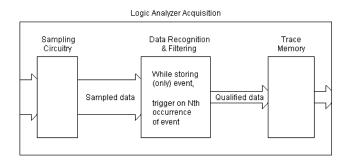
s	Search   Goto   Markers   Comments   Analysis   Mixed Signal								
La	Label ADDR 🛓 Value 🗓 🞍 when Present 🛓 Next Prev								
Ĥ	Advanced searching Set G1 Set G2								
			ĺ	_					
	State Number	ADDR	Time	DATA	STAT	data[0-7]			
	Decimal	Hex	Relative	Hex	Hex	Hex			
G1_	-8	000041B0	232.000 ns	004123D7	0123D7	_00			
	-7	000041B1	120.000 ns	0F4123D7	0923D7	0F			
	-6	000041B2	116.000 ns	2C4123D7	1123D7	20			
	-5	000041B3	116.000 ns	ED4123D7	0923D7	ED			
	-4	FFF03184	272.000 ns	484103E7	0103E7	48			
	-3	FFF03185	116.000 ns	004103E7	0903E7	00			
	-2	FFF03186	116.000 ns	034103E7	1103E7	03			
	-1	FFF03187	120.000 ns	554103E7	0903E7	55			
<u>tr</u>	0	FFF034D8	192.000 ns	7C4103E7	0103E7	70			
	1	FFF034D9	120.000 ns	084103E7	0903E7	08			
	2	FFF034DA	116.000 ns	024103E7	1103E7	02			
	3	FFF034DB	116.000 ns	A64103E7	0903E7	A6			
	4	FFF034DC	272.000 ns	7C4103E7	0103E7	7C			
	5	FFF034DD	120.000 ns	2B4103E7	0903E7	2B			
	6	FFF034DE	116.000 ns	0B4103E7	1103E7	OB			
	7	FFF034DF	116.000 ns	784103E7	0903E7	78			
G2.	8	FFF034E0	272,000 ns	944103E7	0103E7	_94			

If the analyzer never triggers, the event does not occur the number of times specified. You can stop the measurement and look at the Listing display to see how many times the event did occur.

"Use trigger functions for easy measurement set up" on page 305

"If the trigger doesn't occur as expected" on page 309

### To store N samples of an event



You can limit the data that is stored in trace memory at each trace sequence level or by specifying whether trace level branches are stored.

Possible uses:

- To view the first 200 reads and writes to a FIFO.
- To look at 75 pushes onto the stack.

Probing the Target System 1. Configure a state analysis machine.

Expander Card (Master: E)	
16717A Setup	
333MHz Sta Waveform<1>	
Timing Zoom<1>	
Listing(1)	
Source Viewer<1>	
Run 🖻	
Sampling Format Trigger Symbol	1
Analyzer Name: Analyzer(E)	7 On 2GHz Property Timing Zoom
☆ Timing Mode - Asynchronous sampling	clocked internally by analyzer
◆ State Mode – Synchronous sampling o	clocked by the Device Under Test

2. Select the state analyzer's clock input.

—State Mode	-State Mode Controls										
167 MHz / 2M State 🛃 Trigger Position Center											
Acquisition depth 2M											
Clock Set	up—										
Mode: Ma	ster	only	y l		_ Advanced Clocking						
Pod	E4	E3	E2	E1							
Clock	М	L	К	J							
Activity	-	‡	‡	\$							
Master	Off	Off	0	F	=> E(J†) •(K=0)]						
L											

3. Assign pods if necessary.

	Sampli Po Assign	- Data			
	-			Pod Assignment	
L					
I		-Analyzer 1		-Analyzer 2	
I	Name:	Analyzer <e></e>	Name:	Analyzer <e2></e2>	
I	Type:	State =	Type:	Off =	Unassigned Pods
I	E1: ‡	<b>******</b> ** <b>*</b> ********* J <b>*</b>	E3: -	\$\$**\$\$\$\$_\$ L \$	D3: L _
I	E2: ‡	\$	E4: _	M _	D4:M _
I	D1: _	************* J *			
	D2: ‡	<b>*****</b> ****			
I	D2: ‡	\$			

4. Format labels for the signals on which you will look for the event.

Sampling Fo	rmat	Tra	igger   Symbol		
Pod Assignment			Pod D2	Pod D1	Pod E2
			TTL	TTL	TTL
Setup/Hold	‡‡ KJ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡‡</b>	<b>***********</b> _* 15 87 0
ADDR +		•••		•••••	
DATA +			*****	*****	•••••
STAT +	**	**	*	****	•••••
data[0-7] +		•••	*****		•••••

**Capturing the Data** 1. In the Trigger tab, use the "Find pattern n times" trigger function.

Sampling Format	Trigger Symbol		
Trigger Functions Set	tings Overview Defa	ult Storing Status Sa	we/Recall]
General State		Trigger f	unction libraries
Find pattern n times Store range until pat Store pattern2 until p Store nothing until p. Run until user stop	oattern1 occurs	Occurrence 1 pattern	Occurrence n
Replace 📐	Insert before	Insert after	Delete

2. In the trigger definition, specify the event and the number of samples you want to store. Also, insert an action to turn off default storing after the number of events occur.

3. Set up the default storing to store only the event you're interested in.

Sampling Format Trigger Symbol	1
Trigger Functions Settings Overview	efault Storing Status Save/Recall
Specify what to store by default.	Store if =
Using store actions in a sequence level overrides the "Default Storing".	ADDR = FFF034D8 Hex
Store by default Custom 😑	
At start of acquisition, On = "Default Storing" is	
	Group events

- 4. Select the Run button to start the measurement.
- 5. Select the Stop button after the trigger occurs. (Because default

storing is turned off after the trigger, trace memory will not fill up.)

**Displaying the Data** 1. When the analyzer triggers, use the Listing display to show that N event occurrences are stored.



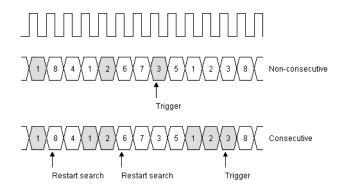
ſs	earch   Goto	Markers	Comments	Analysis	Mixed	Signal ]	
G	1: ADDR	₹ = FFF	034D8 Time	± from	Trigge	r 🛓 = −158.077 ms	A
G	2: ADDR	₹ = FFF	034D8 Time	<u>↓</u> from	Trigge	r <u>t</u> = 0 s	
	State Number	ADDR (	Time	рата (	STAT (	data[0-7]	T
	Decimal	Hex	Relative	Hex	Hex	Hex	
	Decimai		Keracive				
G1_	-9	FFF034D8		7C4103E7	0103E7	_7C	
	-8	FFF034D8	16.726 ms	7C4103E7	0103E7	70	
	-7	FFF034D8	16.968 ms	7C4103E7	0103E7	7C	
	-6	FFF034D8	17.756 ms	7C4103E7	0103E7	7C	
	-5	FFF034D8	20.387 ms	7C4103E7	0103E7	7C	
	-4	FFF034D8	17.577 ms	7C4103E7	0103E7	7C	
	-3	FFF034D8	17.163 ms	7C4103E7	0103E7	7C	
	-2	FFF034D8	17.222 ms	7C4103E7	0103E7	7C	
	-1	FFF034D8	17,196 ms	7C4103E7	0103E7	7C	
62	0	FFF034D8	17.083 ms	7C4103E7	0103E7	7C	-

If the analyzer never triggers, the event does not occur N times. You can stop the measurement and look at the Listing display to see how many times the event did occur.

See Also

"If the trigger doesn't occur as expected" on page 309

#### To trigger on a sequence of events



Possible uses:

- To trigger on the occurrence of a calculation subroutine *after* two initialization subroutines have executed (non-consecutive sequence of events).
- To trigger on the access to an I/O port *after* its two I/O registers have been set (non-consecutive sequence of events).
- To trigger on the occurrence of a subroutine only when it has been called from a specific branch of the main program (consecutive sequence of events).
- To look for data writes to 4 consecutive memory locations with no reads inbetween (consecutive sequence of events).
- **arget** 1. Configure a state analysis machine.

Probing the Target System

Expander Card (Master: E)
16717A Setup
333Hz Sta Waveform(1)
Timing Zoom(1)
Listing(1)
Source Viewer(1)
Run
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E)
$\diamond$ Timing Mode – Asynchronous sampling clocked internally by analyzer
$\diamond$ State Mode – Synchronous sampling clocked by the Device Under Tes

2. Select the state analyzer's clock input.

State Mode Controls											
167 MHz / 2M State ₤ Trigger Position Center ₤											
Acquisition depth 2M											
Clock Set	up—										
Mode: Master only 🖳 🗌 Advanced Clocking											
Pod E4 E3 E2 E1											
Clock	М	L	К	J							
Activity	-	\$	\$	\$							
Master	Off	Off	0	₽	=> [(J†) • (K=0)]						

3. Assign pods if necessary.

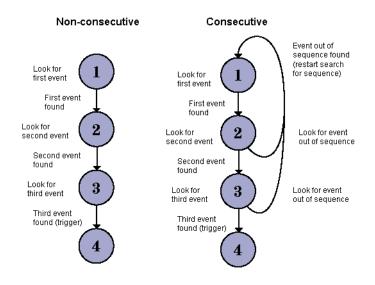
Assignment		
	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: State =	Type: Off =	Unassigned Pods
E1: ++++++++++++++++++++++++++++++++++++	E3:++++++++_+_ L +	D3:L _
E2: <sup>‡‡‡‡‡‡‡‡‡‡‡</sup> _‡ K ‡	E4: M _	D4: M _
D1: _**********		
<b>D2: ‡‡‡‡‡‡‡‡‡‡‡‡‡</b>		

4. Format labels for the signals on which you will look for the event.

Sampling Format

Sampling Fo	ormat	rigger   Sy	ymbol ]					
Pod Assignment	Clocks	- Pod	D2	Pod D1 TTL		Pod E2 TTL		
	1	TT	Έ					
Setup/Hold	<b>‡‡ ‡</b> КЈ К		*******	********** 87	. <b>‡‡‡</b>	<b>‡‡‡‡‡‡‡‡</b> 15 83	;;;;;	‡: 15
ADDR +	·			• • • • • • • • • • •	•••	******	******	**
DATA +	·	. *******	****	******	***	•••••		
STAT +	** *	*	***	*******	***	•••••		
data[0-7] +		. *******		• • • • • • • • • • •		•••••		

Capturing the Data1. To look for a sequence of non-consecutive events, set up a trigger<br/>sequence where level 1 looks for the first event; when it's found, level 2<br/>looks for the second event, and so on. The second to last sequence level<br/>looks for the last event and triggers the analyzer when it's found.



To look for a sequence of consecutive events, set up a trigger sequence where level 1 looks for the first event; when it's found, level 2 looks for the second event (which if found causes a branch to the next level) or a state that is not the second event or indicates an event out of sequence (which if found causes a branch back to the first level to restart the search).

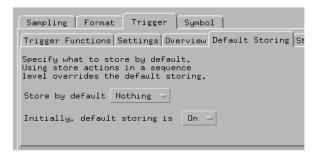
Subsequent levels are the same as level 2. When the last event is found, the analyzer triggers, and the last sequence level is used to specify what is stored after the trigger.

Sampling   Format   Trigger   Symbol	Sampling Format Trigger Symbo			
Trigger Functions Settings Overview Def	Trigger Functions Settings Overview			
General State	General State			
Wait for arm in	Wait for arm in			
Advanced - If/then	Advanced - If/then			
Advanced - 2-way branch Advanced - 3-way branch	Advanced - 2-way branch Advanced - 3-way branch			
Advanced - 4-way branch	Advanced - 4-way branch			
Replace Insert before	Replace Insert befor			
Trigger definition	Trigger definition			
1 If data[0-7] = 7C Hex	1 If data[0-7] = 7C Hex			
occurs 1 time then Store sample	occurs 1 time then Store sample			
Goto Next	Goto Next			
	Else if data[0-7] ≠ 7C Hex then Goto 1			
2 If data[0-7] = 08 Hex occurs 1 time				
then Store sample	2 If data[0-7] = 08 Hex			
Goto Next	occurs 1 time then Store sample			
3 If data[0-7] = 02 Hex	Goto Next			
occurs 1 time	Else if data[0-7] ≠ 08 Hex			
then Store sample Goto Next	then Store sample Goto 1			
4 If data[0-7] = A6 Hex	3 If data[0-7] = 02 Hex occurs 1 time			
	then Store sample			
occurs 1	Goto Next			
then Store sample	Else if data[0-7] ≠ 02 Hex then Store sample			
	Goto 1			
Trigger and fill memory	4 If data[0-7] = A6 Hex			
	occurs 1 🛓 time			
	then Store sample			
	Trigger and fill memory			
	Else if data[0-7] ≠ A6 Hex			
	then Store sample			
	Goto 1			

Note that, when looking for a consecutive sequence of events, any event that disqualifies the sequence you're looking for can be used in the "else on" branches. For example, instead of looking for a particular sequence of states, you can look for a particular sequence of function calls.

To show the differences between these trigger definitions, nothing is

stored by default.



- 2. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Listing display to show that the sequence of events occurred.

Expander Card (Master: E)					
Select	->				
16717A	Setup				
333MHz St.	Waveform<1>				
	Timing Zoom<1>				
	Listing<1> 🖌				
Source Viewer<1>					
	Run 🕨				

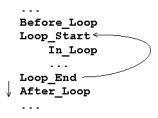
	State Number	ADDR	data[0-7]
	Decimal	Hex	Hex
	-3 -2	FFF03964	7C
		FFF06A71	08
	-1	FFF03A0B	02
tr_	0	FFF06A73	A6

	State Number	ADDR	data[0-7	
	Decimal	Hex	Hex	
	-14	FFF04B58	7C	
	-13	FFF04B59	CA	
	-12	FFF04B6C	7C	
	-11	FFF04B6D	67	
	-10	FFF03214	7C	
	-9	FFF03215	6A	
	-8	FFF03230	7C	
	-7	FFF03231	10	
	-6	FFF06A70	7C	
	-5	FFF06A71	08	
	-4	FFF06A72	03	
	-3 -2	FFF040CC	7C	
		FFF040CD	08	
	-1	FFF040CE	02	
tr_	0	FFF040CF	A6	

If the analyzer never triggers, at least one of the events in the sequence never occurs. You can stop the measurement and look at the Listing display to see which events in the sequence were captured.

"If the trigger doesn't occur as expected" on page 309

### To trigger when a program loop exits



Possible uses:

- To capture execution after a background monitor loop that runs until a control key is pressed.
- To verify that all stacks and registers are restored correctly before exiting a subroutine (that is, to look at code execution before exit).

#### See Also

**Probing the Target** 1. Configure a state analysis machine.

System

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run N
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E)
$\diamond$ Timing Mode – Asynchronous sampling clocked internally by analyzer
$\clubsuit$ State Mode – Synchronous sampling clocked by the Device Under Test

2. Select the state analyzer's clock input.

State Mode Controls					
167 MHz /	2M State	∍ ₹	Trigger Position Center 👤		
Acquisition	n depth[	2M	<u>+</u>		
Clock Set	up				
Mode: Mas	ster onl	y ₹	🔟 Advanced Clocking		
Pod	E4 E3	E2	E1		
Clock	ML	К	J		
Activity	_ ‡	\$	<b>‡</b>		
Master	Off Off	0			

3. Assign pods if necessary.

Assignment.		
	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: State =	Type: Off =	Unassigned Pods
E1: **************	E3:+++++++++++++++++++++++++++++++++	D3:L _
<b>Е2:</b> ‡‡‡‡‡‡‡‡‡‡‡ к ‡	E4: M _	D4:M _
D1: _++++++++++++++ J +		
<b>D2:</b> \$\$\$\$\$\$\$\$\$\$		

Sampling Format

4. Format a label for the address bus signals on which you will look for loop start and loop end events. (If you are using an analysis probe, the included configuration files will format an ADDR label.)

Sampling Format Trigger Symbol								
Pod Assignment	Clocks		Pod D2	Pod D1	Pod E2			
			TTL	TTL	TTL			
Setup/Hold	Setup/Hold KJ KJ		<b>**********</b> 15 87 0	<b>********************</b>	<b>***********</b> _*0			
ADDR +		•••			*****	*		
DATA +			******	******		·		
STAT +	**	**	*	******		•		
data[0-7] +		••	******			•		

Capturing the Data
1. Set up a trigger sequence where level 1 looks for the loop start event; when it's found, level 2 looks for the loop end event; when it's found level 3 looks for an event that isn't the loop start event (which if found triggers the analyzer) or the loop start event (which if found causes a branch back to level 2 where the loop end event is searched for).

Sampling Format Trigger Symbol						
Trigger Functions Settings Overview Defa						
General State						
Wait n external clock states						
Wait for arm in Advanced - If/then						
Advanced - 2-way branch Advanced - 3-way branch						
Havanced - 5-way branch						
Replace Insert before						
Trigger definition						
1 If ADDR = FFF035F4 Hex occurs 1 time						
then Store sample						
Goto Next						
2 If ADDR = FFF03633 Hex						
occurs 1 time						
then Store sample Goto Next						
3 IF ADDR = FFF03634 Hex						
occurs 1 time						
then Turn on default storing						
Trigger and fill memory						
Else if ADDR = FFF035F4 Hex						
then Store sample						
Goto 2						

To show only loop start and loop end events until the logic analyzer triggers, turn default storing off initially.

Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing
Specify what to store by default. Using store actions in a sequence level overrides the default storing.
Store by default Anything -
Initially, default storing is Off -

- 2. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Listing display to show that the program loop exited.

Expand	ler Card (Master: E)			
Select	->  21			
16717A	Setup			
333MHz St.	Waveform<1>			
	Timing Zoom<1>			
	Listing<1> 🖌			
Source Viewer<1>				
	Run 🕨			

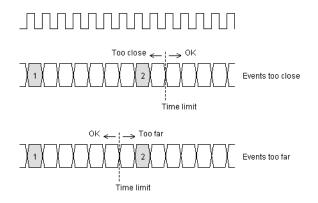
	State Number	ADDR	DATA	STAT	data[0-7]	Time
	Decimal	Hex	Hex	Hex	Hex	Relative
	-6	FFF035F4	384103E7	0103E7	38	
	-5	FFF03633	C44103E7	0903E7	C4	10.924 us
	-4	FFF035F4	384103E7	0103E7	38	196.000 ns
	-3	FFF03633	C44103E7	0903E7	C4	10.928 us
	-2	FFF035F4	384103E7	0103E7	38	192.000 ns
	-1	FFF03633	C44103E7	0903E7	C4	10.932 us
tr	0	FFF03634	3D4103E7	0103E7	3D	272.000 ns
	1	FFF03635	804103E7	0903E7	80	116.000 ns
	2	FFF03636	004103E7	1103E7	00	116.000 ns
	3	FFF03637	004103E7	0903E7	00	120.000 ns
	4	FFF03638	384103E7	0103E7	38	272.000 ns
	5	FFF03639	004103E7	0903E7	00	116.000 ns
	6	FFF0363A	004103E7	1103E7	00	120.000 ns

If the analyzer never triggers, you can look at the run status message line to see which sequence levels are visited, and you can learn more about why the trigger never occurred.

#### See Also

"If the trigger doesn't occur as expected" on page 309

### To find events that are too close or too far



You can measure time by using a timer or by counting states or the occurrences of an event.

Possible uses:

- To detect when a subroutine is exited prematurely from any number of exit points (events too close, not enough cycles between subroutine entry and exit).
- To find a protocol violation in sending control messages to a peripheral (where events that are too close violate the protocol).
- To trigger when secondary cache must be accessed between 2 consecutive memory reads, producing extra cycles (events too far, that is, there are too many cycles between consecutive memory reads).
- To detect when an interrupt routine is executing for an excessive number of cycles (events too far, that is, there are too many cycles between interrupt entry and exit).
- To examine code execution when circuitry issues a data request interrupt more than N times during the execution of a time-critical subroutine (events too far, that is, there are too many interrupts between subroutine entry and exit).
- To trigger if a loop is executed more than 10 times between 2 nonconsecutive routines (events too far, that is, there are too many loop executions between two routines).

Probing the Target 1. Configure a state analysis machine. System

Expander Card (Master: E)
16717A Setup
333MHz Sta Waveform(1)
Timing Zoom<1>
Listing(1)
Source Viewer(1)
Run 🖻
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E)
<ul> <li>Timing Mode - Asynchronous sampling clocked internally by analyzer</li> <li>State Mode - Synchronous sampling clocked by the Device Under Test</li> </ul>

2. Select the state analyzer's clock input.

ļ,	State Mode Controls									
	167 MHz / 2M State 🛃 Trigger Position Center 👤									
	Acquisition depth 2M									
	Clock Setup									
	Mode: Master only 🛃 🔄 Advanced Clocking									
	Pod E4 E3 E2 E1									
	Clock M L K J									
	Activity _ ‡ ‡ ‡									
	Master Off Off O 🗲 => [(J↑) • (K=0)]									

3. Assign pods if necessary.

Sampling Format Pod Assignment			
		Pod Assignment	
Analyzer 1		Analyzer 2	
Name: Analyzer <e></e>	Name:	Analyzer <e2></e2>	
Type: State 🗆	Type:	Off 🖃	Unassigned Pods
<b>E1: ***********</b> J <b>*</b>	E3:	\$\$**\$\$\$\$ L \$	D3: L _
E2: *******************	E4: -	M _	D4:M _
D1: _*********** _ * * J * D2: **********			

4. Format labels for the signals on which you will look for the events.

Sampling F	format	] Tra	igger   Symbol			
Pod Assignment		Pod D2	Pod D1	Pod E2		
	_		TTL	TTL	TTL	
Setup/Hold	-」 ‡‡ ĸJ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	<b>***********</b>	‡: 1
ADDR	+	••			******	*
DATA	+	•••	*****	****		•
STAT	+ **	**	*	****		•
data[0-7]	+	•••	******			•

### **Capturing the Data**

- 1. In the Trigger window, replace level 1 of the trigger specification with one of the following trigger functions:
  - Find pattern2 occurring too soon after pattern1
  - Find too few states between pattern1 and pattern2
  - Find pattern2 occurring too late after pattern1
  - Find too many states between pattern1 and pattern2

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Default S	Storing Status Save/Recall
General State	Trigger function libraries
Run until user stop Find pattern2 occurring immediately after Find pattern1 eventually followed by patter Find pattern2 occurring too soon after pat Find pattern2 occurring too late after pat	pattern 1 duration — I pattern 2
Replace N Insert before	Insert after Delete

2. In the trigger definition, specify the patterns and enter the time limit.

Trigger definition
1 FIND PATTERN2 OCCURRING TOO SOON AFTER PATTERN1
Find ADDR = FFF03648 Hex
less than 72 us
after ADDR = FFF034D8 Hex
Use Timer 1
then Trigger and fill memory

- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Listing display to show that fewer or more than N cycles, events, or some amount of time occurred between the two events.

Expand	ler Card (Master: E)
Select	->
16717A	Setup
333MHz St	Waveform<1>
	Timing Zoom<1>
	Listing<1> 🖌
	Source Viewer<1>
	Run 🕨

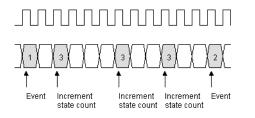
		Markers	Comments 1	Analysis	Y	
56	earch   Goto	Signal				
La	bel ADDR	▲ Next Prev				
0	dvanced search					
H	uvanceu searcr					
[			( <b>F</b>			
	State Number	ADDR	Time	DATA	STAT	data[0-7]
	Decimal	Hex	Absolute	Hex	Hex	Hex
	-340	000041B0	-54.752 us	004123D7	0123D7	00
	-339	000041B1	-54.636 us	024123D7	0923D7	02
	-338	000041B2	-54.516 us	604123D7	1123D7	60
	-337	000041B3	-54.400 us	F14123D7	0923D7	F1
	-336	FFF03184	-54.128 us	484103E7	0103E7	48
	-335	FFF03185	-54.008 us	004103E7	0903E7	00
	-334	FFF03186	-53.892 us	034103E7	1103E7	03
	-333	FFF03187	-53.776 us	554103E7	0903E7	55
G1_	-332	FFF034D8	-53,580 us	7C4103E7	0103E7	70
	-331	FFF034D9	-53.464 us	084103E7	0903E7	08
	-330	FFF034DA	-53.344 us	024103E7	1103E7	02
	-329	FFF034DB	-53.228 us	A64103E7	0903E7	A6
	-328	FFF034DC	-52,956 us	7C4103E7	0103E7	70
	-327	FFF034DD	-52.836 us	2B4103E7	0903E7	2B
	-326	FFF034DE	-52.720 us	0B4103E7	1103E7	0B
	-325	FFF034DF	-52,604 us	784103E7	0903E7	78
	-324	FFF034E0	-52.332 us	944103E7	0103E7	94

#### See Also

"Use trigger functions for easy measurement set up" on page  $305\,$ 

"If the trigger doesn't occur as expected" on page 309

### To count occurrences of an event between two events



Possible uses:

- To verify that a memory refresh routine is executing the number of times expected.
- To count the number of memory write cycles within a segment of code.

Probing the Target 1. Configure a state analysis machine. System

Expander Card (Master: E)
Select $\rightarrow$
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E) 7 On 26Hz 2001 Timing Zoom
$\diamond$ Timing Mode – Asynchronous sampling clocked internally by analyzer
$\clubsuit$ State Mode – Synchronous sampling clocked by the Device Under Test

2. Select the state analyzer's clock input.

ļ,	State Mode Controls									
	167 MHz / 2M State 🛃 Trigger Position Center 👤									
	Acquisition depth 2M									
	Clock Setup									
	Mode: Master only 🛃 🔄 Advanced Clocking									
	Pod E4 E3 E2 E1									
	Clock M L K J									
	Activity _ ‡ ‡ ‡									
	Master Off Off O 🗲 => [(J↑) • (K=0)]									

3. Assign pods if necessary.

Sampling Pod Assignmer	Format Data			
-			Pod Assignment	_
Ar	nalyzer 1		Analyzer 2	1
Name: An	alyzer <e></e>	Name:	Analyzer <e2></e2>	
Type: St	ate =	Type:	Off =	Unassigned Pods
E1: ####	*********** J *	E3: -	\$\$**\$\$\$_\$ L \$	D3:L _
E2: ‡‡‡‡	<b>‡‡‡‡‡</b> <sup>+</sup> <sup>−</sup> <sup>−</sup> K ‡	E4: _	M _	D4:M _
D1: _‡‡‡	********* J *			
D2: ****	<b>‡‡‡‡‡</b> ‡‡ K ‡			

4. Format labels for the signals on which you will look for the events.

Sampling Format Trigger Symbol								
Pod Assignment	Clock: L.DE.		Pod D2	Pod D1	Pod E2			
			TTL	TTL	TTL			
Setup/Hold		‡‡ KJ	<b>***********</b>		<b>***********</b>			
ADDR +		•••			******			
DATA +		•••	*****	*****				
STAT +	**	**	*	****				
data[0-7] +		•••	******					

Capturing the Data1. Set up a trigger sequence where level 1 looks for the start event; when it's<br/>found, level 2 looks for the second event; when it's found trigger the<br/>analyzer.

Sampling Format	Trigger   Symbo	1]			
Trigger Functions Set	tings Overview	Default	Storing Status Sa	ave/Recall]	
General State			Trigger f	unction libr	aries.
Run until user stop Find pattern2 occurri Find pattern1 eventua Find pattern2 occurri Find pattern2 occurri	<mark>lly followed by</mark> ng too soon afte	patte er pat er pat ∭	pattern 1	patte	
	Insert befor		Insert after		cer.
Trigger definition          Insert LEVEL before         Insert LEVEL         Delete LEVEL         Cut LEVEL         Paste LEVEL         Paste LEVEL         Paste LEVEL         Braste LEVEL	ац ХХ 111	1 If then 2 If	r definition ADDR = FFF034D8 H occurs 1 time Store sample Goto 2 ADDR = FFF04 occurs 1 Store sample Trigger and fill m	4C7 Hex	

2. Store the start and end events and the event you want to count.

Sampling Format Trigger Symbol	1
Trigger Functions Settings Overview D	efault Storing Status Save/Re
Specify what to store by default. Using store actions in a sequence level overrides the default storing.	Store if - ADDR = FFF02B38 Hex
Store by default Custom -	
Initially, default storing is 🛛 🗖	
	Group events

3. Change the count qualifier to count the event you are interested in.

Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Stori
Acquisition depth 2M
Trigger Position Center
Count States - Define ADDR = FFF02B38 Hex

4. Select the Run button to start the measurement.

# **Displaying the Data** 1. When the analyzer triggers, use the Listing display to show the start and end events. The difference in count values shows the number of times the count event occurred between the start and end events.

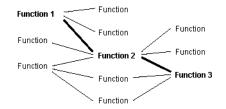
Expand	ler Card (Master: E)
Select	->
16717A	Setup
333MHz St.	Waveform<1>
	Timing Zoom<1>
	Listing<1>
	Source Viewer(1)
	Run 🕨

	1		
	State Number	ADDR	State Counts
	Decimal	Hex	Absolute
	-6	FFF02B38	-4
	-5	FFF02B38	-3
	-4	FFF034D8	-3
	-3	FFF02B38	-2
	-2	FFF02B38	-1
	-1	FFF02B38	0
62	0	FFF044C7	0
	1	FFF02B38	1
	2	FFF02B38	2
	3	FFF02B38	3
	4	FFF02B38	4
	5	FFF02B38	5

If the analyzer never triggers, the start or end events were never found. Look at the run status message to see which sequence levels are visited; this will tell you which event was not found.

"If the trigger doesn't occur as expected" on page 309

### To trigger on a function call sequence



Possible uses:

- To trigger when procedure 3 displays an error message, but only when it's called by procedure 2 and procedure 1 before that.
- To trigger on the 3rd nested occurrence of a recursive subroutine.

1. Configure a state analysis machine.

#### Probing the Target System

See Also

Expander Card (Master: E)	
16717A Setup	
333Hz Sta Waveform(1)	
Timing Zoom(1)	
Listing(1)	
Source Viewer(1)	
Run	
	_
Sampling Format Trigger Symbol	
Analyzer Name: Analyzer (E)	>m
$\diamond$ Timing Mode – Asynchronous sampling clocked internally by analyzer	
$\bullet$ State Mode – Synchronous sampling clocked by the Device Under Tes	t

2. Select the state analyzer's clock input.

State Mode	Cont	rols			
167 MHz /	2M 9	itate	. ₹		Trigger Position Center 👤
Acquisitio	n de	pth[	2M	F	
Clock Set	up—				
Mode: Ma	ster	only	y I		Advanced Clocking
Pod	E4	E3	E2	E1	
Clock	М	L	К	J	
Activity	-	‡	\$	\$	
Master	Off	Off	0	F	=> [(J†) • (K=0)]

3. Assign pods if necessary.

Pod Assignment.		
	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: Analyzer <e></e>	Name: Analyzer <e2></e2>	
Type: State 😑	Type: Off =	Unassigned Pods
E1: **************	E3:‡‡-‡‡‡‡L ‡	D3: L _
E2: <b>#########</b> ##K #	E4: M _	D4:M _
<b>D1:</b> _ <b>‡‡‡‡‡‡‡‡‡‡‡‡‡</b>		
<b>D2: ########</b> ###########################		

Sampling Format

4. Format labels for the signals on which you will look for the program flow events.

Sampling Fo	rmat	Tri	igger   Symbol			
Pod Assignment	Clocks		Pod D2	Pod D1	Pod E2	
			TTL	TTL	TTL	
Setup/Hold	‡‡ KJ	‡‡ KJ	<b>**********</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡</b> _ <b>‡‡‡</b> 15 87 0	<b>‡‡‡‡‡‡‡‡‡‡‡</b> 0	‡ 1
ADDR +				•••••	*****	*
DATA +		••	*****	*****	•••••	ŀ
STAT +	**	**	*	*****		
data[0-7] +		•••	******			

5. Load symbols from your program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	

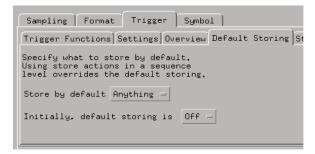
Capturing the Data
1. Set up a trigger sequence where level 1 looks for the procedure 1; when it's found, level 2 looks for procedure 2 or the end of procedure 1 (which if found will restart the search). Level 3 looks for procedure 3 (which if found will trigger the analyzer) or the end of procedure 2 (which if found will branch back to level 2).

Sampling   Format   ]	Frigger   Symbol			
Trigger Functions Set	tings Overview Defaul	t Storing St	tatus Sa	ve/Recall]
General State		Ti	rigger fu	Inction libraries
Wait n external clock Wait for arm in Advanced - If/then Advanced - 2-way branc Advanced - 3-way branc	:h		<	F
Replace 📐	Insert before	Insert af	fter	Delete
General State Wait n external clock Wait for arm in Advanced - If/then Advanced - 2-way branc Advanced - 3-way branc	h	Insert af	<	Inction libraries
Keplace	Insert before 🦹	Insert af	ter	
Sampling Format	Frigger Symbol			
Trigger Functions Set General State	tings Overview Defaul			ve/Recall] Inction libraries
Wait n external clock Wait for arm in Advanced - If/then Advanced - 2-way branc Advanced - 3-way branc	:h		<	ĨF
Replace	Insert before 📐	Insert af	fter	Delete

2. In the trigger definition, specify the program flow events.

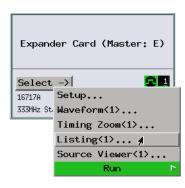
Trigger Sequence
1 If ADDR = oard/source/q.elf:update_display Symbols occurs 1 time then Store sample Goto Next
<pre>2 If ADDR = ard/source/q.elf:clear_hist_buff Symbols</pre>
3 If ADDR =demo_board/source/q.elf:do_sort Symbols occurs 1
Else if ADDR = ource/q.elf:clear_hist_buff+0x5F Symbols then Store sample Goto 2

3. Set up default storing to be initially off.



4. Select the Run button to start the measurement.

**Displaying the Data** 1. When the analyzer triggers, use the Listing display to show the particular sequence was captured.



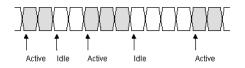
	State Number	ADDR	DATA	STAT	data[0-7]	Time
	Decimal	Symbols	Hex	Hex	Hex	Absolute
	-8	ec:update_display	7C4103E7	0103E7	7C	-52.755 ms
	-7	update_displ+01DB	D04103E7	0903E7	DO	-52,626 ms
	-6	ec:update_display	7C4103E7	0103E7	7C	-37,931 ms
	-5	update_displ+01DB	D04103E7	0903E7	DO	-33.842 ms
	-4	ec:update_display	7C4103E7	0103E7	70	-18.305 ms
	-3	update_displ+01DB	D04103E7	0903E7	DO	-18.176 ms
	-2	ec:update_display	7C4103E7	0103E7	7C	-20.144 us
	-1	e:clear_hist_buff	7C4103E7	0103E7	7C	-4.840 us
62	0	<pre>q.el:ecs2:do_sort</pre>	3D4103E7	0103E7	3D	0 s
	1	ecs2:do_sort+0001	804103E7	0903E7	80	116.000 ns
	2	ecs2:do_sort+0002	004103E7	1103E7	00	232.000 ns
	3	ecs2:do_sort+0003	004103E7	0903E7	00	348.000 ns
	4	ecs2:do_sort+0004	384103E7	0103E7	38	624.000 ns
	5	ecs2:do_sort+0005	004103E7	0903E7	00	740.000 ns

If the analyzer never triggers, look at the run status message to see which sequence levels are visited; this will tell you which event was not found.

See Also

"If the trigger doesn't occur as expected" on page 309

### To analyze bus occupation & bandwidth (with SPA)



Bus occupation and bandwidth measurements generally show the

amount of idle bus states among all bus states.

Requirements:

• This measurement requires the system performance analyzer (SPA) tool set.

Possible uses:

- To show the share of the workload that each processor in a multipleprocessor system carries, or to determine if the system is balanced.
- To analyze headroom by examining the percentage of idle bus states.
- To analyze cache hits and misses.

Probing the Target System

- 1. Probe the bus you wish to analyze.
- 2. Configure a state analysis machine.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E)
$\diamond$ Timing Mode – Asynchronous sampling clocked internally by analyzer
$\blacklozenge$ State Mode $\ -$ Synchronous sampling clocked by the Device Under Test

3. Select the state analyzer's clock input.

—State Mode	-State Mode Controls										
167 MHz /	167 MHz / 2M State 🛓 Trigger Position Center 🛓										
Acquisitio	Acquisition depth 2M										
Clock Set	up—										
Mode: Mas	Mode: Master only 🛓 🔄 Advanced Clocking										
Pod	E4	E3	E2	E1							
Clock	М	L	К	J							
Activity	Activity _ + + +										
Master	Off	Off	Off	₽	=> J†						

4. Assign pods if necessary.

	Sampling Format								
-			Pod Assignment						
	Analyzer 1		Analyzer 2						
Name:	Analyzer <e></e>	Name:	Analyzer <e2></e2>						
Type:	State =	Type:	Off =	Unassigned Pods					
E1: ‡	*************** J *	E3:	\$\$**\$\$\$_\$ L \$	D3:L _					
E2: ‡	<b>‡‡‡‡‡‡</b> <sup>−−</sup> K <b>‡</b>	E4: _	M _	D4:M _					
D1: _	************* J *								
D2: ‡	\$\$\$\$\$\$\$\$\$\$\$\$\$\$								

5. Label the logic analyzer signals. If you are using an analysis probe, you can configure and label signals with provided configuration files.

Sampling Format Trigger Symbol									
Pod Assignment	Cloc LDE	ks F	Pod D2	Pod D1	Pod E2				
			TTL	TTL	TTL				
Setup/Hold	<b>‡‡</b> KJ	‡‡ KJ	<b>**********</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡</b> _ <b>‡‡‡</b> 15 87 0	<b>************</b> _*0	‡: 15			
ADDR	+			•••••	****	**			
DATA	+	•••	*****	*****	•••••				
STAT	+ **	**	*	*****					
data[0-7]	+	••	******	••••••	•••••				

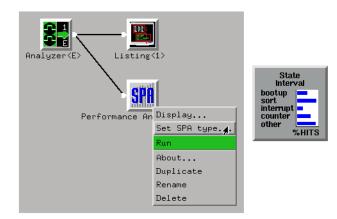
**Capturing the Data** 1. Set up a trigger specification to capture all activity on the bus.

Sampling Format Trigger Symbol								
Trigger Functions Settings Overview Defau								
General State								
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop								
Replace Insert before								
Trigger definition           1         FIND PATTERN N TIMES								
Find 1								
ADDR = XXXXXXXX Hex								
then Trigger and fill memory								

2. Select the Run button to start the measurement.

**Displaying the Data** 

1. Use the System Performance Analyzer's State Interval display to view the measurement result.



2. Define a state range that corresponds to idle bus states.

- Pe	rformance Analysis<1>							
File Window Graph Options Sort								
Define ranges Quali	ify data Include other							
	State Interval							
Def	fine Ranges[STAT] – Performance Analysis<1>							
*								
Select Label	Current ranges							
ADDR ADDR_TZ	Idle Pattern XX1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX							
DATA								
data[0-7] data[0-7]_TZ								
DATA_TZ STAT								
STAT_TZ								
	Edit ranges							
	Not_Idle Binary - Pattern - XX0XXXX							
¥	Add Replace Delete Symbo							
0К	Apply Cancel							

There is no limit to the number of ranges that can be simultaneously defined and displayed. The ranges can be sorted alphabetically or by number of hits.

3. Run the measurement and view the results.

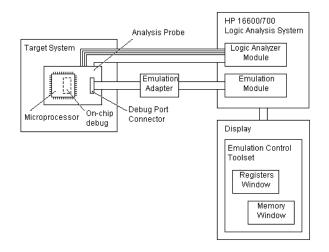
	State 3	Interval
[STAT] Ranges	Hits(%)	
Idle	792,747(76.2%)	
Not_Idle	247,637(23,8%)	
	0% 10%	20% 30% 40% 50% 60% 70% 80% 90% 100%
Display Informa	ation	Bucket Information
Qualified State:	s: 1,040,384	Name:
Total Acquisiti	ons: 1	Hits(%):
Total Display %		Range:
Total Buckets:	2	Min/Max:
TOCAL BUCKEES;	2	File:



"To view bus activity" on page 174

# Exercising the Microprocessor (with the Emulation Probe)

- "To initialize registers, access memory" on page 137
- "To use the emulation probe as a test tool" on page 140



## To initialize registers, access memory

**Requirements:** 

• Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

• You need either the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system or you need third-party debugger software to control the microprocessor debug interaction.

Possible uses:

- To test microprocessor access to target system memory or I/O.
- To modify the contents of microprocessor data or configuration registers.
- To prepare the target system for downloading code to RAM.

# Probing the Target1. MaSystemad.

- 1. Make sure the emulation probe (or emulation module and emulation adapter) has been connected to the target system.
- 2. Set up the emulation control tool set or third-party debugger connection to the emulation probe.

**Starting the Emulation** 1. Start an emulation module session. **Control Software** 

Select -> 15,6,253,125	Connect – Em	ulator<1>	
E5901A Em. Connect to Emulator A Motorola & Update Firmware Performance Verification	No active connection: Emul Motorola 800 PowerPC Emula		
Help	Connect to Emulator	Disconnec Emulat	
	Help	Clos	

If you have third-party debugger software (on a computer in the network), start that software, and connect it to the emulation probe.

Accessing/Modifying Registers, Memory, or I/O

1. In the emulation control tool set, open the Register, Memory, or I/O window, display the locations you're interested in, and modify particular locations.

Select	-> 15.6.253.125				
E5901A Emu	Disconnect from Emulator				
Motorola 8	Run Control				
	Load Executable				
	Breakpoints				
	Registers				
	Memory 🖌				
	I/0				
	Memory Disassembly				
	Status/Error Log				
	Command Line				
	Setup				

-		Mem	ory – Emulator 1								
F	File Navigate Update										
	Write 00000000 to 10 Apply Read from 409d Apply Base of Data Hex -										
F	Address Data			ASCII da	ta						
	000040AC         20202020         202           000040AEC         20202020         202           000040DE         43757272         656           000040DE         54617267         696           000040FC         00202020         202           000040EC         4379636C         657           000040FC         00202020         202           0000410C         4379636C         657           0000411C         00202020         202           0000411C         41202020         202	02020 20202020 02020 20202020 02020 5465ED20 E7420 20333920 42020 20333420 46520 20363020 02020 20202020 32020 202020138 02020 20202020 02020 20202020 02020 20202020	20202020 48756D20 20202020 20202020 20202020 20202020 20202020 39303832 20202020 20202020	COOL Current Target Outside Ċycles Ă	 Tem Hum 39 34 60 189082						

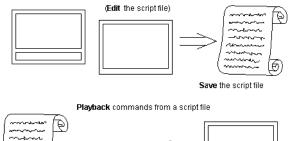
If you are using a third-party debugger, perform these tasks using its interface.

See Also "To download boot code" on page 144

"To use the emulation probe as a test tool" on page  $140\,$ 

## To use the emulation probe as a test tool

Open the Command Line window





Requirements:

• Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

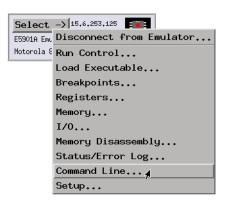
• You need the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system.

#### Possible uses:

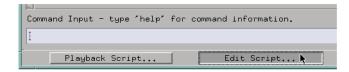
- To automate a sequence of register, memory, or I/O access commands.
- To control the execution of the microprocessor as part of a system test.
- Probing the Target<br/>System1. Make sure the emulation probe (or emulation module and emulation<br/>adapter) has been connected to the target system.
  - 2. Set up the emulation control tool set connection to the emulation probe.



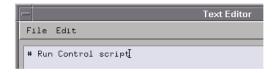
Creating/Editing the Command Script 1. Access the Command Line window.



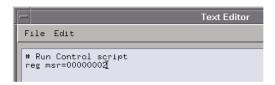
2. Select the Edit Script button.



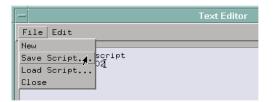
3. Enter the comment line that identifies the file as a run control script.



4. Enter commands.



5. Save the script to a file.



Playing Back the<br/>Command Script1. Select the Playback Script button and choose the run control script file.

Comma	and	Input	-	type	"help"	for
Ĭ						
		Playba	ick	Scri	pt 📐	

See Also

"To initialize registers, access memory" on page  $137\,$ 

# Firmware Development

Testing Boot Code (with the Emulation Probe)

- "To download boot code" on page 144
- "To start or stop processor execution" on page 147
- "To stop processor execution using breakpoints" on page 149
- "To capture startup execution" on page 152

Making Driver Development Measurements

- "To trigger on an 8-bit serial pattern" on page 155
- "To view serial data in parallel" on page 160
- "To capture driver execution (& view HW and SW)" on page 165
- "To capture execution up to a failure or halt" on page 171
- "To view bus activity" on page 174
- "To capture simple program messages" on page 175
- "To trigger on packet data (with DataComm Analysis)" on page 177

Making Interrupt Service Routine Measurements

- "To capture interrupt frequency and type" on page 183
- "To measure interrupt latency and execution time" on page 186
- "To simulate particular interrupt sequences" on page 191
- "To view the occurrence rate of an event (with SPA)" on page 192

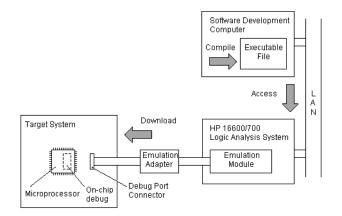
# Testing Boot Code (with the Emulation Probe)

- "To download boot code" on page 144
- "To start or stop processor execution" on page 147
- "To stop processor execution using breakpoints" on page 149

Chapter 1: Measurement Examples Firmware Development

• "To capture startup execution" on page 152

### To download boot code



Requirements:

• Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

• You need the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system.

#### Possible uses:

• To move boot code into target system RAM for execution.

#### Probing the Target System

- **et** 1. Make sure the emulation probe (or emulation module and emulation adapter) has been connected to the target system.
  - 2. Set up the emulation control tool set connection to the emulation probe.

Select -> 15.6.253.125	– Connect – E	mulator<1>
E5901A Emu Connect to Emulator 4 Motorola E Update Firmware	No active connectio	on: Emulation
Performance Verification	Motorola 800 PowerF	°C Emulator
Help	Connect to Emulator	Disconnec Emulat
	-	
	Help	Clos

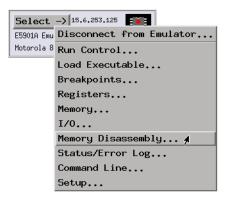
Downloading Boot Code 1. Access the Load Executable window.

Select	-> 15.6.253.125
E5901A Emu	Disconnect from Emulator
Motorola 8	Run Control
	Load Executable 🖌
	Breakpoints
	Registers
	Memory
	I/0
	Memory Disassembly
	Status/Error Log
	Command Line
	Setup

-	Load Executable - Emulator<1>
F	ile Window
$\left  \right $	Operation
	Load Options
	Access Size: 4 bytes = Set PC after load
	File Format: Motorola S-records = Start Address: 10
	/logic/demo/860_demo_board/download/demo.srec Browse

- 2. Select the appropriate file format, options, and executable file name.
- 3. Select the Apply button to download the executable file.

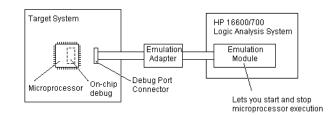
Verifying the Download 1. You can verify the download by looking at the memory locations in disassembled format.



	-	Memory Disass	embly – Emulato	or 1
l	File Navigate Upo	date		
	Starting Address: Page Back Page F		a Column Widtł	n: 15 🔺
	00008000 00008004 00008008 0000800C 00008010 00008014 00008018 0000801E 0000801C	0000405C 7C0802A6 7C280B78 9421FFE0 480007F5 7C7D1B78 7C9C2378 7CB2278 3C400001	Undefin mfspr mr stwu bl mr mr mr lis	ed Opcode 00004 r0 lr r1 r1 r1 FFE0(r1) 00008804 r29 r3 r28 r4 r27 r5 r2 0001

See Also

"To initialize registers, access memory" on page 137 (if you need to initialize registers before code download)



### To start or stop processor execution

#### **Requirements**:

• Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

• You need either the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system or you need third-party debugger software to control the microprocessor debug interaction.

#### Possible uses:

- To control the target system boot up sequence.
- To view the state of the microprocessor at particular points during program execution.

## Probing the Target1. Make sure the emulation probe (or emulation module and emulationSystem1. Make sure the emulation probe (or emulation module and emulation<br/>adapter) has been connected to the target system.

2. Set up the emulation control tool set or third-party debugger connection to the emulation probe.

# Starting the Emulation1. If you have the emulation control tool set in your Agilent TechnologiesControl Software1. If you have the emulation control tool set in your Agilent Technologies16700A/B-series logic analysis system, drag the icon to the workspace, and<br/>connect to the emulation probe.



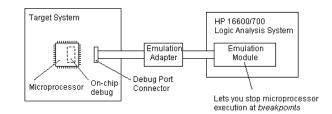
If you have third-party debugger software, start that software, and connect to the emulation probe.

Controlling Processor1.In the emulation control tool set, open the Run Control window, and select<br/>the Run or Stop buttons.



If you have a third-party debugger, perform these tasks using its interface.

See Also "To stop processor execution using breakpoints" on page 149



### To stop processor execution using breakpoints

#### **Requirements:**

• Your target system microprocessor must have on-chip debug circuitry that an emulation probe can work with.

The emulation probe connects to a debug port connector on the analysis probe or to a debug port connector designed into your target system.

• You need either the emulation control tool set in the Agilent Technologies 16700A/B-series logic analysis system or you need third-party debugger software to control the microprocessor debug interaction.

#### Possible uses:

- To stop microprocessor execution on a particular line of source code.
- To view the state of the microprocessor at particular points during program execution.

## Probing the Target1. Make sure the emulation probe (or emulation module and emulation<br/>adapter) has been connected to the target system.

2. Set up the emulation control tool set or third-party debugger connection to the emulation probe.

# Starting the Emulation1.If you have the emulation control tool set in your Agilent TechnologiesControl Software1.16700A/B-series logic analysis system, drag the icon to the workspace, and<br/>connect to the emulation probe.



If you have third-party debugger software, start that software, and connect to the emulation probe.

**Setting Breakpoints** 1. In the emulation control tool set, open the Breakpoints window, select a breakpoint to use, and enter the address at which microprocessor execution should stop.

Select	-> 15,6,253,125
E5901A Emul	Disconnect from Emulator
Motorola 80	Run Control
	Load Executable
	Breakpoints 🖌
	Registers
	Memory
	I/0
	Memory Disassembly
	Status/Error Log
	Command Line
	Setup

			Breakpoints – Emu	ilator 1	•
Fil	le Nav	vigate			Help
	Brea	akpoint #	Memory Address	Break On	
	■ 1	(Hardware)	fff034d8	Instruction Execution	-
	□ 2	(Hardware)	fff040cc	Instruction Execution	-
	⊒ 3	(Software)	00000000		
	⊒ 4	(Software)	00000000		
Ĺ			Read Breakpo	pints	-

Hardware breakpoints can be used for addresses in target system ROM.

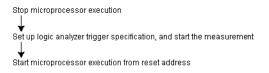
*Software breakpoints* replace existing code with a *breakpoint instruction*, so they only work for addresses in target system RAM.

If you have a third-party debugger, perform these tasks using its interface.

"To start or stop processor execution" on page 147

See Also

### To capture startup execution

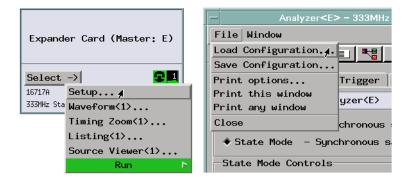


#### Possible uses:

• To verify boot code operation.

#### Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



You may also want to connect an emulation probe to the target system microprocessor's debug port (either via a connector designed into the target system or the connector provided on the analysis probe). The emulation control tool set's reset/run control lets you make this measurement without cycling power in your target system.

#### **Capturing the Data**

1. Stop microprocessor execution (either by turning OFF power to the target system or using the emulation control tool set's Reset command).



2. Set up a trigger specification to trigger on anything and capture everything.

Sampling Format	Trigger Symbol
Trigger Functions Set	tings Overview Defau
General State	
Find pattern n times Store range until pat Store pattern2 until Store nothing until p Run until user stop	pattern1 occurs
Replace	Insert before
Trigger definition	
Trigger definition	
1 FIND PATTERN N TIMES	currence of
1 FIND PATTERN N TIMES	

- 3. Select the Run button to start the measurement.
- 4. Start microprocessor execution from its reset address (either by turning power ON to the target system or by using the emulation control tool set's Reset and Run commands).

- Run	Control – E	Emulator<1	> · 🗆	
File Window Help				
Run	Stop	Step	Reset	
MPC860Emulation hard reset				

**Displaying the Data** 1. Use the Listing display to show the data that was captured when the target system started up.



	State Number	PC MPC821/860 Inverse Assembler			Time
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Absolute
tr	0	ource/q.elf:reset	andi.	r1 r1 0000	0 s
	4	/q.elf:reset+0004	oris	r1 r1 FFFF	5.324 us
	8	∕q.elf:reset+0008	ori	r1 r1 FF88	10.660 us
	12	/q.elf:reset+000C	andi.	r2 r2 0000	16.004 us
	16	∕q.elf:reset+0010	oris	r2 r2 FF00	21.348 us
	20	/q.elf:reset+0014	ori	r2 r2 0004	26.692 us
	24	∕q.elf:reset+0018	stw	r1 0000(r2)	32.032 us
	28	/q.elf:reset+001C	andi.	r1 r1 0000	37.572 us
	32	/q.elf:reset+0020	oris	r1 r1 FFF0	42.920 us
	36	/q.elf:reset+0024	ori	r1 r1 0401	48.268 us
	40	/q.elf:reset+0028	andi.	r2 r2 0000	53.612 us
	44	/q.elf:reset+002C	oris	r2 r2 FF00	58.952 us
G1_	48	/g.elf:reset+0030	ori	r2 r2 0100	64.296 us
	52	/q.elf:reset+0034	stw	r1 0000(r2)	69.640 us
	56	∕q.elf:reset+0038	andi.	r1 r1 0000	75.180 us
	60	/q.elf:reset+003C	oris	r1 r1 FFFO	80.524 us
G2_	64	/g.elf:reset+0040	ori	r1 r1 0110	85,872 us

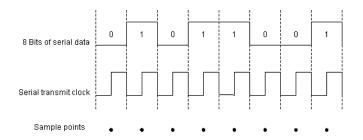
See Also

"To start or stop processor execution" on page 147

### Making Driver Development Measurements

- "To trigger on an 8-bit serial pattern" on page 155
- "To view serial data in parallel" on page 160
- "To capture driver execution (& view HW and SW)" on page 165
- "To capture execution up to a failure or halt" on page 171
- "To view bus activity" on page 174
- "To capture simple program messages" on page 175
- "To trigger on packet data (with DataComm Analysis)" on page 177

### To trigger on an 8-bit serial pattern



Possible uses:

- To view system activity after pattern transmission.
- To look at system status when an error pattern is detected.

#### Probing the Target System

- 1. Connect a logic analyzer probe to the target system's serial signal, and connect the analyzer's clock input probe to the serial transmit/receive clock signal.
  - 2. Configure a state analysis machine and specify the serial transmit/receive clock as the analyzer's clock input.

Or:

Configure a timing analysis machine and make the sample rate the same as

the serial transmit/receive clock.

Expander Card (Master: E)	
Select ->	_
16717A Slot E: Analyzer(E)	>
333MHz Sta Slot E: Analyzer (E2)	Setup
Run /	Waveform(2)
	Timing Zoom<2>
	Listing<2>
	Source Viewer<2>
Sampling Format Trigger	Symbol ]
Analyzer Name: Analyzer(E2	≥ I On ZGHZ Patient Timing Zoom
Timing Mode - Asynchronou	s sampling clocked internally by analyzer
💠 State Mode – Synchronous	s sampling clocked by the Device Under Test
-Timing Mode Controls	
333 MHz Full Channel 2M Sa	mple 🛓 Trigger Position Center 🛓
Acquisition Depth 2M	
Sample Period 3.0ns	

3. Format a label for the signal on which you will look for the serial pattern.

Sampling Format Trigger Sy	mbol ]	
Pod Assignment	Pod E4	Pod E3
	TTL	TTL
	ī5 <sup></sup> 87 <sup></sup> 0	15 87 0
CAN_TXD +		*

Capturing the Data1. Set up a trigger sequence where level 1 looks for the first logic level of the<br/>pattern; when the pattern is found, level 2 looks for the second logic level<br/>of the pattern, etc. If, in each of levels 2 through 8, the appropriate logic

level is not found, branch back to level 1 to start looking for the serial pattern again.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Stat	us
General Timing	
Wait t seconds	
Wait for arm in Advanced - If/then <mark>Advanced - 2-way branch</mark>	
Advanced - 2-way branch Advanced - 3-way branch	
Replace Insert before	2
	_
Trigger definition           1         If CAN_TXD         0         Binary	_
occurs 1 time	
then Goto Next	
2 If CAN_TXD = 1 Binary	
Else if CAN_TXD ≠ 1 Binary then Goto 1	
	_
3 If CAN_TXD = 1 Binary occurs 1 time	
then Goto Next	
Else if CAN_TXD ≠ 1 Binary then Goto 1	
	-
4 If CAN_TXD = 0 Binary occurs 1 time	
then Goto Next	
Else if CAN_TXD ≠ 0 Binary then Goto 1	
5 If CAN_TXD = 0 Binary	
Else if CAN_TXD ≠ 0 Binary	
then Goto 1	
6 If CAN_TXD = 1 Binary occurs 1 time	
occurs 1 time then Goto Next	
Else if CAN_TXD ≠ 1 Binary then Goto 6	
7 If CAN_TXD = 1 Binary	
Else if CAN_TXD ≠ 1 Binary then Goto 1	
8 IF CAN_TXD = 0 Binary	
occurs 1	
then Trigger and fill memory	
Else if CAN_TXD ≠ 0 Binary	
then Goto 1	

To look look for longer (or shorter) patterns, use more (or fewer) sequence levels.

- 2. Select the Run button to start the measurement.
- **Displaying the Data** 1. When the analyzer triggers, use the Waveform display to show the 8-bit serial pattern was captured.

Expander Card (Master: E) Select -> Card (Master: E) 16717A Slot E: Analyzer(E> >
333MHz Sta Slot E: Analyzer(E2) Setup
Run 🕨 Waveform<2> 🕯
Timing Zoom<2>
Listing<2>
Source Viewer<2>
Search Goto Markers Comments Analysis Mixed Signal Goto Time 🛃 🔯 s 🛃 Goto
Trigger Beginning End G1 G2
Trigger Beginning End G1 G2
G1 & G2 Centered
B1 & G2 Fentered       Seconds/div =       [2,000 us]       I       Delay       I

If the analyzer never triggers, the serial pattern does not occur. Depending on the level that was reached in the sequence above, you can see how much (or how little) of the pattern was found.

"To view serial data in parallel" on page 160

See Also

"If the trigger doesn't occur as expected" on page 309

### To view serial data in parallel

Serial Data Stream Synchronize Conversion Convert this Data to this Pattern 101011011011 01001100 10100111 01010111 00110001 11101010 10111101 Display of 8-bit Parallel Word 01001100 4 10100111 🗲 01010111 + 00110001 \* 11101010 4 10111101 4

**Requirements:** 

• This measurement requires the serial to parallel tool set.

Possible uses:

• To view serial data in a format consistent with a particular protocol format.

Probing the Target System

- 1. Connect a logic analyzer probe to the serial data signal, and connect the serial data transmit (or receive, etc.) clock signal to one of the analyzer's clock input channels.
- 2. Configure the logic analyzer as a state analyzer and use the serial transmit/ receive clock as the analyzer's clock input.

Or:

Configure the analyzer as a timing analyzer and use the Serial To Parallel tool's *Clock Recovery* option.

Expander Card (Master: E) Select -> 16717A 333MHz Sta Slot E: Analyzer <e> Run N</e>	Jetup		
Sampling Format Trigger			Timing Zoom
<pre>* Timing Mode - Asynchronous</pre>	s sampling clocked in	ternally by	, U
Timing Mode Controls	mple 🛃 Trigger Posit	tion Center	• <u>•</u>
Acquisition Depth 2M 🛓			
Sample Period 3.0ns			

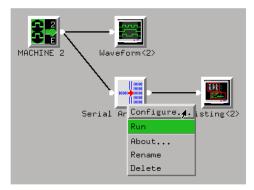
3. Format a label for the serial data channel.

Sampling Format Trigger Sy	mbol ]	
Pod Assignment	Pod E4	Pod E3
ineergraneret.tt	TTL	TTL
	15 87 0	15 87 0
CAN_TXD +	·····	*

Capturing the Data1. Set up a trigger specification to capture data on the serial channel. You<br/>may want to trigger on a particular pattern and capture data that occurs<br/>after that pattern.

Sampling Format Trigger Symbol			
Trigger Functions Settings Overview Status Save/Rec	all		
General Timing	Trigger f	unction	libraries.
Find width violation on pattern/pulse			
Find pattern present for > duration Find pattern present for < duration Find pattern absent for > duration	pa   duratior	ttern	_X_
Replace Insert before Insert	after	I	Delete
Trigger definition			
I       FIND PATTERN PRESENT FOR > DURATION         Find CAN_TXD       =       1         Hex       present for > 1       ms         then       Goto       Next			
2 FIND PATTERN PRESENT FOR > DURATION			
Find CAN_TXD = 0 Hex			
present for > 300 ns			
then Trigger and fill memory			

- 2. Configure the measurement workspace so that the logic analyzer feeds the Serial To Parallel tool, which in turn feeds the Listing display.
- 3. Open the Serial To Parallel tool and set up the type of conversion.



💠 Disable Serial Analysis 🔶 E	nable Serial Analysis
Input Label (Serial)	Output Label (Parallel)
CAN_TXD CAN_TXD_TZ	Output label Parallel
	Word width 🛛 🖉 bits
	Start on state -3
	Bit Order
Y	Advanced Options
	Enable frame processing     Define
Select input bit 0 🔺	■ Enable clock recovery Define.
💷 Invert input data	

Clock Recovery – Serial	Analysis<1>
-Clock Recovery	
Sample serial data that does	not have a clock.
For best results set the time	ing analyzer
Sample Period <= 1/4 serial	l Bit Time.
Sampled data label	[Samples
Bit Time: Sample data every	1.2000us 📕
and re-sync on data edges.	
Data Encoding Method	
🔹 Normal 💠 NRZI (edg	e = 0, level = 1)
OK Execute	Cancel

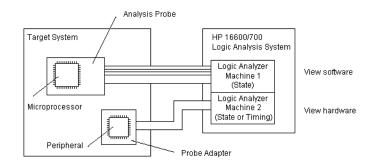
- 4. Select the Run button to start the measurement.
- **Displaying the Data** 1. Open the Listing display window to see the results of the serial to parallel conversion.

Expander Card (Master: E)					
Select	->		21		
16717A	Slot	Е:	Analyzer <e></e>		
333MHz Sta	Slot	Е:	Analyzer(E2)		Setup
			Run	Þ	Waveform<2>
					Timing Zoom<2>
					Listing<2>4
					Source Viewer<2>

	Sampled Data Label:State Number	Samples	Parallel	Time
	Decimal	Binary	Hex	Relative
	0	0		
	1	1		1,200 us
	2	1		1.200 us
	3			1.200 us
	4			1,200 us
	5	1		1.200 us
	6	1		1.200 us
G1_	.7	0	66	1,500 us
	8	0		1.200 us
	9	1		1,200 us
	10	1		1.200 us
	11	0		1,200 us
	12	0		1.200 us
	13	0		1.200 us
	14	0		1.200 us
G2.	15	1	61	1,500 us



"To trigger on an 8-bit serial pattern" on page  $155\,$ 



### To capture driver execution (& view HW and SW)

Possible uses:

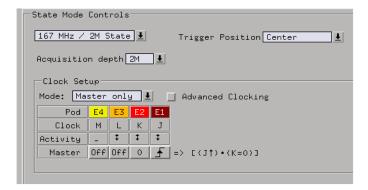
• To view and correlate driver software execution with hardware signals.

Probing the Target System

- 1. Probe the microprocessor (using an analysis probe if possible).
- 2. Probe the peripheral with other logic analysis channels (and possibly a probe adapter).
- 3. Configure a state analysis machine.

Expander Card (Master: E)				
Select ->				
16717A Setup				
333MHz Sta Waveform<1>				
Timing Zoom<1>				
Listing<1>				
Source Viewer<1>				
Run				
Sampling Format Trigge	er   Symbol	1		
Analyzer Name: Analyzer(	E>	🔽 0n 🔽 🕅	oom Ti	ming Zoom
$\diamond$ Timing Mode – Asynchrono	ous sampling	clocked intern	ally by a	nalyzer
🚸 State Mode – Synchronou	us sampling	clocked by the	Device Un	der Test

4. Select the state analyzer's clock input.



5. Assign pods. Use one logic analyzer machine for analyzing the microprocessor. Create another logic analysis machine for analyzing the peripheral by specifying the Analyzer 2 type.

Sampling Format				
-	Po	d Assignment		
Analyzer 1	Ar	alyzer 2		
Name: MPC860 BUS	Name: MA	CHINE 2		
Type: State =	Type: Tim	ning =	Unassigned	Pods
E1: **************	E3:	*******_* L *	D3:	L _
E2: <sup>‡‡‡‡‡‡‡‡‡‡‡</sup> _‡ K ‡	E4:	M _	D4:	M _
D1: _*********** _*** J *				
<b>D2: ********</b> *******				

6. Specify the sampling options for the second logic analyzer machine.

Sampling Format Trigger Symbol
Analyzer Name: MACHINE 2
◆ Timing Mode - Asynchronous sampling clocked internally by analyzer
Timing Mode Controls
2M Sample Full Channel 333 MHz 🛓 Trigger Position Center 🛓
Acquisition depth 2M
Sample Period 102.0ns

7. Format labels for the signals that are probing the microprocessor.

Sampling	For	mat	Tri	igger   Symbol			
Pod Assignment	Т	Cloc L. DE		Pod D2	Pod D1	Pod E2	
				TTL	TTL	TTL	
Setup/Hold	•	‡‡ KJ	‡‡ KJ	<b>**********</b> 15 87 0	<b>********************</b>	<b>***********</b> _*0	‡ 1
ADDR	+	••	•••			*****	*
DATA	+	••	••	*****	*****		•
STAT	+	**	**	*	******		•
data[0-7]	+		••	******			•

8. Format labels for the signals that are probing the peripheral.

Sampling	Format Trigger	Symbol ]	
Pod Assignment,	Data On Clock		Pod E3
	••••	TTL	TTL
	 MI	t 1587	
CAN_CLK	+		
CAN_TXD	+		
CAN_RXD	+		
*IRQ2	+		

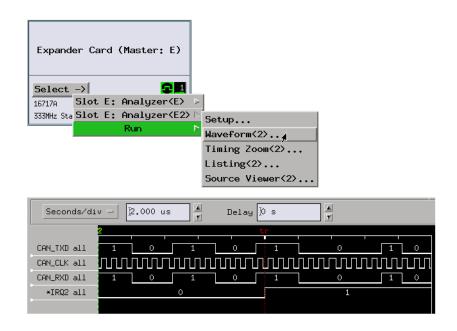
Capturing the Data1. In each machine, set up the trigger and resources for the data you want to<br/>capture.

Sampling Format	Frigger Symbol	1		
Trigger Functions Set	tings Overview De	fault Storing	]Status]Sa	ve/Recall]
General State				unction libraries
Find pattern n times Store range until patt Store pattern2 until p Store nothing until pa Run until user stop	attern1 occurs		ourrence 1	Occurrence n
Replace	Insert before	Insert	t after 🛛	Delete
Trigger definition				
1 FIND PATTERN N TIMES				
Find 1	currence of			
ADDR = board/s	ource/q.elf:update_s	ystem Symbol	s	
then Trigger and fi	ll memory			
	rigger Symbol	I		
Trigger Functions Set	tings Overview St	atus]Save/Red		
General Timing			Trigger f	unction libraries
Find pattern Find edge Find edge AND pattern Find width violation o		Ē		
Find Nth occurrence of	an edge		e	dge
				-
Replace	Insert before	Insert	: after	Delete
Trigger definition				
1 FIND EDGE				
Find <u>*IRQ2</u> Edge <u>†</u>				
then Trigger and fi	ll memory			

- 2. Select the Run button to start measurements in each logic analyzer.
- **Displaying the Data** 1. Use the Listing display to show the captured software execution and use the Waveform display to show the captured hardware signals.

Expander Card (Master: E)	
	4
16717A Slot E: Analyzer(E)	Setup
333MHz Stal Slot E: Analyzer (E2)	
Run	Waveform<1>
i Miri	Timing Zoom<1>
	Listing(1) 4
	Source Viewer<1>

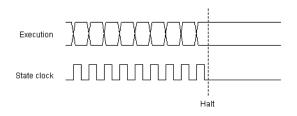
	State Number	PC	MPC821/8	60 Inverse Assembler	Time
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Absolute
	-20	q.:ecs2:main+0048	b	q.elf:ecs2:main+0018	-2,928 us
	-16	q.:ecs2:main+0018	lis	r12 0000	-2.380 us
	-12	q.:ecs2:main+001C	lwz	r3 41B0(r12)	-1.756 us
G1.	-8	e/g.elf:.bss+01B0		read 00xxxxxx	-1,172 us
	-7	e/q.elf:.bss+01B1		read 05	-1.052 us
	-6	e/q.elf:.bss+01B2		read 7Exx	-936.000 n:
	-5	e/q.elf:.bss+01B3		read 05	-820,000 n:
	-4	q.:ecs2:main+0020	bl	update:update_system	-548.000 n:
tr	0	upd:update_system	mfspr	r0 lr	0 s
	4	update_syste+0004	mr	r11 r1	624.000 ns
	8	update_syste+0008	stwu	r1 FFE8(r1)	1.248 us
	12	update_syste+000C	bl	rce/q.elf:.text+4AOC	2.028 us
	16	/q.elf:.text+4A0C	stw	r29 FFF4(r11)	2.576 us
	20	/q.elf:.text+4A10	stw	r30 FFF8(r11)	3.356 us
	24	/q.elf:.text+4A14	stw	r31 FFFC(r11)	4.136 us
	28	/q.elf:.text+4A18	stw	r0 0004(r11)	4.916 us
	32	/q.elf:.text+4A1C	blr		5,696 us



Each analyzer triggers according to its own setup. You can change this by setting up one analyzer to be armed by another analyzer.

**See Also** "System Integration" on page 257 for information on coordinating the collection of data with a group run or arming a measurement in one machine by a trigger in the other.

### To capture execution up to a failure or halt



Possible uses:

• To store and display all activity leading up to a system crash.

• To run the logic analyzer indefinitely until the Stop button is selected so that you can observe system activity at your discretion.

Probing the Target System 1. Configure a state analysis machine.



Sampling Forma	at   Trigger   Symbol	]	
Analyzer Name:	Analyzer <e></e>	🖉 0n 📿 GHz 📴 🖉	Timing Zoom
-	Asynchronous sampling Synchronous sampling		

2. Select the state analyzer's clock input.

State Mode Controls								
167 MHz / 2M State 🛃 Trigger Position Center								
Acquisition depth 2M								
Clock Set	up—							
Mode: Master only 👤 🔲 Advanced Clocking								
Pod	E4	E3	E2	E1				
Pod Clock	E4 M	E3 L	E2 K	E1 J				
				E1 J t				
Clock	M -	L ‡	К ‡	E1 J t => [(J†)+(K=0)]				

3. Format labels for the signals on which you will look for the event.

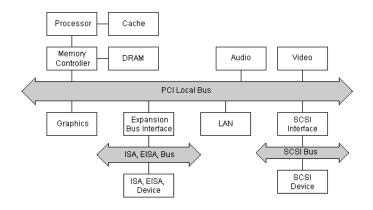
Sampling Format Trigger Symbol							
Pod Assignment	1	Cloc L. DF	ks F	Pod D2	Pod D1	Pod E2	
	<u> </u>			TTL	TTL	TTL	
Setup/Hold	•	<b>‡‡</b> КЈ	‡‡ KJ	<b>**********</b>		<b>***********</b> _*0	‡ 1
ADDR	+	••	••			*****	*
DATA	+		••	*****	****		
STAT	+	**	**	*	*****		ŀ
data[0-7]	+	••	•••	******			ŀ

**Capturing the Data** 1. Use the "Run until user stop" trigger function.

Sampling Format Trigger	Symbol ]				
Trigger Functions Settings Ove	erview Default	Storing	Status Sa	ave/Recal	1
General State			Trigger f	unction	libraries
Find pattern n times Store range until pattern occur Store pattern2 until patterni oc Store nothing until pattern occ Run until user stop	occurs	 ∧ Run			STOP
Replace Insert	before	Insert	after	}	Oelete
Trigger definition					
1 RUN UNTIL USER STOP					

- 2. Select the Run button to start the measurement.
- 3. When the system fails, crashes, or halts, select the Stop button to see the states that were captured before the failure.
- **Displaying the Data** 1. Use the Listing display to show the states that led up to the failure.

### To view bus activity



Target system buses are good locations to view system activity and may be the first place you look when isolating problems (especially in multiprocessor systems).

#### Requirements:

• This measurement requires an analysis probe for the standard bus you wish to view.

#### Possible uses:

- To isolate system problems.
- To view and correlate activity on multiple buses.
- To view and correlate bus activity to microprocessor execution.

## Probing the Target<br/>System1. Use a standard bus analysis probe to make the physical connection<br/>between the logic analyzer and the bus.

2. Use the configuration files included with the analysis probe to configure the analyzer and format labels.

## **Capturing the Data** 1. Set up a trigger specification using the labels defined by the analysis probe configuration files.

## **Displaying the Data** 1. Use the Listing display to view captured bus activity. If the standard bus analysis probe provides an inverse assembler, you will see mnemonics for

bus commands, status, etc.

See Also

"To analyze bus stability (with SPA)" on page 88

"To analyze bus occupation & bandwidth (with SPA)" on page 131

"To simulate bus occupation and measure SW performance" on page 287

### To capture simple program messages

Disp	played File: /logic/demo/860_demo_board/source/ecs2.c
98 99 100 101 102 103 104	char ME_update_system; Create memory
Disp	played File: /logic/demo/860_demo_board/source/update_sys.c
55 56 57	void update_system(int passes) {
58 59 60 61 62 63	ME_update_system = 1; Add code that /* get new targets */ get_targets(⌖_temp); (blocations
64 65 66 67 68 69 70 71 72 73	<pre>/* Read the environment conditions. */ read_conditions(passes, &amp;current_temp); /* Set the func_needed based on the actual environment condition versus the desired environment condition. */ set_outputs(%func_needed, current_temp); /* Update the howr_encode value so the external devices can react to modify the environment.*/ write_howr(runc-needed, howr_encode);</pre>
74 75 76 77 78 79	<pre>/* Save the current temp for later processing */ save_points(); /* MX_update_system = 1; </pre>

By adding program code that causes activity external to the processor (also known as "instrumenting your code"), you can create specific program message events that can be captured by the analyzer.

Possible uses:

- To view processor execution when the instruction cache is turned ON.
- To view higher-level program activity (like Real-Time OS function calls or OS calls).

# Probing the Target<br/>System1. Typically, you will use an analysis probe to connect the logic analyzer to<br/>the microprocessor, and you will use the provided configuration files to<br/>configure the analyzer and define labels.

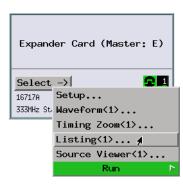
Expander Card (Master: E)	MHz
Expanden Cand (Masters E) File Window	
Load Configuration.4.	3
Save Configuration	<u> </u>
Select -> Print options Trigg	er
16717A Setup A Print this window	-
333MHz Sta Waveform<1> Print any window	E>
Timing Zoom(1) Close chron	ous
Listing(1)	
Source Viewer(1)	us s
Run 🕞 State Mode Controls	

**Capturing the Data** 1. Set up a trigger specification and use storage qualifiers that capture the program messages you have coded into your programs.

Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing Status Save/Recall
Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing".
Store by default Custom
At start of acquisition, On =
Group events
Trigger definition
1 FIND PATTERN N TIMES
Find 1 occurrence of
ADDR = board/source/q.elf:update_system Symbols
then Trigger and fill memory

2. Select the Run button to start the measurement.

**Displaying the Data** 1. Use the Listing display to view the captured program messages.



	State Number	PC	MPC821/860 Inverse Assembler	Time
	Decimal	Symbols	10=hex, 10.=decimal, %10=binary	Relative
tr	0	upd:update_system	Undefined Opcode 7C010101	5,764
	1	:ME_update_system	write 01	8.692 (
	2	q.:ME_get_targets	write 01	3.892
	3	q.:MX_get_targets	write 01	37.008
	4	ME_read_condition	write 01	5.064
	5	MX_read_condition	write 01	194.556
	6	q.:ME_set_outputs	write 01	6.556
	7	q.:MX_set_outputs	write 01	1.070 r
	8	q.e:ME_write_hdwr	write 01	15.188
	9	q.e:MX_write_hdwr	write 01	1.057 r
	10	q.:ME_save_points	write 01	9.532
	11	q.:MX_save_points	write 01	366.816
	12	:MX_update_system	write 01	2.652
	13	ME_update_display	write 01	22.508
	14	MX_update_display	write 01	118.040
	15	:ME_proc_specific	write 01	11.704
	16	:MX_proc_specific	write 01	4,532 (

#### To trigger on packet data (with DataComm Analysis)

**Requirements:** 

- The DataComm Analysis tool set.
- An Agilent Technologies 16715/16/17/18/19A logic analyzer module (and its VisiTrigger capabilities).

Possible uses:

• To look at data traveling across parallel buses inside network switching systems.

Probing the Target System

- 1. Connect logic analyzer probe channels to:
  - A communication data bus.
  - Control signals that identify the start of packet, the packet data, and the end of packet.
  - 2. You also need to connect a logic analyzer CLK input channel to:
    - A clock signal that identifies when the data bus and control signals are valid and should be sampled by the logic analyzer.
  - 3. Configure a (synchronous sampling) state analysis machine.

	Expander Card (Master: E)
	Select ->
I	16717A Setup 4
	333MHz Sta Waveform<1>
	Timing Zoom<1>
	Listing(1)
	Source Viewer<1>
	Run 🖻
ì	
	Sampling Format Trigger Symbol
	Analyzer Name: Analyzer(E)
	$\diamondsuit$ Timing Mode – Asynchronous sampling clocked internally by analyzer
	ightarrow State Mode – Synchronous sampling clocked by the Device Under Test

4. Select the state analyzer's clock input.

—State Mode	Cont	rols					
167 MHz / 2M State 🛓 Trigger Position Center 🛓							
Acquisition depth 2M							
_Clock Set	Clock Setup						
Mode: Ma	ster	only	<u>ا</u> ا		Advanced Clocking		
Pod	E4	E3	E2	E1			
Clock	М	L	К	J			
Activity	-	‡	\$	\$			
Master	Off	Off	0	₹	=> [(J†) • (K=0)]		

5. Format a DATA label for the logic analyzer channels that are probing the data bus. Format 1-bit labels for the channels that are probing the start of packet, packet data, and end of packet signals.

Sampling	For	mat   Trigger   S	ymbol					
Pod	1	Data On Clocks AA	Pod A2			Pod A1		
Assignment.				TTL			TTL	
Setup/Hold.	••	κī	ī	87		15	87	
DATA	+		••••		••••	****	******	****
SOP	+		••••		••••*	••••		
EOP	+			• • • • • • • • •	*.			

**Capturing the Data** 1. Select the "Find Packet" trigger function.

Sampling Format Trigger Symbol	]
Trigger Functions Settings Overview D	efault Storing Status Save/Recall
General State, Telecom State	Trigger function libraries.
OR Trigger Find Packet	SRC ADDR SRC ADDR SRC ADDR
Advanced - If/then Advanced - 2-way branch	DST ADDR
Replace Insert before	Insert after Delete

- 2. In the "Find Packet" trigger sequence level, select the bus button.
- 3. In the Bus Selector dialog, select the bus definition you want to use and select the OK button.

Trigger Sequence				
1 FIND PACKET				
On bus No Bus Selected!				
If Any Packet occurs once				
then Trigger and fill memory Help				

4. Specify the packet event to find.

Trigger Sequence
1 FIND PACKET
On bus Bus #1
If Any Packet occurs once
then Tr Edit Events Help Help
Event #1

- 5. Select the Run button to start the measurement.
- **Displaying the Data** 1. In the logic analyzer's Listing window, load the INETWRKE network protocol decoder inverse assembler.

Listing<1>
File Window Edit Options Invasm Source
Load Invasm – Listing<1>
<pre>/logic/ia/IAPIC2 /logic/ia/IAPIC4 /logic/ia/IARM7E /logic/ia/IARM7M_E /logic/ia/ICPU32B /logic/ia/ICPU32BE /logic/ia/INETWRKE /logic/ia/IPSTRIKE</pre>
Frame 10:Slot A:Analyzer(A)_TZ
OK Cancel About Invasm Enable Auto Lo

2. View the captured data.

Listing<1>						
File Window Edit Options Invasm Source						
		_				
Goto Markers Search Comments Analysis Mixed Sig	nal					
Trigger Beginning End G1 G2						
Trigger Beginning End G1 G2						
Goto Time 🛓 🖉 s 上 Goto						
State Number Network Protocol Decoder	DATA	STAT				
Decimal	Hex	Hex				
-50 IEEE 802.3 (Ethernet V2)	0100	1				
Dest Addr = 01-00-5e-00-00-05 Src Addr = 00-10-21-47-20-0d						
Src Addr = 00-10-21-47-20-0d						
Length/Type = 0800 Hex (Internet Protocol)						
Internet Protocol Version = 4 Hex						
Header Length = 5 Decimal						
Precedence = 2 Hex (Immediate)						
Delay = 0 Binary (Normal)						
Throughput = 0 Binary (Normal)						
Reliability = 0 Binary (Normal)						
Cost = 0 Binary (Normal)						
MBZ = 0 Binary						
Total Length = 124 Decimal						
Identification = c2e4 Hex						
Zero = 0 Binary						
Do not fragment = 0 Binary						
May Fragment = 0 Binary Fragment Offset = 0 Decimal						
Fragment Offset = O Decimal Time To Live = 1 Decimal						
Protocol = 59 Hex						
Header Checksum = 066b Hex						
Src Addr = 15.19.0.2						
Dest Addr = 96.0.0.5						
-49 Packet header data	5E00					
-48 Packet header data	0005	0				
-47 Packet header data	0010	0				
-46 Packet header data	2147	0				
-45 Packet header data	200D	0				
-44 Packet header data	0800	0				
-43 Packet header data -42 Packet header data	4540	0				
−42 Packet header data −41 Packet header data	007C C2E4	0 0				
-41 Packet header data	0000	ŏ				
-39 Packet header data	0159	ŏ				
-38 Packet header data	066B	õ				
-37 Packet header data	0F13	0				
-36 Packet header data	0002	0				
-35 Packet header data	6000	0				
-34 Packet header data	0005	0				

See Also

Using the DataComm Analysis Toolset (see the *DataComm Analysis* 

*Toolset* help volume)

# Making Interrupt Service Routine Measurements

- "To capture interrupt frequency and type" on page 183
- "To measure interrupt latency and execution time" on page 186
- "To simulate particular interrupt sequences" on page 191
- "To view the occurrence rate of an event (with SPA)" on page 192

## To capture interrupt frequency and type



Possible uses:

• To analyze interrupt processing.

#### Probing the Target System

 Typically, you will use an analysis probe to connect the logic analyzer to the microprocessor, and you will use the provided configuration files to configure the analyzer and define labels.

Analyzer <e></e>	- 333MHz
Expander Card (Master: E) File Window	
Load Configuration.4.	
Save Configuration	
Select -> Print options	Trigger
16717A Setup A Print this window	(5)
333MHz Sta Waveform<1> Print any window	yzer <e></e>
Timing Zoom(1) Close	chronous
Listing<1>	haanaya a
Source Viewer(1)	nronous s
Run 🕞 State Mode Controls	

# Capturing the Data1. The trigger specification will depend on the interrupt mechanism of your<br/>microprocessor.

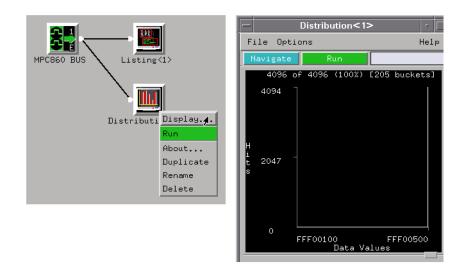
If your processor uses an interrupt vector table, set up a trigger specification that only stores accesses to the interrupt vector table locations.

Sampling Format	Trigger   Symbol		
Trigger Functions Set	tings Overview Defau	lt Storing Status Sa	ve/Recall]
General State		Trigger fu	unction libraries
Find pattern n times Store range until pat Store pattern2 until p Store nothing until p Run until user stop	oattern1 occurs	Occurrence 1 pattern	Occurrence n pattern
Replace 📐	Insert before	Insert after	Delete

Trigger definition
1 FIND PATTERN N TIMES
Find 1
ADDR = 111111111110000000XXXXX00000000 Binary
then Trigger and fill memory

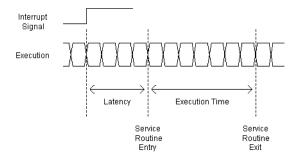
	Sampling Format Trigger Symbol	1
	Trigger Functions Settings Overview D	efault Storing Status Save/Recall
Specify what to store by default.		Store if -
	Using store actions in a sequence level overrides the "Default Storing".	ADDR = 111111111110000000XXXXX000000
	Store by default Custom 😑	
	At start of acquisition, On =	
		Group events

- 2. Select the Run button to start the measurement.
- **Displaying the Data** 1. In the Workspace window, add the Distribution display to view the captured data.



If your trigger specification stored interrupt vector table accesses, the captured table locations will show the interrupt types.

## To measure interrupt latency and execution time



## Possible uses:

• To see if interrupt processing meets specifications.

#### Probing the Target System

1. Configure a state analysis machine.

Expander Card (Master: E)					
Select ->					
16717A Setup	1				
333MHz Sta Waveform<1>	1				
Timing Zoom<1>					
Listing<1>					
Source Viewer<1>					
Run 🕨					
Sampling Format Trigger	r   Symbol	1			
Analyzer Name: Analyzer(E	>	🖉 On 🗾 🛛 🖉 GHz	lin ind Zoom	Timing	Zoor
♦ Timing Mode - Asynchronou	us sampling	clocked inte	rnally b	y analy	zer
<ul> <li>State Mode – Synchronous</li> </ul>			-		
• State House Sgriefil offoda	5 Scalpring	crocked by th	0 004100	, onder	1030

2. Select the state analyzer's clock input.

—State Mode	Contro	ls								
167 MHz /	2M Sta	te 🛓		Trigge	r Po	osition	n Cen	ter	_	Ŧ
Acquisitio	n depti	n 2M	Ŧ							
_Clock Set	up									
Mode: Ma	ster or	nly 🤳		Advanc	ced	Clocki	ng			
Pod	E4 E	3 E2	E1							
Clock	ML	. К	J							
Activity	_ ‡	\$	\$							
Master	Off Of	f 0	<b>_</b> =	=> E(J,	†) • (	(K=0)]				

3. Assign pods if necessary.

Sampling Format Pod Assignment.			
-		Pod Assignment	
Analyzer 1		Analyzer 2	7
Name: Analyzer <e></e>	Name:	Analyzer <e2></e2>	
Type: State =	Type:	Off -	Unassigned Pods
<b>E1: ************</b> J <b>*</b>			D3:L _
E2: ++++++++++++++++++++++++++++++++++++			D4:M _
E3:+++++++++++++++++++++++++++++++++			
E4: M _			
<b>D1:</b> _####################################			
<b>D2: ********</b>			

4. Format labels for the signals on which you will look for the event.

Sampling Fo	rmat   Tr	igger   Symbol			
Pod Assignment	Clocks DFF	Pod D2	Pod D1	Pod E4	
		TTL	TTL	TTL	
Setup/Hold	‡‡_‡‡‡ KJMLKJ	<b>‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡‡</b>	ī5 <sup></sup> 87 <sup></sup> 0	15
ADDR +					•••
DATA +		*****	****		•••
STAT +	****	*	****		•••
data[0-7] +		******			
*IRQ3 +					

**Capturing the Data** 1. Set up a trigger specification that stores the state when the interrupt signal becomes active, the interrupt service routine entry point, and the interrupt routine exit point.

Sampling   Format   Trigg	ger Symbol	
Trigger Functions Settings	s Overview Default Stori	ng Status
General State		
Wait for arm in Advanced - If/then Advanced - 2-way branch Advanced - 3-way branch Advanced - 4-way branch		
Replace	Insert before	Ir
Trigger definition		
1 If *IRQ3 = 0 Binary occurs 1 time then Store sample Trigger and goto Nex	t	
2 If ADDR = board/source/o occurs 1 time then Store sample Goto Next	q.elf:ext_exception Symbols	
3 If ADDR = source	/q.elf:ext_exception+0x2F7	Symbols
occurs 1	time	
then Store sample		
Goto 1		
Sampling   Format   Trigg	ger Symbol	
Trigger Functions Settings	s Overview Default Stori	ng Status
Specify what to store by d Using store actions in a s level overrides the "Defau	equence	
Store by default Nothing		
At start of acquisition, "Default Storing" is	0n =	

- 2. Make sure the analyzer is counting time.
- 3. Select the Run button to start the measurement.
- **Displaying the Data** 1. Use the Listing display to view the captured states.

# Chapter 1: Measurement Examples Firmware Development

Expand	ler Card (Master: E)
Select	->
16717A	Setup
333MHz St.	Waveform<1>
	Timing Zoom<1>
	Listing<1> 🖌
	Source Viewer(1)
	Run 🕨

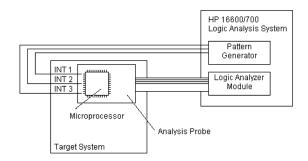
	State Number	*IRQ3	ADDR	Time
	Decimal	Binary	Symbols	Relative
tr	0	0	lcd_write_st+0020	
	1	1	isr:ext_exception	14.320 us
	2	1	ext_exceptio+02F7	51,276 us
G1.	3	0	<pre>:set_outputs+000F</pre>	2,294 ms
	4	1	isr:ext_exception	11.634 ms
	5	1	ext_exceptio+02F7	25.444 us
<u>62.</u>	6	0	lc:lcd_ready+0008	2,319 ms
	7	1	isr:ext_exception	14.324 us
	8	1	ext_exceptio+02F7	51.280 us
	9	0	<pre>:set_outputs+00ED</pre>	2.399 ms
	10	1	isr:ext_exception	11.579 ms
	11	1	ext_exceptio+02F7	25.456 us
	12		lc:lcd_ready+0004	2.423 ms
	13	1	isr:ext_exception	14.316 us
	14	1	ext_exceptio+02F7	51.256 us
	15		<pre>:set_outputs+00C4</pre>	2.507 ms
	16	1	isr:ext_exception	11.457 ms
	17	1	ext_exceptio+02F7	25.456 us

The relative time between the state where the interrupt signal became active and the service routine entry shows the interrupt latency.

The relative time between the service routine entry and exit shows the interrupt execution time.

See Also

"To measure function execution time (with SPA)" on page 218



# To simulate particular interrupt sequences

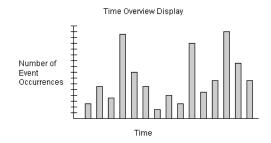
#### **Requirements**:

• This measurement requires a pattern generator module (Agilent Technologies 16522A).

Possible uses:

- To test the processing of multiple interrupts.
- Probing the Target System
  1. Connect pattern generator outputs to microprocessor interrupt inputs.
  2. Configure the pattern generator to output the desired sequence of interrupt signals.
  3. Typically, you will use an analysis probe to connect the logic analyzer to the microprocessor, and you will use the provided configuration files to configure the analyzer and define labels.
  Capturing the Data
  Displaying the Data
  Use the Listing display to view the captured interrupt processing.
  See Also
  "To generate pattern stimulus on devices" on page 75
  "To generate patterns when a source line executes" on page 262

## To view the occurrence rate of an event (with SPA)



#### Requirements:

• This measurement requires the system performance analyzer (SPA) tool set.

#### Possible uses:

- To display interrupt loading.
- To measure the frequency at which data is acquired from sensors.

#### **Probing the Target** System

- 1. Connect the logic analyzer to the target system signals on which you will look for the event. You can use an analysis probe to connect the logic analyzer to a microprocessor or standard bus.
  - 2. Configure a state analysis machine. (If you are using an analysis probe, use the provided configuration files to configure the analyzer and define labels.)

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing(1)
Source Viewer<1>
Run 🖻
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E)
$\diamond$ Timing Mode – Asynchronous sampling clocked internally by analyzer
$\clubsuit$ State Mode - Synchronous sampling clocked by the Device Under Test

3. Select the state analyzer's clock input.

167 MHz / 2M State     ▲       Acquisition depth     2M       Acquisition depth     2M       Clock Setup     ▲       Mode:     Master only       ▲     ▲       Clock M     L       K     J       Activity     -       ▲     ↓       Master     Off       0ff     0	State Mode Controls					
Clock Setup Mode: Master only Pod E4 E3 E2 E1 Clock M L K J Activity - Clock + Clock + Cl	167 MHz /	2M S	tate	<u> </u>		Trigger Position Center 👤
Mode: Master only Pod E4 E3 E2 E1 Clock M L K J Activity - Clock A	Acquisition depth 2M					
Pod     E4     E3     E2     E1       Clock     M     L     K     J       Activity     _     ‡     ‡	—Clock Set	up—				
Clock M L K J Activity _ <b>‡ ‡</b>	Mode: Ma	ster	only	<u>ا</u> ا		Advanced Clocking
Activity _ + + +	Pod	E4	E3	E2	E1	
	Clock	М	L	К	J	
Master Off Off 0 $\mathbf{F}$ => $\Gamma(1^{\dagger}) \cdot (K=0)1$	Activity	-	\$	\$	\$	
	Master	Off	Off	0	F	=> [(J†) • (K=0)]

4. Assign pods if necessary.

# Chapter 1: Measurement Examples Firmware Development

Sampling Format

Pod Data			
		Pod Assignment	
Analyzer 1		Analyzer 2	7
Name: Analyzer <e></e>	Name:	Analyzer <e2></e2>	
Type: State 🗆	Type:	Off -	Unassigned Pods
E1: ++++++++++++++++++++++++++++++++++++			D3: L _
<b>Е2:</b> ‡‡‡‡‡‡‡‡‡‡‡ к ‡			D4: M _
E3:+++++++++++++++++++++++++++++++++			
E4: M _			
D1: _ # # # # # # # # # # # # # # # # # #			
<b>D2: #########</b> ##########################			

5. Format labels for the signals on which you will look for the event.

Sampling Format Trigger Symbol						
Pod Assignment	Clocks	Pod D2	Pod D1	Pod E4		
	1	TTL	TTL	TTL		
Setup/Hold	1 <b>‡‡_‡‡‡</b> KJMLKJ	<b>‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡‡‡</b>	15 87 0	15	
ADDR +	·	•••••		•••••		
DATA +	·	*****	*****			
STAT +	****	****	****			
data[0-7] +	·	******				
*IRQ3 +		•••••		•••••		

## **Capturing the Data**

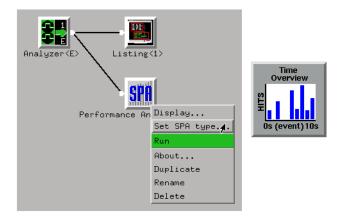
1. Set up the logic analyzer trigger specification to capture the events you're interested in.

St

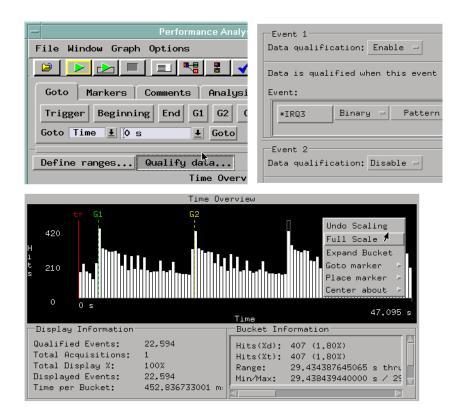
Sampling   Format	Trigger	Symbol )	
Trigger Functions Se	ttings Ov	erview Defa	iu.
General State			
Find pattern n times Store range until pat Store pattern2 until Store nothing until p Run until user stop	pattern1	occurs	
Replace	Inser	t before	
Trigger definition			
1 FIND PATTERN N TIMES Find 1 occurrence of *IRQ3 = 0 Binary then Store sample Trigger and go	ĥ		
	6 ccurrence o inary	f	
Sampling   Format	Trigger	Symbol	
Trigger Functions Se	ttings Ov	erview Defa	ault Stor
Specify what to store Using store actions i level overrides the ' Store by default <u>No</u> At start of acquisit	in a seque 'Default S thing =	nce toring".	

- 2. Select the Run button to start the measurement.
- **Displaying the Data** 1. Use the system performance analyzer's Time Overview display to show the results of the measurement.

# Chapter 1: Measurement Examples Firmware Development



2. Define the event whose occurrence rate you wish to measure.



You can define both the event and the time period in which the events are counted.

# Software Development

Analyzing Real-Time Software Execution

- "To trace about a source line" on page 199
- "To trace function flow" on page 203
- "To trace callers of a function" on page 206
- "To trace execution within a function" on page 210
- "To measure function execution time" on page 214
- "To measure function execution time (with SPA)" on page 218
- "To omit monitor cycles from the trace" on page 223
- "To stop execution at a source line (in ROM)" on page 226

Analyzing Real-Time Variable Access

- "To find NULL pointer de-references" on page 229
- "To trace a variable's values" on page 231
- "To find where variables are accessed from" on page 236
- "To trace before a variable value" on page 240
- "To stop execution on a corrupt variable" on page 245

Analyzing Real-Time Memory Usage

- "To monitor stack or heap usage" on page 251
- "To find stack overflow or guarded memory access" on page 255

# Analyzing Real-Time Software Execution

- "To trace about a source line" on page 199
- "To trace function flow" on page 203
- "To trace callers of a function" on page 206

- "To trace execution within a function" on page 210
- "To measure function execution time" on page 214
- "To measure function execution time (with SPA)" on page 218
- "To omit monitor cycles from the trace" on page 223
- "To stop execution at a source line (in ROM)" on page 226

## To trace about a source line

140 141 142 143 144 145 146 147 148 149	<pre>main() {     boot_q();     init_system();     proc_spec_init();     for (;;)     {         update_system(num_checks);     } }</pre>
150 151	num_checks++; line # 149
152 153	proc_specific(); Trace before this line
153	3 Trace about this line
155	Trace after this line 🖌
156 157	* FUNCTION: update_display Goto this line in listing before current state
158 159	* PARMS: counter loop count * DESCRIPTION:

#### **Requirements:**

• This measurement requires the source correlation tool set product. When this product is installed, you can view high-level source files in a special listing window.

Possible uses:

- To quickly capture and view execution around a particular high-level source line.
- Probing the Target1.Use an analysis probe to connect the logic analyzer to the microprocessor,<br/>and use the provided configuration files to configure the analyzer and<br/>define labels.

			- Analyzer <e></e>	▶ - 333MHz
Expande	er Card (Master: E)		File Window	
•		ľ	Load Configuration	
		-	Save Configuration	
Select			Print options	Trigger
16717A	Setup 🖌		Print this window	(5)
333MHz Sta	Waveform<1>		Print any window	yzer <e></e>
	Timing Zoom<1>		Close	chronous
	Listing<1>		A Chata Mada Coma	
	Source Viewer<1>		🔷 State Mode – Sync	nronous s
	Run 1		-State Mode Controls-	

**Capturing the Data** 1. Download symbols from your target system program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	

2. Open the Source Viewer window.

Expander Card (Master: E)					
Select					
16717A	Setup				
333MHz Sta	Waveform<1>				
	Timing Zoom<1>				
	Listing<1>				
	Source Viewer<1>				
	Run 🕨				

- 3. Browse the source file that contains the line you want to trigger on.
- 4. Select the line you want to trigger on and choose the "Trigger after this line" menu item.

Ste	ep Source   Goto In List	ting Browse Source Text Search Symbols
Ne	ew Source File Name	
e	cs2.c	File Selection
Disp	layed File: /logic/demo	/860_demo_board/source/ecs2.c
140	main()	
141	(	
142	boot_q();	
143 144	init	
144	init_system(); proc_spec_init();	
146	proc_spec_init();	
147	for (;;)	
148	(	
149	update_system(r	
150	num_checks++;	line # 149
151 152	update_display proc_specific(	Trace before this line
153	}	Trace about this line
154 155	}	Trace after this line 🖌
155	/*****	· · · · · · · · · · · · · · · · · · ·
157	* FUNCTION: update_di	Goto this line in listing before current sta
158	* PARMS: counter -	Goto this line in listing after current stat
159	* DESCRIPTION:	
160	* clear out the hist	tory buffer and update the current ascii disp

#### NOTE:

Source Viewer commands that set up triggers only modify the trigger condition. They do not modify the trigger position, storage qualifiers, else branch conditions, or other levels in the trigger sequence.

# Chapter 1: Measurement Examples **Software Development**

- 5. Select the Run button to start the measurement.
- **Displaying the Data** 1. Open the Listing window to display the captured data. You may want to load an inverse assembler and display symbols in the address label column.

Expander Card (Master: E)					
Select					
16717A	Setup				
333MHz St	Waveform<1>				
	Timing Zoom<1>				
	Listing<1> 🖌				
	Source Viewer<1>				
	Run 🕨				

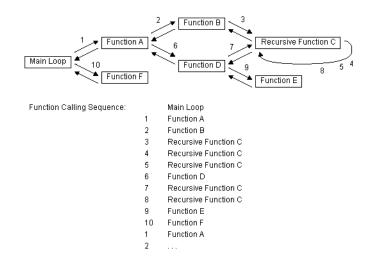
	State Number	PC	MPC821/86	0 Inverse Assembler	ADDR
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Hex
	•				
tr	0	q.:ecs2:main+0018	lis	r12 0000	FFF0317C
	4	q.:ecs2:main+001C	lwz	r3 41B0(r12)	FFF03180
	8	e/q.elf:.bss+01B0		read 00xxxxxx	000041B0
	9	e∕q.elf:.bss+01B1		read 00	000041B1
	10	e/q.elf:.bss+01B2		read 69xx	000041B2
	11	e∕q.elf:.bss+01B3		read 84	000041B3
G1.	12	g.:ecs2:main+0020	bl	update:update_system	FFF03184
	16	upd:update_system	mfspr	r0 lr	FFF034D8
	20	update_syste+0004	mr	r11 r1	FFF034DC
	24	update_syste+0008	stwu	r1 FFE8(r1)	FFF034E0
	28	update_syste+000C	bl	rce/q.elf:.text+4AOC	FFF034E4
	32	/q.elf:.text+4A0C	stw	r29 FFF4(r11)	FFF06A0C
	36	/q.elf:.text+4A10	stw	r30 FFF8(r11)	FFF06A10
	40	/q.elf:.text+4A14	stw	r31 FFFC(r11)	FFF06A14
	44	/q.elf:.text+4A18	stw	r0 0004(r11)	FFF06A18
G2.	48	/g.elf:.text+4A1C	blr		FFF06A1C

2. You can use the Step Source Previous and Next buttons in the Source Viewer window to browse the captured data by associated source lines.

See Also

"To stop execution at a source line (in ROM)" on page 226 "To generate patterns when a source line executes" on page 262 "To trigger an oscilloscope when a source line executes" on page 294

# To trace function flow

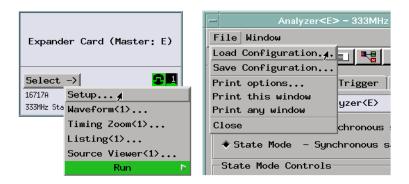


Possible uses:

• To view the execution order of routines.

#### Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



2. Load symbols from your program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	

**Capturing the Data** 1. Set up a trigger specification that stores only the type of execution that occurs when a function is entered.

For example, in the Motorola 68XXX microprocessors, the LINK instruction is commonly used on function entry to set up a new stack frame. In the PowerPC microprocessors, the "mfspr r0,lr" instruction is commonly used at function entry.

Another way to identify function entry points is to add statements at the beginning of functions that write to particular memory locations (also known as instrumenting your code). This is the best way to identify function entry points when instruction caches are turned ON.

Trigger definition
1 If ADDR = board/source/q.elf;update_system Symbols occurs 1 time
then Trigger and goto Next Specifies when to start capturing function flow.
2 If DATA = 7CXXXXX Hex occurs1 time then Goto Next Else if DATA ≠ 7CXXXXX Hex then Goto 2
3       If DATA = 08XXXXXX Hex occurs 1 time then Goto Next Else if DATA ≠ 08XXXXX Hex then Goto 2         Capture 32-bit instruction fetch
4 IF DATA = 02XXXXX Hex occurs 1 time then Goto Next 4 B-bit data bus (4 consecutive re of the instruction opcode).
Else if DATA ≠ 02XXXXX Hex then Goto 2
5       If DATA = A6XXXXX Hex occurs1 time then Goto Next Else if DATA ≠ A6XXXXX Hex then Goto 2
6 IF ADDR = XXXXXXXX Hex
occurs 1 dime
then Store sample Store state after instruction fetch. The address value will identify the function.
Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing St
Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing".
Store by default Nothing =
At start of acquisition, On = "Default Storing" is

2. Select the Run button to start the measurement.

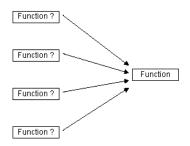
# **Displaying the Data** 1. Open the Listing display to view the captured execution. By viewing the symbolic information associated with the captured states, you will see the function execution sequence.

# Chapter 1: Measurement Examples Software Development

Expander Card (Master: E)				
Select				
16717A	Setup			
333MHz St	Waveform<1>			
	Timing Zoom<1>			
Listing(1) 🖌				
Source Viewer<1>				
Run 🕅				

	State Number	PC	MPC821/860 Inverse Assembler	ADDR
	Decimal	Symbols	10=hex, 10.=decimal, %10=binary	Hex
	0	upd:update system	Undefined Opcode 7C7C7C90	FFF034D8
ur_	4			FFF034D0
		u:write_hdwr+0004	Undefined Opcode 7C7C9090	
	2	update_displ+0004	Undefined Opcode 7C909090	FFF031B8
	3	proc_specifi+0004	stw r4 907C(r16)	FFF040D0
	4	ext_exceptio+0004	stw r4 7C7C(r16)	FFF02C78
	5	ccsp:sprintf+0004	stw r3 7C7C(r28)	FFF04BA8
	6	ccs:vsprintf+0004	Undefined Opcode 7C7C7C90	FFF04C20
	7	ccmem:memset+0004	Undefined Opcode 7C7C9090	FFF047A0
	8	ccv:vfprintf+0004	Undefined Opcode 7C909090	FFF0583C
	9	ccfput:fputc+0004	stw r4 9090(r16)	FFF06764
	10	ccfput:fputc+0004	stw r4 9090(r16)	FFF06764
	11	ccfput:fputc+0004	stw r4 9090(r16)	FFF06764

# To trace callers of a function



Possible uses:

- To show the callers of a particular function.
- To find out from where an exception or task call originates.
- Probing the Target1.Use an analysis probe to connect the logic analyzer to the microprocessor,<br/>and use the provided configuration files to configure the analyzer and<br/>define labels.

	Analyzer <e> - 333MHz</e>
Expander Card (Master: E)	File Window
	Load Configuration.4.
	Save Configuration
Select ->	Print options Trigger
16717A Setup	Print this window
333MHz Sta Waveform<1>	Print any window yzer(E)
Timing Zoom<1>	Close chronous
Listing(1)	State Mode - Synchronous s
Source Viewer<1>	- Juice node - Juice -
Run	State Mode Controls

2. Load symbols from your program's object module file.

Sampling Format Trigger Symbol					
Object File User Defined					
Load This Object/Symbol File For Label: ADDR					
//logic/demo/860_demo_board/source/q.elf Browse					
Create Symbol File (.ns) In This Directory:					
/logic/symbols/ Browse					
Object Files with Symbols Loaded For Label: ADDR					
/logic/demo/860_demo_board/source/q.elf					

Capturing the Data1. Set up a trace that captures and stores only entry into a particular function<br/>and uses context store to store the states that occurred before entry into<br/>the function.

If your logic analyzer doesn't have the context store feature, you can set

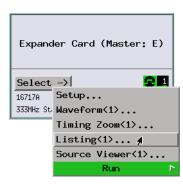
up a trigger sequence that stores a function's exit and where the execution returns to (which should identify the calling function).

Sampling Format Trigger Symbol	01
Trigger Functions Settings Overview	Default Storing Status Save/Recall
General State	Trigger function libraries
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop Replace Insert befo	pattern pattern
	reInsert afterDelete
Trigger definition          1       FIND PATTERN N TIMES         Find 1       occurrence of         ADDR = _board/source/q.elf:sprintf*(         then Trigger and goto Next         2       FIND PATTERN N TIMES         Find 1	Dx73 Symbols
Sampling Format Trigger Symb	
Trigger Functions Settings Overview	Default Storing Status Save/Recall
Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing	
Store by default Custom =	
At start of acquisition, On = "Default Storing" is	Group events

2. Select the Run button to start the measurement.

**Displaying the Data** 

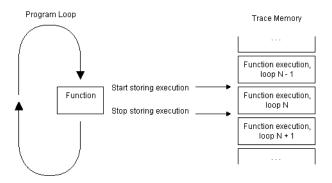
1. Open the Listing window to view the captured execution. Include symbols in the listing.



	State Number	PC	MPC821/8	60 Inverse Assembler	Time
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Relative
tr	0			pgm 20	
	1	proc_specifi+00A4	li	r1 00003820	196.000
	2			pgm 20	1.986 r
	3	proc_specifi+0108	li	r1 00003D20	196.000 (
	4			pgm 20	9.718 r
	5	update_displ+0154	lis	r9 3C20	196.000 (
	6			pgm 20	901.788
	7	update_displ+017C	lis	r1 3D20	196.000
	8			pgm 20	901.740
	9	update_displ+01A4	lis	r9 3D20	192.000
	10			pgm 20	901.896
	11	update_displ+01C8	lis	r9 4820	196.000
	12			pgm 20	679.512 (
	13	proc_specifi+0090	b	0010797C	192.000
	14			pgm 20	1.812 r
	15	proc_specifi+0108	li	r1 00004820	196.000 (
	16			pgm 20	9,073 r

You can also open the Source Viewer window and use the Step Source Previous and Next buttons to browse the captured data by associated source lines.

# To trace execution within a function

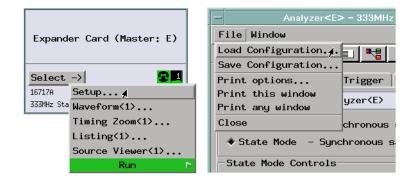


### Possible uses:

- To capture more function execution data (because only function execution states are stored in trace memory).
- To capture a "window" of program execution or look at consecutive executions of a function.
- To store (and time) the execution of a memory management subroutine.
- To store (and time) an access to a disk drive.

#### Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and format labels.



2. Load symbols from your program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	

- Capturing the Data1. Set up pattern resources that define the window start and window end<br/>events.
  - 2. Set up a trigger sequence where level 1, while storing no states, looks for the window start event; when it's found, the analyzer triggers. Level 2, while storing all states, looks for the window end event. At this point, the next sequence level can store no states (or you can branch back to the first level and store consecutive windows of program execution).

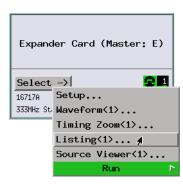
# Chapter 1: Measurement Examples **Software Development**

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Defaul	t Storing]Sta
General State	Tri
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop	Occurrenc
Replace Insert before	Insert aft
Trigger definition	
FIND PATTERN N TIMES         Find 1 occurrence of         ADDR = oard/source/q.elf:add_to_history Sym         then Turn on default storing         Trigger and goto Next	bols
2 FIND PATTERN N TIMES Find 1 ▲ occurrence of ADDR = source/q.elf:add_to_history+0x63 then Turn off default storing Goto 1	3 Symbols
Sampling Format Trigger Symbol Trigger Functions Settings Overview Defaul	t Storing Sta
Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing". Store by default Anything = At start of acquisition, Off = "Default Storing" is	

3. Select the Run button to start the measurement.

Unless the window of program execution fills trace memory, you may have to select the Stop button in order to display the captured states.

**Displaying the Data** 1. When the analyzer triggers, open the Listing window to show that the window of program execution was captured.



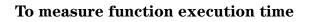
State Number	PC	MPC821/8	60 Inverse Assembler	Time
Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Relative
107	add_to_histo+004C	add	r8 r7 r8	624.000 ns
111	add_to_histo+0050	stb	r3 0000(r8)	624.000 ns
115	history_buff+0001		write 46	584.000 ns
116	add_to_histo+0054	lis	r12 0000	272.000 ns
120	add_to_histo+0058	li	r0 0000001	624.000 ns
124	add_to_histo+005C	stb	r0 41CE(r12)	624.000 ns
128	MX_add_to_history		write 01	588.000 ns
129	add_to_histo+0060	blr		272.000 ns
136	add_to_histo+0004	li	r8 0000000	14.837 ms
140	add_to_histo+0008	lis	r12 0000	620,000 ns
144	add_to_histo+000C	li	r0 0000001	624.000 ns
148	add_to_histo+0010	stb	r0 41CD(r12)	624.000 ns
152	ME_add_to_history		write 01	588.000 ns
153	add_to_histo+0014	b	:add_to_history+001C	272.000 ns
157	add_to_histo+001C	lis	r10 0000	544.000 ns
161	add_to_histo+0020	addi	r10 r10 412C	624.000 ns
165	_add_to_histo+0024_	add	<u>r11 r10 r8</u>	624,000 ns

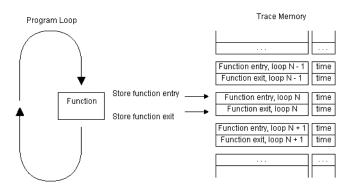
If the analyzer never triggers, the window start event never occurs.

You can also open the Source Viewer window and use the Step Source Previous and Next buttons to browse the captured data by associated source lines.

See Also

"If the trigger doesn't occur as expected" on page 309





Possible uses:

• To see if function execution times fall within specifications.

Probing the Target System 1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

Expander Card (Master: E) <ul> <li>File Window</li> <li>Load Configuration</li> <li>Save Configuration</li> <li>Print options</li> <li>Print this window</li> <li>yzer(E)</li> </ul> 16717A 333HHz Sta     Waveform(1)           Kaney Zoom(1)         Timing Zoom(1)           Listing(1)         State Mode - Synchronous s           Source Viewer(1)         State Mode Controls           State Mode Controls						
Select ->       Image: Card (Master: E)         Solution       Image: Card (Master: E)         Solution       Image: Card (Master: E)         Setup: Card (Master: E)       Image: Card (Master: E)         Solution       Image: Card (Master: E)         Image: Card (Master: E)       Image: Card (Master)         Image: Card (Mast				E	Analyzer <e< th=""><th>&gt; – 333MHz</th></e<>	> – 333MHz
Select ->       Save Configuration         16717A       Setup4         333Hz Sta       Waveform(1)         Timing Zoom(1)       Close         Listing(1)       State Mode - Synchronous so         Source Viewer(1)       Close in Mode - Synchronous so	Expar	der Card (Master: E)		F	ile Window	
Select ->       Print options       Trigger         16717A       Setup 4       Print this window       yzer(E)         333%Hz Sta       Waveform(1)       Print any window       chronous         Listing(1)       Source Viewer(1)       State Mode - Synchronous is				ĹĿ	oad Configuration	
16717A       Setup /       Print this window         333%Hz Sta       Waveform(1)       Print any window         Timing Zoom(1)       Close       chronous         Listing(1)       Source Viewer(1)       Other Mode - Synchronous is				S	ave Configuration	
333MHz Sta     Waveform(1)     Print any window     yzer(E)       Timing Zoom(1)     Close     chronous       Listing(1)     State Mode - Synchronous s       Source Viewer(1)     Close to b	Selec			P	rint options	Trigger
333mm2 3ta     Waveform(1)     Print any window     org/line       Timing Zoom(1)     Close     chronous       Listing(1)     State Mode - Synchronous s       Source Viewer(1)     Org/line	16717A	Setup		P	rint this window	(5)
Listing(1) Source Viewer(1)	333MHz 9	<sup>ta</sup> Waveform<1>		P	rint any window	yzer(E)
Source Viewer(1)		Timing Zoom<1>		С	lose	chronous
Source Viewer(1)		Listing(1)			A State Made Suma	
Run 🕞 State Mode Controls		Source Viewer(1)			<ul> <li>State Hode – Sync</li> </ul>	nronous s
		Run	>		State Mode Controls-	

2. Load symbols from your program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR /logic/demo/860_demo_board/source/q.elf	

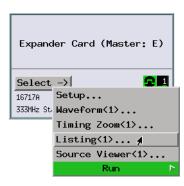
- **Capturing the Data** 1. Set up pattern resources that define function entry and exit events.
  - 2. Set up a trigger specification that stores only the entry and exit states of the function you're interested in. (This is the same as looking at the execution of a particular function, except only the entry and exit states are stored.) Be sure to turn ON the time count.

# Chapter 1: Measurement Examples **Software Development**

Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing S
General State
Find pattern n times
Store range until pattern occurs Store pattern2 until pattern1 occurs
Store nothing until pattern occurs 🛛 🗌 🗙 patte
Run until user stop
Replace Insert before Insert a
Trigger definition
1       FIND PATTERN N TIMES         Find 1 occurrence of         ADDR = oard/source/q.elf;add_to_history         Store sample         Trigger and goto
2 FIND PATTERN N TIMES
Find 1
ADDR = source/q.elf:add_to_history+0x60 Symbols
then Store sample
Goto 1
Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing S
Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing".
Store by default Nothing =
At start of acquisition, On = "Default Storing" is

3. Select the Run button to start the measurement.

**Displaying the Data** 1. Use the Listing display to show the captured function entry and exit points. Count relative time to show relative function execution times.

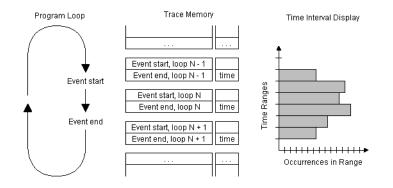


	State Number	PC	MPC821/860 Inverse Assembler	Time
	Decimal	Symbols	10=hex, 10.=decimal, %10=binary	Relative
tr	0	up:add_to_history	Undefined Opcode 7C4E7C4E	
	1	add_to_histo+0060	Undefined Opcode 4E7C4E7C	13.904 us
G1_	2	up:add_to_history	Undefined Opcode 7C4E7C4E	7,333 ms
	3	add_to_histo+0060	Undefined Opcode 4E7C4E7C	20.220 us
G2_	4	up:add_to_history	Undefined Opcode 7C4E7C4E	7,449 ms
	5	add_to_histo+0060	Undefined Opcode 4E7C4E7C	26.552 us
	6	up:add_to_history	Undefined Opcode 7C4E7C4E	7.461 ms
	7	add_to_histo+0060	Undefined Opcode 4E7C4E7C	32.872 us
	8	up:add_to_history	Undefined Opcode 7C4E7C4E	7.496 ms
	9	add_to_histo+0060	Undefined Opcode 4E7C4E7C	39.184 us
	10	up:add_to_history	Undefined Opcode 7C4E7C4E	22.858 ms
	11	add_to_histo+0060	Undefined Opcode 4E7C4E7C	45.496 us
	12	up:add_to_history	Undefined Opcode 7C4E7C4E	7.732 ms
	13	add_to_histo+0060		51.844 us
	14	up:add_to_history	Undefined Opcode 7C4E7C4E	7.878 ms
	15	add_to_histo+0060	Undefined Opcode 4E7C4E7C	58.144 us
	16	up:add_to_history	Undefined Opcode 7C4E7C4E	7.844 ms

You can use a global marker to search for the function exit states whose relative time values show the function execution time.

**See Also** "To measure function execution time (with SPA)" on page 218

"To trace execution within a function" on page 210



### To measure function execution time (with SPA)

The system performance analyzer's Time Interval display gives you a histogram of (and statistics on) event execution times.

**Requirements:** 

• This measurement requires the system performance analyzer (SPA) tool set.

#### Possible uses:

- To check how the execution time of a particular function varies.
- To determine if optimization of a function is needed.

Probing the Target<br/>System1. Use an analysis probe to connect the logic analyzer to the microprocessor,<br/>and use the provided configuration files to configure the analyzer and<br/>define labels.

Expander Card (Master: E)			- Analyzer <e> - 333MHz</e>		
			File Window		
	•			Load Configuration.	
				Save Configuration	
	Select		. I	Print options	Trigger
	16717A	Setup 🖌		Print this window	(5)
	333MHz Sta	Waveform<1>		Print any window	yzer <e></e>
		Timing Zoom<1>		Close	chronous
		Listing<1>		◆ State Mode - Sunc	hnonouc c
		Source Viewer<1>		▼ state riode - Sync	hronous s
		Run t	•	-State Mode Controls-	

2. Load symbols from your program's object module file.

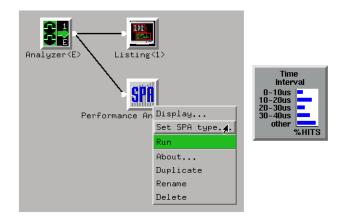
Sampling Format Trigger Symbol						
Object File User Defined						
Load This Object/Symbol File For Label: ADDR						
[/logic/demo/860_demo_board/source/q.elf Browse						
Create Symbol File (.ns) In This Directory:						
/logic/symbols/						
Object Files with Symbols Loaded For Label: ADDR						
/logic/demo/860_demo_board/source/q.elf						

**Capturing the Data** 1. Set up the logic analyzer trigger specification to capture only the entry and exit points of a function.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Defaul	lt Storing Sta
General State	Tri
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop	Occurrent patterr
Replace Insert before	Insert aft
Trigger definition	
1 FIND PATTERN N TIMES Find 1 occurrence of ADDR = oand/source/q.elf:add_to_history Sym then Store sample Trigger and goto Next	nbols
2 FIND PATTERN N TIMES Find 1 ♣ occurrence of ADDR = source/q.elf;add_to_history+0x6 then Store sample Goto 1	0 Symbols
Sampling Format Trigger Symbol Trigger Functions Settings Overview Defaul Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing".	lt Storing]Sta

2. Select the Run button to start the measurement.

**Displaying the Data** 1. In the Workspace window, use the system performance analyzer's Time Interval display to view the captured data.



2. Define the start and end of the event (that is, the function) whose time variations you wish to measure.

Performance Analysis<1>
File Window Graph Options Sort
Define ranges Qualify data Include other
Time Interval
Start of event
The start of the duration occurs when this event is: Present -
Event:
ADDR Symbols - Pattern - oard/source/q.elf;add_to_history
End of event
The end of the duration occurs when this event is: Present $=$
Event:
ADDR Symbols - Pattern - source/q.elf:add_to_history+0x60

3. Define *buckets* for captured time ranges.

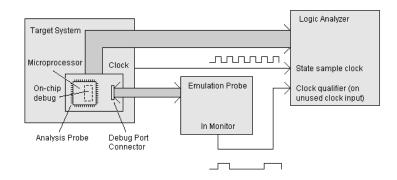
Time Ranges Hits(%)	– Auto Range – Performance Analys
0 s 100 ms 227 321(100 0%) 100 ms 200 Undo	-Range limits
200 ms 300 300 ms 400 Expand bucket 利	Start time: 13.872 us
400 ms 500 Delete	End time: 209.944 us
600 ms 700 Delete thru end	<u>y</u>
700 ms 800 View event times 800 ms 900	◆ Linear
800 ms 900 900 ms 1 Find range	Number of buckets: 10
	↓ Logarithmic
	Log base 🕩 🎽
	OK Apply Cancel

4. View the measurement results.

	Time Ir	nterval			
Time Ranges	Hits(%)	Ĩ			
13.872 us 33.479 us	51,191(22,5%)				
33.479 us 53.086 us	36,060(15,9%)				
53.086 us 72.694 us	30,226(13,3%)				
72.694 us 92.301 us	25,210(11,1%)				
92.301 us 111.908 us	21,354(9,4%)				
111.908 us 131.515 us	18,229(8.0%)				
131.515 us 151.122 us	15,294(6.7%)	_			
151.122 us 170.73 us	12,632(5,6%)	_			
170.73 us 190.337 us	9,751(4.3%)	_			
190.337 us 209.944 us	7,374(3.2%)				
	c	20%	40%	60% 80	100%
Display Information		-Bucket In	nformati	on	
Qualified Events: 2	227,321	Hits(%):	51,191	(22,52%)	
Total Acquisitions: 1		Range:	13,872	us - 33.47	79 us
	100.00%	Min/Max:	13.872	us/32.88 u	is 🗌
. –	0	Mean:	23.363		

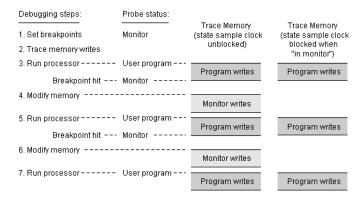


"To measure function execution time" on page  $214\,$ 



#### To omit monitor cycles from the trace

Use the emulator probe's "in monitor" signal to block the logic analyzer's state clock.



#### **Requirements:**

• You need a signal to tell the logic analyzer when the processor is executing in the monitor. The emulation probe provides such a signal.

Possible uses:

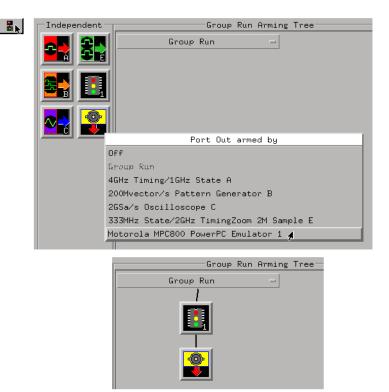
• To capture execution between a debugger's breakpoints.

### Probing the Target1. If you're using the emulation probe to provide the "in monitor" signal, set<br/>up the emulation probe connection to the target system processor.

- 2. Also, configure the emulation probe to output its "in monitor" signal.
- 3. Connect the logic analyzer Port Out signal to one of the unused clock inputs on the logic analyzer pods. (An Intermodule setup will be used to route the emulation module's "in monitor" signal to the logic analyzer's Port Out BNC connector.)
- 4. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

**Capturing the Data** 

1. Set up the Intermodule window so that Port Out is armed by the emulation module.



2. Set up the trigger specification as you would normally, but also set up the logic analyzer state clock to only occur when the "in monitor" signal is false.

Expander Card (Master: E)								
Select ->								
16717A Setu	ip 🗚							
333MHz Sta Wave	form<1>							
Timi	ng Zoom<1>							
List	ing<1>							
Sour	ce Viewer<1>							
	Run 🕨							
Clock Set	up							
Mode: Ma	ster only 👤 🔄 Advanced Clocking							
Pod	E4 E3 E2 E1							
Clock	MLKJ							
Activity								
Master	0ffl0ffl 0   ♣  => r (J†) • (K=0) ]							
	* Off							
	Rising Edge							
	Falling Edge							
	Both Edges							

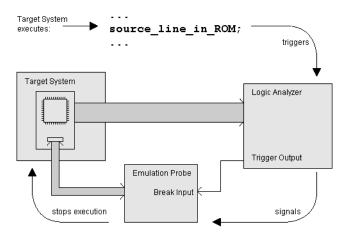
3. Select the Group Run button to start the trace measurement.

You can freely set breakpoints and examine the state of the microprocessor knowing that the logic analyzer will not capture any of the monitor cycles.

If your logic analyzer measurements count time, you'll see large time values in the trace when the microprocessor is executing in it's debug monitor mode.

**Displaying the Data** 1. Use the Listing display to the captured data. You may have to stop the measurement to view captured data (if the events you're capturing are infrequent enough to allow breaks and monitor cycles to be captured without clock qualification).

### To stop execution at a source line (in ROM)



Normally, you would use a debugger to stop microprocessor execution at a particular source line. However, if the debugger implements breakpoints by replacing code (or some other mechanism that requires code to be in RAM), you will not be able to set breakpoints on source code that exists in ROM. Luckily, the logic analyzer can tell the emulation probe to stop processor execution when it captures a particular event.

Requirements:

• To make this measurement, you need a microprocessor run control mechanism, like an emulation probe, that can stop microprocessor execution when a trigger signal is received from the logic analyzer.

Possible uses:

• To stop microprocessor execution when debugger breakpoints cannot.

 Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

- 2. Download program symbols to the logic analyzer and set up access to the program source files.
- 3. Make sure the emulation probe (or emulation module and emulation

### Probing the Target System

adapter) has been connected to the target system.

**Capturing the Data** 1. Set up the logic analyzer to trigger on the source line you're interested in.

Expander Card (Master: E)					
Select					
16717A	Setup				
333MHz Sta	Waveform<1>				
	Timing Zoom<1>				
	Listing<1>				
Source Viewer<1>					
Run 🕨					

St	ep Source 🗍 Goto In List	ing Br	rowse Source	Text Search	Symbols
	Courses Edite Name				
	ew Source File Name				
e	ecs2.d		<u> </u>	<u>File Se</u>	lection
Disp	layed File: /logic/demo.	/860_dem	o_board/sourc	e/ecs2.c	
139					
140	main()				
141	(				
142	<pre>boot_q();</pre>				
143	-				
144	init_system();				
145	proc_spec_init();				
146	- / .				
147	for (;;)				
148	{		`		
149	update_system(r	um_cneci			
150 151	num_checks++;		lit	ne # 149	
151	update_display proc_specific()	Trace be	fore this lin	ne	
153	<pre>proc_specific(; }</pre>				
154	>	Trace at	oout this line	° 🐔 📃 🕺	
155	,	Trace at	fter this line	9	
156	/********************	Coto thi	ie lino in lie	sting before c	urront etc
157	* FUNCTION: update_di	0010 111		string berole c	urrent sta
158	* PARMS: counter -	Goto thi	is line in lig	sting after cu	rrent stat
159	* DESCRIPTION:				

#### NOTE:

Source Viewer commands that set up triggers only modify the trigger condition. They do not modify the trigger position, storage qualifiers, else branch conditions, or other levels in the trigger sequence.

2. In the Source Viewer window, choose the Trace->Enable - Break Emulator

On Trigger command.

-	Source Viewer<1>			
File Window Options	Trace			
	Trace Everything			
	Trace Until Stop			
	Trace About Variable			
	Trace About Function			
	Trace About Line #			
	Enable 🚽 Break Emulator On Trigger			
	Disable - Break Emulator On Trigger			

This command will automatically set up the Intermodule window to specify that the emulation module's break input be armed by the logic analyzer's trigger.

3. Select the Group Run button to start the measurement.

**Displaying the Data** 1. When the logic analyzer trigger is found, microprocessor execution stops.

-	S	tatus/Erroi	<sup>-</sup> Log – Emul	ator<1>	•				
File Navigate									
Popup dialog upon receiving error/status message? 🕈 Yes 💠 No									
15:13:11 - Intermodule trigger break									
Clear Messages Close									
File Win	Stop	Step	Help Reset						
MPC860	In Backgr	ound							

Note that microprocessor execution does not stop immediately after the logic analyzer trigger because of delay in the intermodule signals and the

speed of the processor.

**See Also** "To trace about a source line" on page 199

"To stop execution on a corrupt variable" on page 245

### Analyzing Real-Time Variable Access

- "To find NULL pointer de-references" on page 229
- "To trace a variable's values" on page 231
- "To find where variables are accessed from" on page 236
- "To trace before a variable value" on page 240
- "To stop execution on a corrupt variable" on page 245

#### To find NULL pointer de-references

Because a NULL pointer has particular address, commonly 0, you can trace accesses of the NULL address to find NULL pointer dereferences.

Possible uses:

• To check for the possible cause of NULL pointer de-references.

# Probing the Target1. Use an analysis probe to connect the logic analyzer to the microprocessor,<br/>and use the provided configuration files to configure the analyzer and<br/>define labels.

			- Analyzer <e></e>	> – 333MHz
Expander	Card (Master: E)		File Window	
•		1	Load Configuration.	
		- 1	Save Configuration	
Select $\rightarrow$			Print options	Trigger
16717A Set	tup,		Print this window	
	veform<1>		Print any window	yzer <e></e>
Tiı	ming Zoom<1>		Close	chronous
Lis	sting<1>			
50	urce Viewer<1>		🔷 State Mode – Sync	hronous s
500				
	Run		-State Mode Controls-	

**Capturing the Data** 1. Set up to trigger on an access of address 0.

Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default
General State
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop
Replace Insert before
Trigger definition
1 FIND PATTERN N TIMES
Find 1
ADDR = 00000000 Hex

The states that are stored before the trigger may show a NULL pointer dereference.

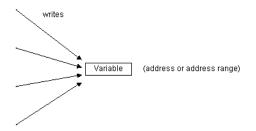
2. Select the Run button to start the measurement.

# **Displaying the Data** 1. If the analyzer triggers, open the Listing window to display the access of address 0. You may want to load an inverse assembler, load symbols, and display symbols in the address label column.

2. You can also open the Source Viewer window and use the Step Source

Previous and Next buttons to browse the captured data by associated source lines.

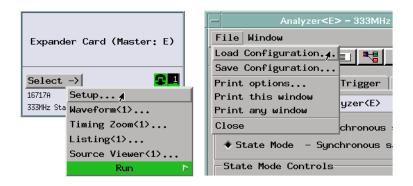
#### To trace a variable's values



Store only variable write accesses.

Possible uses:

- To look for inappropriate variable values (in memory locations, not local variables on the stack or in microprocessor registers).
- Probing the Target1.Use an analysis probe to connect the logic analyzer to the microprocessor,<br/>and use the provided configuration files to configure the analyzer and<br/>define labels.



2. Load symbols from your program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label:ADDR	
/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	

**Capturing the Data** 1. Set up a trigger specification that stores only writes to the variable addresses.

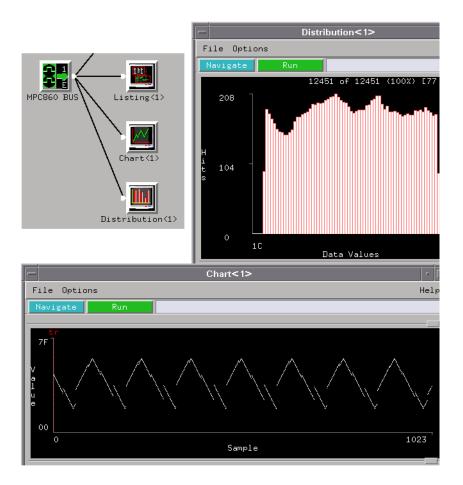
Sampling Format Trigger	Symbol		
Trigger Functions Settings (	)verview]Defau	lt Storing Status Sa	ave/Recall]
General State		Trigger f	unction libraries
Find pattern n times Store range until pattern oc Store pattern2 until pattern Store nothing until pattern Run until user stop	1 occurs	Occurrence 1 pattern	Occurrence n pattern
Replace Inse	ert before	Insert after	Delete
Trigger definition			
Find 1 occurrence	q.elf:target_te		l
Sampling Format Trigger	Symbol		
Trigger Functions Settings	Dverview Defau	lt Storing Status S	ave/Recall]
Specify what to store by def Using store actions in a seq level overrides the "Default	uence	Store if = DR o_board/sourc	e/q.elf:target_temp
Store by default <u>Custom</u> — At start of acquisition, <u>On</u> "Default Storing" is		AT = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	XXXXXXX0X Binary

- 2. Select the Run button to start the measurement.
- **Displaying the Data** 1. Open the Listing window to view the captured variable values.

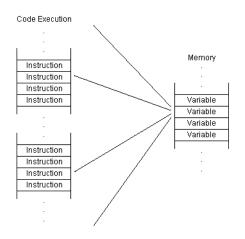
	ler Card (Master: E)
Select	
16717A	Setup
333MHz St.	Waveform<1>
<u> </u>	Timing Zoom<1>
	Listing<1> 🖌
	Source Viewer(1)
	Run 🕨

	State Number	PC	MPC821/860 Inverse #	Assembler	Time
	Decimal	Symbols	10=hex, 10.=decimal	l, %10=binary	Relative
tr	0	q.elf:target_temp	write 4E		
	1	q.elf:target_temp	write 4I	)	11.687 ms
	2	q.elf:target_temp	write 40		13.694 ms
	3	q.elf:target_temp	write 4E		11.028 ms
	4	q.elf:target_temp	write 4f		11.578 ms
	5	q.elf:target_temp	write 49		14.511 ms
	6	q.elf:target_temp	write 48		11.869 ms
	7	q.elf:target_temp	write 47		11.336 ms
	8	q.elf:target_temp	write 46	5	11.223 ms
	9	q.elf:target_temp	write 45		11.191 ms
	10	q.elf:target_temp	write 44	ļ.	11.268 ms
	11	q.elf:target_temp	write 43		11.213 ms
	12	q.elf:target_temp	write 42		11.283 ms
	13	q.elf:target_temp	write 41		11.352 ms
	14	q.elf:target_temp	write 40		11.346 ms
	15	q.elf:target_temp	write 3F		11.353 ms
	16	g.elf:target_temp_	write 3E		11,415 ms

In the Workspace window, you can also set up the Chart, Distribution, or system performance analyzer displays for different views of the variable values.



### To find where variables are accessed from



By storing only writes to a variable and context storing instructions, you will see the code that writes to the variable.

Possible uses:

- To find which functions access a global variable.
- To trace writers of a variable.

#### Probing the Target System

1. Typically, this measurement is made with a state analyzer and an analysis probe capturing software execution. Configure the analyzer and format labels by loading the configuration files provided with the analysis probe.

	_	Analyzer <e></e>	• – 333MHz
Expander Card (Master: E)	F	ile Window	
	Ŀ	oad Configuration.	<b>∎   ¤∹≋  </b>
	S	ave Configuration	
Select ->	P	rint options	Trigger
16717A Setup	P	rint this window	
333MHz Sta Waveform<1>	P	rint any window	yzer <e></e>
Timing Zoom<1>	С	lose	chronous
Listing<1>	Τ	◆ State Mode – Sync	hronous s
Source Viewer<1>		• state node - sync	ni onous s
Run		State Mode Controls-	

2. Load symbols from your program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label: ADDR	
[/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR	
/logic/demo/860_demo_board/source/q.elf	

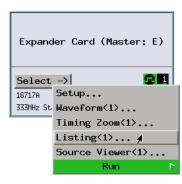
Capturing the Data1. Set up a trigger specification to store only variable accesses and, for each<br/>variable access, another state that indicates where the variable was<br/>accessed from.

If the analyzer has context store capability, use it to store any instruction fetch that occurs before the variable access.

If the analyzer doesn't have context store capability, you can look for the variable access and store any instruction fetch that occurs after the access.

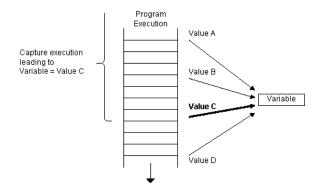
2. Select the Run button to start the measurement.

**Displaying the Data** 1. Open the Listing window to display the captured data. You may want to load an inverse assembler and display symbols in the address label column.



	State Number	PC	MPC821/80	50 Inverse Assembler	Time
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Relative
	55	:get_targets+006C	ba	005F485C	272.000 ns
	56	q.elf:target_temp		write 5F	16.148 ms
	57	:get_targets+006C	bl	004E7E24	272.000 ns
	58	q.elf:target_temp		write 5E	16.933 ms
	59	:get_targets+006C	bl	004D7E24	272.000 ns
	60	q.elf:target_temp		write 5D	19.594 ms
	61	:get_targets+006C	b	004D7324	272.000 ns
	62	q.elf:target_temp		write 5D	10.978 ms
G1_	63	<pre>save_points+0034</pre>	addis	r10 r28 485B	272,000 ns
	64	q.elf:target_temp		write 50	5.859 ms
	65	:get_targets+006C	ba	005B4858	276.000 ns
	66	q.elf:target_temp		write 5B	17.478 ms
	67	:get_targets+006C	bl	004A7E20	272.000 ns
	68	q.elf:target_temp		write 5A	16.138 ms
	69	:get_targets+006C	b	00497E20	272.000 ns
	70	q.elf:target_temp		write 59	16.068 ms
	71	:get_targets+006C	bla	00584854	272,000 ns

### To trace before a variable value

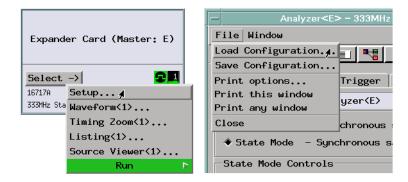


Possible uses:

- To find the cause of a variable corruption.
- To find a variable value outside a specified range.
- To find when a variable equals a particular value.

### Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



2. Load symbols from your program's object module file.

Sampling Format Trigger Symbol	
Object File User Defined	
Load This Object/Symbol File For Label: ADDR	
[/logic/demo/860_demo_board/source/q.elf	Browse
Create Symbol File (.ns) In This Directory:	
/logic/symbols/	Browse
Object Files with Symbols Loaded For Label: ADDR	
/ Togic/ demo/ 000_demo_boar d/ source/ q.eit	

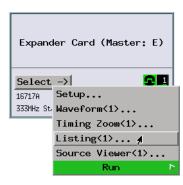
Capturing the Data1. Set up a trigger specification that, while storing all states, looks for a particular value being written to a variable. Make sure the trigger point appears at the end of the trace.

Sampling Format Trigger Symbol
Analyzer Name: MPC860 BUS 🔽 On
√ Timing Mode - Asynchronous sampling clocked internally by an     ◆ State Mode - Synchronous sampling clocked by the Device Und
State Mode Controls
167 MHz / 2M State ₤ Trigger Position End ₤
Acquisition depth 8K 🛓 Start End User Defined
Clock Setup
Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing Status Sav
General State Trigger fu
General State Trigger fu
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop
Find pattern n times       Store range until pattern occurs       Store pattern2 until pattern1 occurs       Store nothing until pattern occurs       Run until user stop       Replace       Insert before       Insert after
Find pattern n times         Store range until pattern occurs         Store pattern2 until pattern occurs         Store nothing until pattern occurs         Run until user stop         Replace       Insert before         Trigger definition
Find pattern n times       Store range until pattern occurs       Store pattern2 until pattern1 occurs       Store nothing until pattern occurs       Run until user stop       Replace       Insert before       Insert after
Find pattern n times         Store range until pattern occurs         Store pattern2 until pattern occurs         Store nothing until pattern occurs         Run until user stop         Replace       Insert before         Trigger definition
ind pattern n times         Store range until pattern occurs         Store nothing until pattern occurs         Run until user stop         Replace         Insert before         Insert after         Trigger definition         1         FIND PATTERN N TIMES
Find pattern n times         Store range until pattern occurs         Store pattern2 until pattern occurs         Store nothing until pattern occurs         Run until user stop         Replace         Insert before         Insert after         Trigger definition         1         FIND PATTERN N TIMES         Find         1
ind pattern n times         Store range until pattern occurs         Store pattern2 until pattern occurs         Store nothing until pattern occurs         Run until user stop         Replace         Insert before         Insert after         Trigger definition         1         FIND PATTERN N TIMES         Find         Occurrence of         ADDR         Docodd/source/q.elf:target_temp         Symbols
ind pattern n times         Store range until pattern occurs         Store pattern2 until pattern occurs         Store nothing until pattern occurs         Run until user stop         Replace         Insert before         Insert after         Trigger definition         1         FIND PATTERN N TIMES         Find         Image: Pattern occurs         Image: Pattern occurs         Replace         Image: Pattern occurs         Replace         Insert before         Insert after         Trigger definition         1         FIND PATTERN N TIMES         Find         Image: Pattern occurs         Image: Patt

2. Select the Run button to start the measurement.

#### Displaying the Data

1. Open the Listing window to view the execution that led to the write of the particular value to the variable.

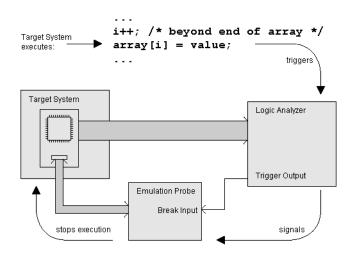


	State Number	PC	MPC821/86	MPC821/860 Inverse Assembler	
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Relative
	-44	q.elf:temp_target		read 20	588.000 ns
	-43	:get_targets+0018	cmplw	cr0 r11 r10	272.000 ns
	-39	:get_targets+001C	bne	cr0 upd:get_targets+0(	624.000 ns
	-35	:get_targets+004C	lbz	r8 0000(r3)	544.000 ns
	-31	q.elf:target_temp		read 2E	584.000 ns
	-30	:get_targets+0050	lis	r12 0000	272,000 ns
	-26	:get_targets+0054	lbz	r7 408B(r12)	624.000 ns
	-22	q.elf:temp_target		read 20	584.000 ns
	-21	<pre>:get_targets+0058</pre>	стры	cr0 r8 r7	272.000 ns
	-17	:get_targets+005C	ble	cr0 upd:get_targets+0(	624.000 ns
	-13	:get_targets+0060	lbz	r11 0000(r3)	624.000 ns
	-9	q.elf:target_temp		read 2E	584.000 ns
	-8	:get_targets+0064	subi	r11 r11 0001	272.000 ns
	-4	:get_targets+0068	stb	r11 0000(r3)	624.000 ns
tr	0	q.elf:target_temp		write 2D	584,000 ns
	1	:get_targets+006C		pgm 4800****	272.000 ns

A correlated source viewer may be helpful in relating the execution to your high-level program.

Expander Card (Master: E)				
Select	->			
16717A	Setup			
333MHz Sta	Waveform<1>			
	Timing Zoom<1>			
	Listing<1>			
Source Viewer<1>				
Run				

Step Source   Goto In Listing   Browse Source   Text Search					
To Captured Source Line					
Previous Next					
Displayed File: /logic/demo/860_demo_board/source/update_sys.c					
110 temp_target = MIN_TEMP;					
111 }					
112 }					
113					
114 if (*temperature > temp_target)					
115 (					
116 (*temperature);					
117 }					
118 else					
119 {					
120 (*temperature)++;					
121 }					
122					
123 MAKEBAR(ARG1);					
124					
125 MX_get_targets = 1;					
126 }					
127					



### To stop execution on a corrupt variable

Possible uses:

• To inspect the state of the microprocessor at some point while capturing the real-time execution that leads up to it.

#### Probing the Target System

1. Use an emulation probe (either connected to a debug port in target system or connected to an analysis probe) and some kind of debugging interface (3rd party debugger, emulation control tool set, etc.).



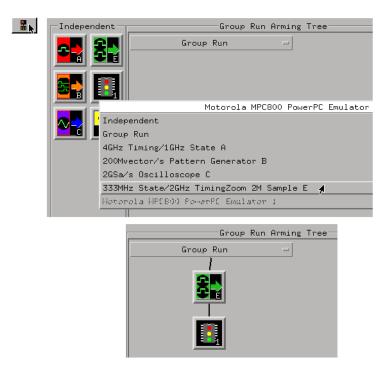
2. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.

Expander Card (Master: E)			- Analyzer <e></e>	> – 333MHz
			File Window	
			Load Configuration.	
			Save Configuration	
Select			Print options	Trigger
16717A	Setup 🖌		Print this window	
333MHz Sta	Waveform<1>		Print any window	yzer <e></e>
	Timing Zoom<1>		Close	chronous
Listing(1)			State Mode - Sync	hranauc c
	Source Viewer(1)		• State Hode - Sync	in onous s
	Run		-State Mode Controls-	

3. Load symbols from your program's object module file.

Sampling Format Trigger Symbol				
Object File User Defined				
Load This Object/Symbol File For Label:ADDR				
/logic/demo/860_demo_board/source/q.elf	Browse			
Create Symbol File (.ns) In This Directory:				
/logic/symbols/	Browse			
Object Files with Symbols Loaded For Label: ADDR				
/logic/demo/860_demo_board/source/q.elf				

4. Open the Intermodule window and set up the emulation probe to be armed by the logic analyzer.



**Capturing the Data** 1. Set up the logic analyzer to trigger on variable address and a particular data value write.

Sampling Format Trigger Symbol					
Trigger Functions Settings Overview Default Storing	Status Save				
General State	Trigger fun				
Store pattern2 until pattern1 occurs	attern				
Replace Insert before Insert	after				
Trigger definition					
1 FIND PATTERN N TIMES					
Find 1					
ADDR = o_board/source/q.elf:target_temp Symbols	And				
DATA = 2DXXXXXX Hex And					
STAT = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX					
then Trigger, arm out, and fill memory					

2. Select the Group Run button to start the measurement.

When the logic analyzer triggers, the emulation probe stops user program execution and the processor continues to run in its background mode.

- Status/Error Log - Emulator<1>	· 🗆			
File Navigate				
Popup dialog upon receiving error/status message? 🔶 Yes 💠 No				
15:13:11 - Intermodule trigger break				
Clear Messages Close				
Run Control – Emulator<1>				
File Window Help				
Run Stop Step Reset MPC860In Background				

**Displaying the Data** 1. Use the Listing display to view the execution that leads up to the variable write. (You may have to stop the measurement before you can display the captured data.)

Expander Card (Master: E)				
Select				
16717A	Setup			
333MHz St	Waveform<1>			
	Timing Zoom<1>			
Listing<1> 🖌				
Source Viewer<1>				
Run 🕨				

	State Number	PC	MPC821/860	Inverse Assembler	Time
	Decimal	Symbols	10=hex, 1	0.=decimal, %10=binary	Relative
	-26	:get_targets+0054	lbz	r7 408B(r12)	624.000 ns
	-22	q.elf:temp_target		read 20	584.000 ns
	-21	:get_targets+0058	cmpw	cr0 r8 r7	272.000 ns
	-17	:get_targets+005C	ble	cr0 upd:get_targets+0(	624.000 ns
	-13	:get_targets+0060	lbz	r11 0000(r3)	624.000 ns
	-9	q.elf:target_temp		read 2E	584.000 ns
	-8	:get_targets+0064	subi	r11 r11 0001	280.000 ns
	-4	:get_targets+0068	stb	r11 0000(r3)	624.000 ns
tr	0	q.elf:target_temp		write 2D	584.000 ns
	1	:get_targets+006C	b	upd:get_targets+007C	272.000 ns
	5	:get_targets+007C	lis	r12 0000	544.000 ns
	9	:get_targets+0080	lbz	r9 4349(r12)	624.000 ns
	13	rce∕q.elf:MaKeBaR		read 01	584.000 ns
	14	<pre>:get_targets+0084</pre>	extsb	r9 r9	272.000 ns
	18	:get_targets+0088	cmplwi	cr0 r9 0000	624.000 ns
	22	<pre>:get_targets+008C</pre>	beq	cr0 upd:get_targets+0(	624.000 ns
	26	:get_targets+0090	<u>li</u>	r6 00000000	624,000 ns

Note that processor execution doesn't stop immediately after the variable write and that additional cycles are executed before the processor is halted.

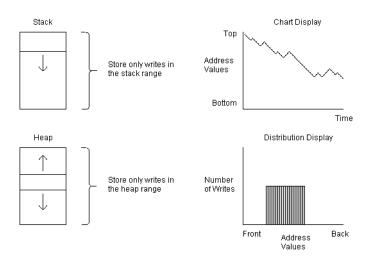
"To trace before a variable value" on page 240

"To stop execution at a source line (in ROM)" on page 226

### Analyzing Real-Time Memory Usage

- "To monitor stack or heap usage" on page 251
- "To find stack overflow or guarded memory access" on page 255

See Also



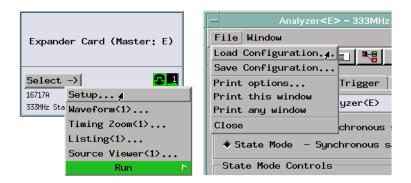
### To monitor stack or heap usage

Possible uses:

- To determine the necessary stack or heap size or view stack or heap usage.
- To look for stack or heap corruption.

#### Probing the Target System

1. Use an analysis probe to connect the logic analyzer to the microprocessor, and use the provided configuration files to configure the analyzer and define labels.



2. Load symbols from your program's object module file.

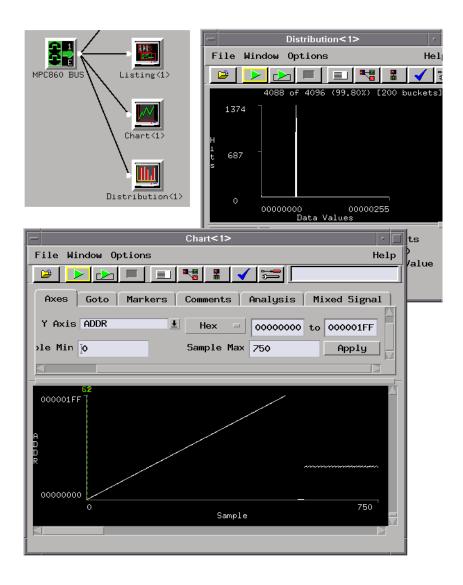
Sampling Format Trigger Symbol				
Object File User Defined				
Load This Object/Symbol File For Label: ADDR				
[/logic/demo/860_demo_board/source/q.elf Browse				
Create Symbol File (.ns) In This Directory:				
/logic/symbols/	Browse			
Object Files with Symbols Loaded For Label: ADDR				
/logic/demo/860_demo_board/source/q.elf				

**Capturing the Data** 1. Set up a trigger specification that stores only writes into the range of memory addresses reserved for the stack or heap.

Sampling Format Trigger Symbol	ſ
Trigger Functions Settings Overview D	efault Storing Status Save/Recall
General State	Trigger function libraries
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop	Occurrence 1 Occurrence n pattern pattern
Replace Insert before	Insert after Delete
Trigger definition	
Find 1 cocurrence of ADDR In range 00000000 00003FF STAT = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx	F Hex And Binary
Sampling Format Trigger Symbol	1
Trigger Functions Settings Overview D	efault Storing Status Save/Recall
Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing".	Store if         Image         00000000         00003FFF         Hex         Hex
Store by default Custom = At start of acquisition, On = "Default Storing" is	= xxxxxxxxxxxxx Binary Group events

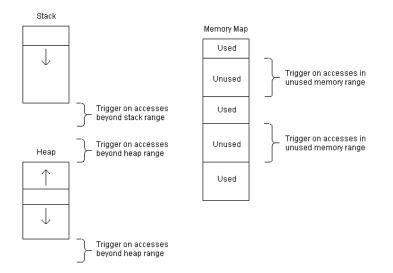
- 2. Select the Run button to start the measurement.
- **Displaying the Data** 1. In the Workspace window, you can use the Chart display to view the addresses in the stack or heap range that were written to.

You can also use the Distribution display to view the address locations that were written to and the number of number of writes that were made to each location.





"To find stack overflow or guarded memory access" on page  $255\,$ 



#### To find stack overflow or guarded memory access

Possible uses:

- To trace execution that leads to stack or heap overflow.
- To find accesses to non-existent memory or memory that should not be accessed (in other words, guarded memory).

# Probing the Target1.Use an analysis probe to connect the logic analyzer to the microprocessor,<br/>and use the provided configuration files to configure the analyzer and<br/>define labels.

		_
	Analyzer <e> - 333MI</e>	Hz
Expander Card (Master: E)	File Window	
	Load Configuration.4.	ī
	Save Configuration	-
Select ->	Print options Trigger	- 1
16717A Setup 🖌	Print this window	_
333MHz Sta Waveform<1>	Print any window	
Timing Zoom<1>	Close chronous	s
Listing<1>	◆ State Mode - Synchronous	
Source Viewer<1>	• state node - synchronous	2
Run	State Mode Controls	

Capturing the Data
 Set up the logic analyzer to trigger on accesses outside the range of the stack or heap, or accesses of memory that does not exist, and store execution that leads up to the bad access.

- 2. Select the Run button to start the measurement.
- Displaying the Data1. Open the Listing display and correlated Source Viewer window to view the<br/>execution that led to the stack or heap overflow or the guarded memory<br/>access.
- See Also "To monitor stack or heap usage" on page 251

### System Integration

Making Cross-Domain Measurements

- "To capture software execution when a scope triggers" on page 258
- "To generate patterns when a source line executes" on page 262
- "To arm one logic analyzer with another's trigger" on page 266
- "To arm a state machine with a timing machine trigger" on page 271
- "To arm an oscilloscope when the analyzer triggers" on page 277

Making System Profile Measurements

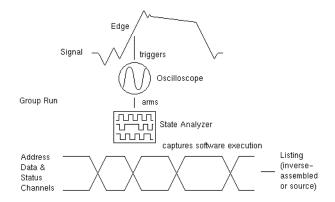
- "To isolate the root cause of a performance bottleneck" on page 283
- "To simulate bus occupation and measure SW performance" on page 287

Isolating Critical Defects

- "To capture SW execution on a setup or hold violation" on page 289
- "To trigger an oscilloscope when a source line executes" on page 294

### Making Cross-Domain Measurements

- "To capture software execution when a scope triggers" on page 258
- "To generate patterns when a source line executes" on page 262
- "To arm one logic analyzer with another's trigger" on page 266
- "To arm a state machine with a timing machine trigger" on page 271
- "To arm an oscilloscope when the analyzer triggers" on page 277



#### To capture software execution when a scope triggers

#### Possible uses:

To check whether a signal anomaly is related to software execution. ٠

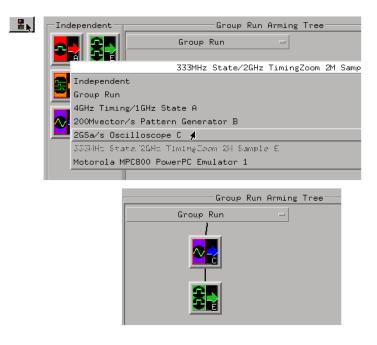
**Probing the Target** 1. Connect the oscilloscope channel probe to the signal of interest in the target system.

- 2. Open the oscilloscope display, select the Channels tab, and set up the oscilloscope channel.
- 3. Configure a state analysis machine (with an analysis probe) to capture software execution (pre-defined format is included with the analysis probe).

#### **Capturing the Data**

System

- 1. Set up the oscilloscope to trigger on the signal edge of interest.
  - 2. Open the Intermodule window, and set up the logic analyzer to be armed by the oscilloscope trigger.

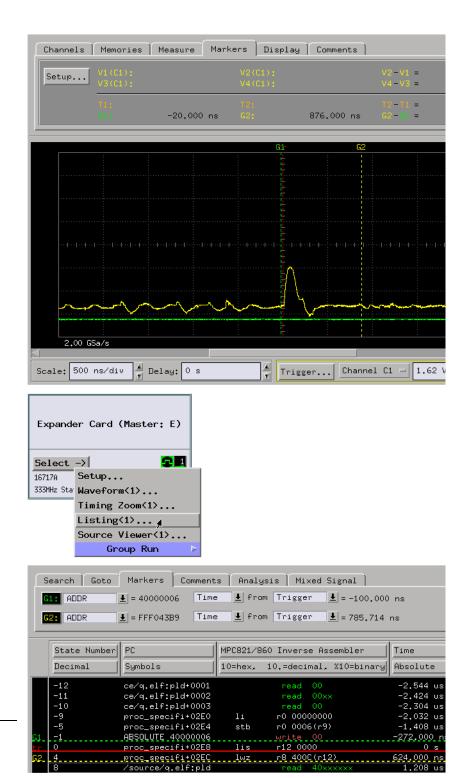


3. Set up the state analyzer to trigger on anything (after the arm).

Expande	r Card (Mast	er: E)	
Select -	->		
	Setup		
	laveform<1>.		
·	Timing Zoom<	1>	
	isting<1>		
5	Source Viewe	r<1>	
	Group R	un Þ	
			_
Samplin	g   Format	Trigger Symbol	
Trigger	European lo		
1148801	runctions [Set	tings Overview Defa	ult
General	,	tings Overview Defa	ult
General Find pat Store ra Store pa Store no	State .tern n times .nge until pat	tern occurs pattern1 occurs	
General Find pat Store ra Store pa Store no	, tern n times nge until pat ttern2 until thing until p	tern occurs pattern1 occurs	
General Find pat Store ra Store pa Store no Run unti	, tern n times nge until pat ttern2 until thing until p	tern occurs pattern1 occurs	
General Find pat Store ra Store no Run unti	State tern n times inge until pat ittern2 until thing until p l user stop	tern occurs pattern1 occurs attern occurs	
General Find pat Store ra Store no Run unti	State tern n times nge until pat ttern2 until thing until p l user stop teplace	tern occurs pattern1 occurs attern occurs	
General Find pat Store ra Store no Run unti	State tern n times inge until pat ittern2 until p l user stop eplace definition	tern occurs pattern1 occurs attern occurs	
General Find pat Store ra Store no Run unti	State tern n times inge until pat ittern2 until itting until p l user stop deplace definition FOR ARM IN	tern occurs pattern1 occurs attern occurs	

Count time in the logic analyzer so that the captured data may be correlated.

- 4. Select the Group Run button to start the measurement.
- **Displaying the Data** 1. Use global markers to show the correlation between the oscilloscope trigger and the captured software execution.

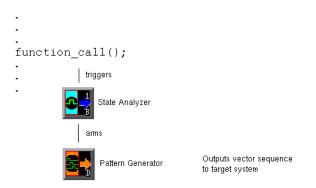


You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

You may want to open the Source Viewer window to view the source code associated with the oscilloscope trigger.

"To make basic oscilloscope measurements" on page 11

#### To generate patterns when a source line executes



Possible uses:

• To synchronize a sequence of target system test vectors.

Probing the Target<br/>System1. Configure a logic analyzer (with an analysis probe) for capturing software<br/>execution by loading the configuration file that is included with the<br/>analysis probe.

2. Select the pattern generator probing, connect the probes, map probe channels to labels, configure the vector output mode and clock source, and build a sequence of test vectors.

**Capturing the Data** 

See Also

**the Data** 1. Set up the logic analyzer to trigger on the source line of interest.

Expander Card	(Master: E)		
Select $\rightarrow$			
16717A Setup.			
333MHz Sta Wavefor	~m<1>		
Timing	Zoom<1>		
Listing	<b>_</b>		
Source	Viewer<1>		
	Run 🕨		
Sampling For	mat Trigger	Symbol ]	
Trigger Functio	ons Settings Ove	erview]Defau	lt Storing Sta
General State			Tri
Store pattern2	il pattern occu until pattern1 until pattern oc	occurs	Occurrenc
	F		
Replace	Insert	: before	Insert aft
Trigger defini			
Trigger defini			
		-	
1 FIND PATTERN	N TIMES		19 Line #s

2. Open the Intermodule window, and set up the pattern generator to be armed by the logic analyzer trigger.

	ndependent	Group Run	Arming Tree	
		Group Run	-	
		200Mvector/s Patt	onn Conceptor i	D armod bu
	Independent Group Run	200HVBCC017'S Pact	ern denerator .	b anned by
	4GHz Timing/1GHz St	ate A		
	200Hvector a Patter	n Generator B		
	2GSa/s Oscilloscope	С		
	333MHz State/2GHz T.	imingZoom 2M Sampl	.e E 🔺	
	Motorola MPC800 Pow	erPC Emulator 1		
from 333N vector. b	- 200Mved or/s Pattern Generator (Hz State/2GHz TimingZ where should it go? here: Start of Init ±	oom 2M Sample E, a	n signal	IT IMB'
	OK Ed	it Sequence	Cancel	
		Group Run Group Run	Arming Tree	

3. Set up the pattern generator to wait for the logic analyzer trigger before outputting its test vectors.



F	ormat	Sequence	Macr	·o ]			
F		n Fills					
	Fix	ed C	ount	. Rotat	.e	Toggle.	Random
			Data	Dreg/*Ireg	R/*₩ [	Enable^[	*PGenEnable
		<b>-</b>					
	Line	Instruction	Hex	Binary	Binary	Binary	Binary
0		INIT START					
1			00				0
2							
3		WAIT IMB					
4			00				1
5							0
6		MACRO	_	Instruction			
7		MACRO		Instruction			
8		MACRO	_	Instruction			
9	<u> </u>	MACRO	Write	Instruction	(0000000)	1)	
10		INIT END					
11	_	MAIN START	00	0	0	0	0
13					v 	U 	U 
14	-	MACRO	l.l.n.i.t.c	Instruction	0000000	0)	
19	·	MACRO		Data(0000004		0/	
16	-	MACRO		Data(0000000			
1	-	MACRO		Data(0000000			
18		MACRO		Data(0000000			
19	9	MACRO	Write	Data(0000000	SF)		
20	0						
21	1	MAIN END					

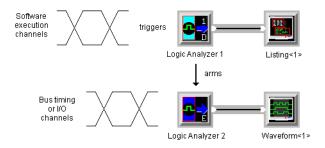
- 4. Select the Group Run button to start the measurement.
- **Displaying the Data** 1. You can use the Listing display to show the logic analyzer states captured after the source line trigger.

The states captured after the trigger will show the target system's response to the pattern generator stimulus.

**See Also** "To generate pattern stimulus on devices" on page 75

"To simulate particular interrupt sequences" on page 191

### To arm one logic analyzer with another's trigger



#### Possible uses:

• To correlate execution in different parts of the target, for example, software execution and standard bus execution, or software execution in different parts of a multi-processor system.

Probing the Target1. Configure each logic analyzer, set up the analyzer's sample clock input if<br/>it's a state analyzer, and format labels for the logic analysis channels used.

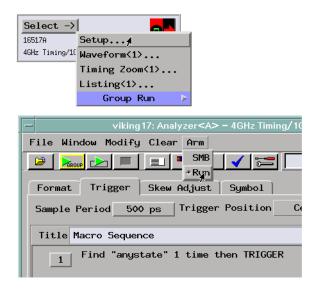
#### **Capturing the Data** 1. Set up one logic analyzer to trigger on the event of interest.

Expande	er Card (Mast	ter: E)	
Select	->		
16717A	Setup 🖌		
333MHz Sta	Waveform<1>.		
	Timing Zoom<	(1)	
	Listing<1>		
	Source Viewe	er<1>	
	Run		
	ng Format	Trigger Symbol ) ttings Overview Defa	ult Stoping Sta
General	,	ccings   over view   bere	Tri
Store r Store p Store n	ttern n times ange until pat attern2 until othing until p il user stop	pattern1 occurs	Occurrenc
<u> </u>	Replace	Insert before	Insert aft
Trigge	r definition		
1 FIND	PATTERN N TIMES	6	
Find		ccurrence of	
ADI	)R = o_boar	d/source/q.elf:ecs2.c:	149 Line #s
then	Trigger and f	fill memory	

2. Open the Intermodule window, and set up one logic analyzer to be armed by the trigger of the other logic analyzer.

8	Ind	ependent		Group	Run	Arming	; Tree	
	•			Group Run		-		
				4GHz Timing/1	lGHz	State	A armed	by
	<u>ج</u>	Independent	t					
		Group Run						
		46Hz Timin	e.1GH⊂ Sta	ete A				
		200Mvector/	/s Patterr	Generator B				
		2GSa/s Osc:	illoscope	С				
		333MHz Stat	te/2GHz Ti	mingZoom 2M S	Sampl	.e E 🔺		
		Motorola MM	PC800 Powe	rPC Emulator	1			
				Group	Run	Arming	Tree	
		10		Group Run				
				)				
				8				

3. Set up the other logic analyzer's trigger in terms of the arming signal it receives from the first analyzer.



Count time in each logic analyzer so that the captured data displays may be correlated.

- 4. Select the Group Run button to start the measurement.
- **Displaying the Data** 1. Use the Listing or Waveform display tools to view the data captured by each analyzer.

Expander Card (Master: E)					
Select					
16717A	Setup				
333MHz Sta	Waveform<1>				
	Timing Zoom<1>				
Listing(1) A					
Source Viewer<1>					
	Group Run 👂				

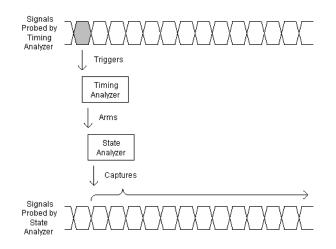
	State Number	PC	MPC821/86	0 Inverse Assembler	Time
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Relative
	-29	proc_specifi+03E4	li	r0 0000001	624.000 ns
	-25	proc_specifi+03E8	stb	r0 41CO(r12)	624.000 ns
G1_	-21	:MX_proc_specific		write 01	584,000 ns
	-20	proc_specifi+03EC	lwz	r0 0024(r1)	272.000 ns
	-16	proc_specifi+03F0	mtspr	lr r0	780.000 ns
	-12	proc_specifi+03F4	addi	r1 r1 0020	624.000 ns
	-8	proc_specifi+03F8	blr		624.000 ns
	-4	q.:ecs2:main+0048	b	q.elf:ecs2:main+0018	548.000 ns
tr_	0	q.:ecs2:main+0018	lis	r12 0000	544,000 ns
	4	q.:ecs2:main+001C	lwz	r3 41B0(r12)	624.000 ns
	8	/q.elf:num_checks		read 00xxxxxx	584.000 ns
	9	q:num_checks+0001		read 01	116.000 ns
	10	q:num_checks+0002		read 11xx	116.000 ns
	11	q:num_checks+0003		read B8	120.000 ns
	12	q.:ecs2:main+0020	bl	update:update_system	272.000 ns
<u>62.</u>	16	upd:update_system	mfspr	r0 lr	544,000 ns
	20	update_syste+0004	mr	r11 r1	624.000 ns



Search Got Label *CS1 Advanced sea	J Value I ± when Entering ↓ Next Prev
Seconds/div	
R/*W all	1 1
*MEM_OE all	
*MEM_WE all	1 1
*CS1 all	
data[0-3] all	

Use markers to correlate the data that was captured.

You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in both logic analyzers.



#### To arm a state machine with a timing machine trigger

Possible uses:

- To examine software execution when a timing violation occurs.
- To determine whether an incorrectly timed pulse is the result of a hardware defect or an incorrectly programmed counter.
- To capture software execution and correlate it with separate bus timing data or an I/O data stream.
- Probing the Target<br/>System1. Configure a state analysis machine (with an analysis probe) to capture<br/>software execution.

Expander Card (Master: E)
Select ->
16717A Setup
333MHz Sta Waveform<1>
Timing Zoom<1>
Listing<1>
Source Viewer<1>
Run 🖻
Sampling Format Trigger Symbol
Analyzer Name: Analyzer (E) I On 26Hz Timing Zoom
$\sim$ Timing Mode – Asynchronous sampling clocked internally by analyzer
State Mode - Synchronous sampling clocked by the Device Under Test

2. Select the state analyzer's clock input.

State Mode	Control:		
167 MHz /	2M Stat	∍ ₹	Trigger Position Center 👤
Acquisitio	n depth[	2M	<u>t</u>
Clock Set	ир ———		
Mode: Mas	ster onl	y 🛓	🔟 Advanced Clocking
Pod	E4 E3	E2	E1
Clock	ML	К	J
Activity	_ ‡	\$	<b>t</b>
Master	OFF OFF	0.	

3. Assign pods. Use one logic analyzer machine for analyzing the software execution. Create another logic analysis machine for analyzing bus timing data by specifying the Analyzer 2 type.

Sampling Format Pod Assignment		
	Pod Assignment	
Analyzer 1	Analyzer 2	
Name: MPC860 BUS	Name: MACHINE 2	
Type: State =	Type: Timing -	Unassigned Pods
E1: ++++++++++++++++++++++++++++++++++++	E3:+++++++++++++++++++++++++++++++++	D3:L _
E2: ************	E4: M _	D4:M _
D1: _**********		
D2: ++++++++++++++++++++++++++++++++++++		

4. Specify the sampling options for the second logic analyzer machine.

Sampling Format Trigger Symbol
Analyzer Name: MACHINE 2 🔽 🔽 On
◆ Timing Mode - Asynchronous sampling clocked internally by analyzer ◇ State Mode - Synchronous sampling clocked by the Device Under Test
Timing Mode Controls
2M Sample Full Channel 333 MHz 🛓 Trigger Position Center 🛓
Acquisition depth 2M
Sample Period 102.0ns

5. Format state analyzer labels for the signals that capture software execution.

Sampling Fo	rmat	Tri	igger   Symbol			
Pod Assignment	Cloc L. DE		Pod D2	Pod D1	Pod E2	
			TTL	TTL	TTL	
Setup/Hold	‡‡ KJ	‡‡ KJ	<b>‡‡‡‡‡‡‡‡‡‡‡</b> 15 87 0	_ <b>‡‡‡‡‡‡‡‡‡‡‡</b> _ <b>‡‡‡</b> 15 87 0	<b>************</b> _* 15 87 0	‡: 15
ADDR +					*****	**
DATA +		•••	*****	*****		•
STAT +	**	**	*	******		•
data[0-7] +		••	******			

6. Format timing analyzer labels for the signals that capture bus timing data.

Sampling Format Trigger	Gymbol	
Pod Assignment	Pod E4	Pod E3
100181110110111	TTL	TTL
Å	15 87 0	15 87 0
CAN_CLK +		*
CAN_TXD +		*
CAN_RXD +		*
*IRQ2 +		*

**Capturing the Data** 1. Set up the timing analyzer to trigger on the timing event of interest.

Sampling Format Trigger Symbol	
Trigger Functions Settings Overview Sta	atus]
General Timing	
Find pattern Find edge	
Find edge AND pattern Find width violation on pattern/pulse Find Nth occurrence of an edge	
Replace Insert before	
Replace Insert before	
Replace Insert before Trigger definition	

2. Set up the state analyzer to trigger on anything, immediately after it is armed.

Sampling   Format	Trigger Symbol	1
Trigger Functions Se General State	ettings Overview De	fault Storing Status S Trigger
Wait n external cloc Wait for arm in Wait for other machin Advanced - If/then Advanced - 2-way bran	ne to trigger	
Replace	Insert before	Insert after
	NT T before	Anything Anything MACHINE 2 triggers Timer Counter Flag ADDR DATA STAT data[0-7]

Trigger definition
1 If MACHINE 2 triggers And
ADDR = XXXXXXXX Hex
occurs 1 🛓 time
then Trigger and fill memory

Count time in the state analyzer so that the displays may be correlated.

3. Select the Run button to start the measurement.

**Displaying the Data** 1. Use the Waveform display to show the captured bus timing and use the Listing display to show the captured software execution.

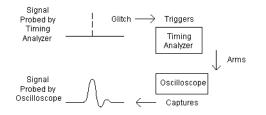
Expander Card (Master: E)	
Select     >     Image: 1       16717A     Slot E: Analyzer <e>       333MHz Sta     Slot E: Analyzer<e2>       Run</e2></e>	Setup
Seconds/div - 1.000 us	Delay 🕅 s 🖌
CAN_CLK allG1 CAN_TXD all 0 1 CAN_TXD all 0 1	
*IRQ2 all O	1



	State Number	PC	MPC821/86	0 Inverse Assembler	Time
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Relative
G1.	-20 -16	ext_exceptio+0104 _ext_exceptio+0108	lis lwz	r12 0000 r11 400C(r12)	820.000 ns 624.000 ns
	-12 -11 -10	/source/q.elf:pld ce/q.elf:pld+0001 ce/q.elf:pld+0002		read 40xxxxxx read 00 read 00xx	588.000 ns 116.000 ns 116.000 ns
	-9 -8	ce/q.elf:pld+0003 ext_exceptio+010C	li	read 00 ro 00002000	120.000 ns 120.000 ns 272.000 ns
<u>tr</u>	-4 0	ext_exceptio+0110 ABSOLUTE 40000004	sth	r0 0004(r11) write 20xx	624.000 ns 1.132 us
G2.	1 5	ext_exceptio+0114 g.elf:can:can_irg	bl mfspr		276.000 ns 544.000 ns
	9 13	<pre>:can:can_irq+0004 :can:can_irq+0008</pre>	stw stwu	r0 0004(r1) r1 FFF0(r1)	624.000 ns 784.000 ns
	17 21 25	:can:can_irq+000C :can:can_irq+0010 /source/q.elf:can	lis lwz	r12 0000 r11 4010(r12) read 30xxxxxx	780.000 ns 624.000 ns 588.000 ns
	26	ce/q.elf:can+0001		read 00	116.000 ns

You can use markers to correlate software execution to what was captured with the timing analyzer.

#### To arm an oscilloscope when the analyzer triggers



Possible uses:

• To view the analog parameters of a glitch captured by the logic analyzer.

#### Probing the Target System

- 1. Configure the logic analyzer.
  - 2. Format labels for the logic analyzer channels.
  - 3. Connect the oscilloscope channel probes to the signals of interest in the target system.
  - 4. Open the oscilloscope display, select the Channels tab, and set up the oscilloscope channels.

**Capturing the Data** 1. Set up the logic analyzer to trigger on the event of interest.

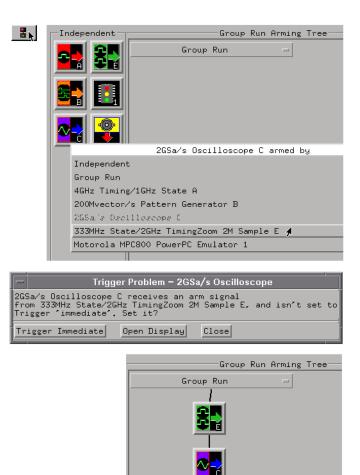


Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing Sta
General State Tr:
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop
Replace Insert before Insert aft
Trigger definition
I FIND PATTERN N TIMES
Find 1 occurrence of
ADDR = rd/source/q.elf;proc_spec.c: 209 Line #s
then Trigger and fill memory

If the logic analyzer is configured as a state analyzer, be sure to count time

so that the measurement displays can be correlated.

2. Open the Intermodule window, and set up the oscilloscope to be armed by the logic analyzer trigger.



3. Set up the oscilloscope to trigger immediately (after the arm).

2GSa/s Osci Calibr	/Display, /Display, ration orm<1> oup Run ►	Trigger		
	Trigger Setup – Scope <c></c>			
*				
Mode	- Sweet>			
⇔ Edge	🔶 Auto			
$\diamond$ Pattern	♦ Triggered			
◆ Immediate				
2GSa/s Oscilloscope will trigger immediately after it is armed. 2GSa/s Oscilloscope may be armed by the Run command or by another module in the system.				
Close Help				

4. Select the Group Run button to start the measurement.

### **Displaying the Data** 1. Use global markers to show the correlation between the logic analyzer trigger and the captured oscilloscope data.

Expander Card (Master: E)			
Select	->		
16717A	Setup		
333MHz Sta	Waveform<1>		
	Timing Zoom<1>		
	Listing<1> 🖌		
	Source Viewer(1)		
	Group Run		

Search	Goto   Markers   Comments   Analysis   Mixed Signal
Label *E	URST 🛓 Value 🗓 🛓 when Present 🛓 Next Prev
Advance	d searching Set G1 Set G2

	State Number	PC	MPC821/860 Inverse Assembler		Time	
	Decimal	Symbols	10=hex, 1	10.=decimal, %10=binary	Relative	
	-35	ce/q.elf:pld+0003		read 00	116.000 ns	
	-34	proc_specifi+02E0	li	r0 0000000	276.000 ns	
G1_	-30	proc_specifi+02E4	stb	r0 0006(r9)	624,000 ns	
	-26	ABSOLUTE 40000006		write 00	1.132 us	
G2.	-25	proc_specifi+02E8	lis	r12 0000	272,000 ns	
	-21	proc_specifi+02EC	lwz	r8 400C(r12)	620.000 ns	
	-17	/source/q.elf:pld		read 40xxxxxx	588.000 ns	
	-16	ce/q.elf:pld+0001		read 00	116.000 ns	
	-15	ce/q.elf:pld+0002		read 00xx	116.000 ns	
	-14	ce/q.elf:pld+0003		read 00	120.000 ns	
	-13	proc_specifi+02F0	lbz	r7 0006(r8)	272.000 ns	
	-9	ABSOLUTE 40000006		read FF	1.132 us	
	-8	proc_specifi+02F4	cmplwi	cr0 r7 0000	276.000 ns	
	-4	proc_specifi+02F8	beq	cr0 p:proc_specific+0	624.000 ns	
tr_	0	proc_specifi+02FC	nop		624.000 ns	
	4	proc_specifi+0300	lis	r12 0000	624.000 ns	
	8	proc_specifi+0304	lwz	r11 4350(r12)	624.000 ns	



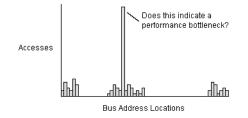
You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

See Also

"To make basic oscilloscope measurements" on page 11

### Making System Profile Measurements

- "To isolate the root cause of a performance bottleneck" on page 283
- "To simulate bus occupation and measure SW performance" on page 287



#### To isolate the root cause of a performance bottleneck

#### **Requirements:**

• This measurement requires the system performance analyzer (SPA) tool set.

#### Possible uses:

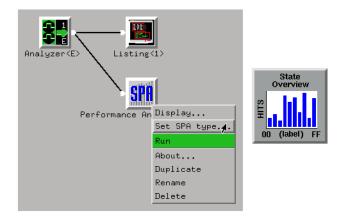
- To determine what's responsible for throughput bottlenecks.
- To detect which peripherals are most frequently used.
- To identify areas for performance improvements.
- To pinpoint regions of high memory activity.
- To measure program coverage.
- To measure stack usage.
- To isolate defects like invalid pointers.

Probing the Target1.Use an analysis probe to probe the microprocessor or standard bus whose<br/>performance you wish to analyze.

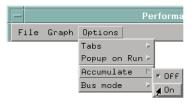
- 2. Load the configuration file included with the analysis probe to configure a state analysis machine.
- **Capturing the Data** 1. Set up a trigger specification to capture all bus cycles.

Sampling Format Trigger Symbol			
Trigger Functions Settings Overview Default			
General State			
Find pattern n times Store range until pattern occurs Store pattern2 until pattern1 occurs Store nothing until pattern occurs Run until user stop			
ReplaceInsert before			
Trigger definition			
1 FIND PATTERN N TIMES			
Find 1			
ADDR = XXXXXXXX Hex			
then Trigger and fill memory			
then Trigger and fill memory			

- 2. In the Workspace window, add the system performance analyzer to the measurement set up.
- 3. Use the system performance analyzer's State Overview display.



4. Set up the system performance analyzer to accumulate data and choose a repetitive run.

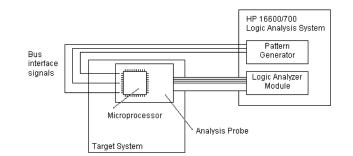


- 5. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.
- **Displaying the Data** 1. Expand buckets that show the most activity.

		)verview )DR			
2.954 M					
i t 1.477 M				Undo	
s				Auto	Sca
				Full	Sea
				Expai	nd∦E
0 0000000			FFFFF	FF	
	I	Data Values			
Display Information-		Bucket Inf	ormation		
	3,187,496		2,953,764 (92,67%)		
Total Acquisitions:	3		2,953,764 (92,67%)		
Total Display %: Displayed States:	100% 3,187,496	-	FD70A3DB-FFFFFFFF		
Base:		Min/Max:	FFF00500/FFF40334		
	Hex 🗆				

2. Select a bucket and use its information to correlate high activity to source code.

State Ove ADDR	rview				
П					
	Matching Symbols		124 Symbols		
	save_points sbrk	Function Function	FFF03984-FFF03AA7 FFF06AD0-FFF06BA3		
	set_outputs	Function	FFF0378C-FFF03897		
	sin sprintf	Function Function	FFF076FC-FFF07F53 FFF04BA4-FFF04C17		
	sprince	Function	FFF04B78-FFF04B9F		
	strlen	Function	FFF067BC-FFF067DF		
	strncpy	Function	FFF04CE0-FFF04D4B		
	<u></u>				
Dat	a Values	FFF08549			
	Bucket Information				
	Hits(%d): 411,682 (42) Hits(%t): 411,682 (39)	.57%)			
	Range: FFF03732-FFF Min/Max: FFF03732/FFF				
	Min/Max: FFF03/32/FFF	-03875			
	Se	ource Viewer<	15		
File Wind					
			=1		
╞╺╧╸┢╸					
Step Sou	rce   Goto In Listing	Browse S	ource   Text Search   S		
New Sou	rce File Name				
update			File Selec		
	File: /logic/demo/86	0_demo_board	l/source/update_sys.c		
202 void					
203 set_c 204 {	outputs(char *function	h, INI_8 tem	perature)		
205					
206 ME_set_outputs = 1;					
	207				
208 if (temperature <= target_temp) 209 {					
210 *function &= ~COOL; /* Cooling off */					
211 )					
212 if (temperature > target_temp+DEAD_BAND) 213 {					
214 *function I= COOL; /* Cooling on */					
<pre>215      } 216      if (temperature &gt;= target_temp) 217</pre>					
218					
219 220	220 if (temperature < target_temp-DEAD_BAND)				
221	221 { 222 *function I= HEAT; /* Heating on */				



### To simulate bus occupation and measure SW performance

#### Requirements:

- This measurement requires a pattern generator module (Agilent Technologies 16522A).
- This measurement requires the system performance analyzer (SPA) tool set.

Possible uses:

• To test how potential bus arbitration sequences can affect software performance.

### Probing the Target1. Connect pattern generator outputs to the appropriate bus interface<br/>signals.

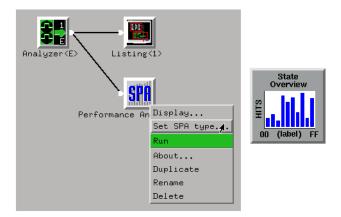
- 2. Configure the pattern generator to output the desired sequence of bus arbitration signals.
- 3. Typically, you will use an analysis probe to connect the logic analyzer to the microprocessor or standard bus, and you will use the provided configuration files to configure the analyzer and define labels.

### **Capturing the Data** 1. Set up the logic analyzer to capture all software execution as the pattern generator simulates bus arbitration sequences.

You may want to set up an intermodule measurement to coordinate the pattern generator stimulus and the logic analyzer's capture of the response.

2. In the Workspace window, add the system performance analyzer to the measurement set up.

3. Use the system performance analyzer's State Overview display.



- 4. Run the measurement (and, perhaps, stop the measurement if it's running repetitively) and view the results.
- **Displaying the Data** 1. Use the system performance analyzer's State Overview display to show which addresses have the most activity.

You may want to expand the buckets that have the most activity and look at the bucket information to see the source code that's responsible for the activity.

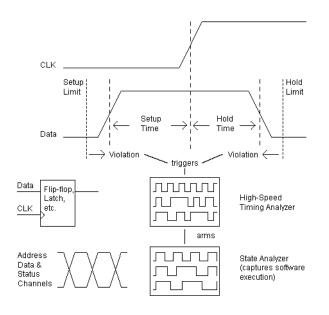
See Also "To generate pattern stimulus on devices" on page 75

"To isolate the root cause of a performance bottleneck" on page  $283\,$ 

"To generate patterns when a source line executes" on page  $262\,$ 

### **Isolating Critical Defects**

- "To capture SW execution on a setup or hold violation" on page 289
- "To trigger an oscilloscope when a source line executes" on page 294



## To capture SW execution on a setup or hold violation

Possible uses:

• To see how setup or hold violations affect software execution.

#### **Requirements:**

• The Agilent Technologies 16517A 4GHz Timing/1GHz State Logic Analyzer can look for setup and hold violations on multiple channels (for example, a data bus).

# Probing the Target<br/>System1. Connect the timing analyzer probes to the signals on which you are looking<br/>for a setup or hold violation.

- 2. Connect the state analyzer probes to the processor whose software execution you wish to capture. (Typically, you use an analysis probe to probe a processor.)
- 3. Configure the timing analyzer and format labels for the signals of interest.
- 4. Configure the state analyzer to capture software execution. (Typically, you use configuration files included with the analysis probe to configure and format labels.)

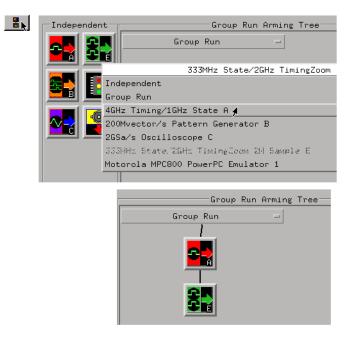
Chapter 1: Measurement Examples **System Integration** 

Capturing the Data1. Set up the timing analyzer to trigger on a setup or hold violation. (The<br/>Agilent Technologies 16517A 4GHz Timing/1GHz State Logic Analyzer<br/>includes a trigger function for capturing setup or hold violations.)

Select -> Setup
Format Trigger Skew Adjust Symbol Sample Period 500 ps Trigger Position Center
Title Macro Sequence
Find setup/hold, "D0_DEL" then TRIGGER clock: "edge1", Tsetup: 64 ns, Thold: 20.00ns
Pattern Edge
edge1 *MEM_WE

Count time in the logic analyzers so that the captured data may be correlated.

2. Open the Intermodule window, and set up the logic analyzer to be armed by the oscilloscope trigger.



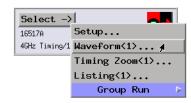
3. Set up the state analyzer to trigger on anything (after the arm).

# Chapter 1: Measurement Examples **System Integration**

Expand	er Card (Mas	ter: E)	
Select	->		
16717A	Setup 🖌		1
333MHz Sta	Waveform<1>	•••	1
	Timing Zoom	<1>	
	Listing<1>.	••	
	Source View	er<1>	
	Group	Run 🕨	
Sampli	ng   Format	Trigger	Symbol
Trigge	r Functions Se	ettings Ove	erview]Defau]
General	. State		erview Defau
General Find pa Store f Store f Store f	· · · ·	ttern occur pattern1 o	rs Docurs
General Find pa Store f Store f Store f	. State Attern n times range until pa pattern2 until nothing until	ttern occur pattern1 o	rs Docurs
General Find pa Store f Store f Store f	. State Attern n times range until pa pattern2 until nothing until	ttern occur patterni o pattern occ	rs Docurs
General Find p Store n Store n Run un	State Attern n times ange until pa pattern2 until nothing until til user stop	ttern occur patterni o pattern occ	rs occurs ours
General Find pr Store n Store n Run un L Trigge	State attern n times range until pa pattern2 until nothing until iil user stop Replace	ttern occur patterni o pattern occ	rs occurs ours
General Find P Store p Store n Run unt Trigge	State attern n times range until pa pattern2 until nothing until iil user stop Replace r definition	ttern occur patterni o pattern occ	rs occurs ours
General Find P Store p Store n Run unt Trigge	. State attern n times range until pa vattern2 until inting until til user stop Replace er definition FOR ARM IN the for arm in	ttern occur patterni o pattern occ	rs occurs ours

4. Select the Group Run button to start the measurement.

**Displaying the Data** 1. Use global markers to show the correlation between the timing analyzer trigger and the captured software execution.



Search   Got	o Markers Comments Analysis Mixed Signal	
G1: *CS1	. I Time I from Trigger I = -585.000 ns	
G2: *CS1	L = 1 Time L from Trigger L = 1.055 us	
Seconds/div	- [200.000 ns 🖌 Delay [255.000 ns 🖌	
	G1 tr G2	2
*MEM_WE all	1 1	
data[0-3] all	0904C0802A7	0
DO_DEL all		0



S	Search   Goto   Markers   Comments   Analysis   Mixed Signal					
G	G1: ★BURST ↓=0 Time ↓ from Trigger ↓=-976.000 ns					
G	G2:     ★BURST     ±     = 0     Time     ±     from     Trigger     ±     = 664,000 ns					
	State Number	PC	MPC821/860	) Inverse Assembler	ADDR	
	Decimal	Symbols	Symbols 10=hex, 10.=decimal, %10=binary Hex			
	-30	proc_specifi+03D0	lis	r20 0A00	FFF0449C	
	-26	proc_specifi+03D4	mtspr	ic_cst r20	FFF044A0	
	-22	proc_specifi+03D8	lis	r20 0C00	FFF044A4	
	-22 -18	proc_specifi+03D8 proc_specifi+03DC	lis mtspr	r20 0C00 ic_cst r20	FFF044A4 FFF044A8	
	-18	proc_specifi+03DC	mtspr	ic_cst r20	FFF044A8	
G1.	-18 -14	proc_specifi+03DC proc_specifi+03E0	mtspr lis	ic_cst r20 r12 0000	FFF044A8 FFF044AC	
G1.	-18 -14 -10	proc_specifi+03DC proc_specifi+03E0 proc_specifi+03E4	mtspr lis li	ic_cst r20 r12 0000 r0 00000001	FFF044A8 FFF044AC FFF044B0 FFF044B4 000041C0	
61. <u>tr</u>	-18 -14 -10 -6 -2 -1	proc_specifi+03DC proc_specifi+03E0 proc_specifi+03E4 proc_specifi+03E8 :MX_proc_specific proc_specifi+03EC	mtspr lis li	ic_cst r20 r12 0000 r0 00000001 r0 41C0(r12)	FFF044A8 FFF044AC FFF044B0 FFF044B4 000041C0 FFF044B8	
61. <u>tr</u> 62.	-18 -14 -10 -6 -2	proc_specifi+03DC proc_specifi+03E0 proc_specifi+03E4 proc_specifi+03E8 :MX_proc_specific	mtspr lis li stb	ic_cst r20 r12 0000 r0 00000001 r0 41C0(r12) write 01	FFF044A8 FFF044AC FFF044B0 FFF044B4 000041C0	

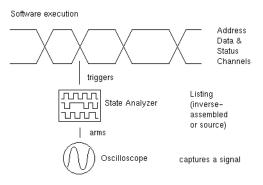
# Chapter 1: Measurement Examples System Integration

You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

You may want to open the Source Viewer window to view the source code associated with the timing analyzer trigger.

"To find setup and hold violations" on page 56

## To trigger an oscilloscope when a source line executes



Possible uses:

• To see the effect of a certain type of software execution on signals. (For example, do certain bus value changes cause ground bounce?)

Probing the Target1. Configure a state analysis machine (with an analysis probe) to capture<br/>software execution (pre-defined format is included with the analysis<br/>probe).

- 2. Connect the oscilloscope channel probes to the signals of interest in the target system.
- 3. Open the oscilloscope display, select the Channels tab, and set up the oscilloscope channels.

# Capturing the Data1. Use the Source Viewer to set up a trigger on the software execution of<br/>interest.

See Also

### Chapter 1: Measurement Examples System Integration

Step	Source Goto In Listing Browse Source Text Search Symbols
New	Source File Name
jpro	c_spec.c <u>+</u> <u>File Selection.</u>
Displa	ayed File: /logic/demo/860_demo_board/source/proc_spec.c
197	pld->gnd_bounce = 0x00; /* Change all bits from one to zero *
198	/* except D0. Correlate the scope *
199	/* measurement to this line of code *
200	/* and notice what happens to DO. *
201	
202	if (pld->gnd_bounce != 0x00)
203	(
204	/* The data has been corrupted due to a ground bounce */
205	/* phenomenon. If execution has reached this point, */
206	/* the data was corrupted and an error would occur in */
207	/* the software. */
208	
209	asm("nop") · /* Dummu instruction to trigger/break */ line # 209
210	} line # 209
211 212	/* Run a piece Trace before this line
212	7* Kun a piece
213	/* Trace about this line 4
214 215	/* The follwin /* running the Trace after this line
215 216	
216	/* logic analy Goto this line in listing before current state /* cycles are
217	7* cycles are Goto this line in listing after current state

### NOTE:

Source Viewer commands that set up triggers only modify the trigger condition. They do not modify the trigger position, storage qualifiers, else branch conditions, or other levels in the trigger sequence.

Count time in the logic analyzer so that the captured data may be correlated.

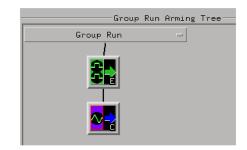
A dialog will inform you when the trigger has been set.

2. Open the Intermodule window, and set up the oscilloscope to be armed by the logic analyzer trigger.

# Chapter 1: Measurement Examples System Integration

Independent	Group Run Arming Tree
	Group Run 📼
	2GSa/s Oscilloscope C armed by
Independent	;
Group Run	
4GHz Timing	z∕1GHz State A
200Mvector/	's Pattern Generator B
265a.'s 0scl	lloscope C
333MHz Stat	:e∕2GHz TimingZoom 2M Sample E 🔺
Motorola MF	PC800 PowerPC Emulator 1

🗖 – Trigge	er Problem – 2GSa	/s Oscillosc	ope		
2GSa/s Oscilloscope ( from 333MHz State/2GA Trigger 'immediate'.	Hz TimingZoom 2M		and isn't	set	to
Trigger Immediate	Open Display	Close			



3. Set up the oscilloscope to trigger immediately (after the arm).

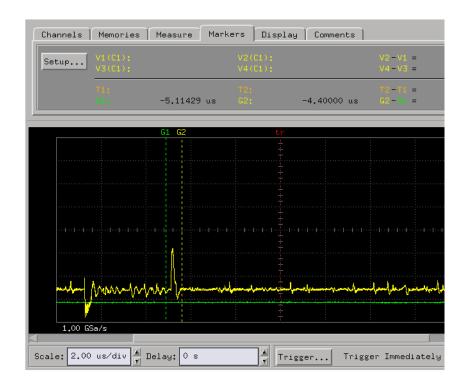
26Sa/s Osci Calibr	/Display, ▲ ration prm<1> oup Run ►	Trigger
F	Trigger Setu	p – Scope <c></c>
*		
Mode	- Sweet>	
🔷 Edge	🔶 Auto	
🔷 Pattern	♦ Triggered	
♦ Immediate		
it is armed. 2GSa/s Oscil.	·	rigger immediately after e armed by the Run command e system.
C1	.ose	Help

- 4. Select the Group Run button to start the measurement.
- **Displaying the Data** 1. Use global markers to show the correlation between the logic analyzer trigger and the captured oscilloscope data.

# Chapter 1: Measurement Examples **System Integration**

Expande	er Card (Master: E)		
Jerect			
16717A	Setup		
333MHz Sta	Waveform<1>		
	Timing Zoom<1>		
	Listing(1) 🖌		
	Source Viewer(1)		
	Group Run 🔰		

La	Search Goto Markers Comments Analysis Mixed Signal Label *BURST & Value & when Present & Next Prev Advanced searching Set G1 Set G2				
	State Number	PC	MPC821/86	50 Inverse Assembler 🏾 🏾	Time
	Decimal	Symbols	10=hex,	10.=decimal, %10=binary	Relative
	-35	ce/q.elf:pld+0003		read 00	116.000 ns
04	-34 -30	proc_specifi+02E0 proc_specifi+02E4	li stb	r0 00000000 r0 0006(r9)	276.000 ns 624.000 ns
G1_	-26	ABSOLUTE 40000006	sco	write 00	1.132 us
G2_	-25	proc_specifi+02E8	lis	r12 0000	272,000 ns
	-21	proc_specifi+02EC	lwz	r8 400C(r12)	620.000 ns
	-17	/source/q.elf:pld		read 40xxxxxx	588.000 ns
	-16	ce/q.elf:pld+0001		read 00	116.000 ns
	-15	ce/q.elf:pld+0002		read 00xx	116.000 ns
	-14	ce/q.elf:pld+0003		read 00	120.000 ns
	-13	proc_specifi+02F0	lbz	r7 0006(r8)	272.000 ns
	-9	ABSOLUTE 40000006		read FF	1.132 us
	-8	proc_specifi+02F4	cmplwi		276.000 ns
	-4	proc_specifi+02F8	beq	cr0 p:proc_specific+0	
<u>tr</u>	0	proc_specifi+02FC	nop		624,000 ns
	4	proc_specifi+0300	lis	r12 0000	624.000 ns
	8	proc_specifi+0304	lwz	r11 4350(r12)	624.000 ns



You can adjust the intermodule skew (in the Intermodule window) so that the relation between the markers and the trigger points are the same in the logic analyzer and in the oscilloscope.

See Also

"To make basic oscilloscope measurements" on page 11

# Measurement Tips & Tricks

- "Setting up 16715/16/17/18/19A triggers" on page 300
- "Setting up triggers in other logic analyzers" on page 302
- "Use trigger functions for easy measurement set up" on page 305
- "Modify trigger functions to build new measurements" on page 307
- "Know how processor execution affects measurements" on page 308
- "Getting the most out of trace memory" on page 309
- "If the trigger doesn't occur as expected" on page 309

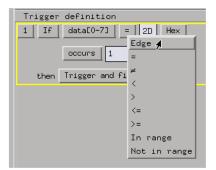
# Setting up 16715/16/17/18/19A triggers

In General...

- Use trigger functions for basic measurements.
- For more complicated measurements, where no trigger function exists, start with a trigger function that's similar to the measurement you want to make. Then, break down the trigger function and edit the advanced trigger sequence levels.

T	rigger definition	
1	ETND 2 EDGES TOO CLOSE	TOGETHER
	Insert LEVEL before	
	Insert LEVEL after	
	Replace LEVEL	L.
	Delete LEVEL	
	Cut LEVEL	s 🛓
	Copy LEVEL	
	Paste LEVEL before	l memory
	Paste LEVEL after	
	Expand function	
	Compress function	
	Breakdown function	

# Timing Analyzer<br/>Triggers• Everything that looks like a button in the trigger definition gives you a way<br/>to modify the trigger setup.



For example, to look for a edge instead of a pattern, select the button that equates a label with a pattern and choose an edge comparison instead.

State AnalyzerFor every state analysis sample, a logic analyzer needs to know two<br/>things:

- 1. Should some action (like a trigger) be taken as a result of this sample?
- 2. What should be done with this sample? That is, should it be stored in logic analyzer memory or should it be discarded? (This question doesn't need to be asked when using a timing analyzer because all samples are stored.)

State analysis trigger definitions are made simpler with a *default storage* qualifier. This makes it possible to ignore, at all trigger sequence levels, the question about what to do with the sample.

Of course, sometimes it's useful to specify storage qualifiers at certain levels in the trigger sequence. For this, you can insert storage *actions* in trigger definitions everywhere there is a trigger or goto action. Storage actions in the trigger definition override the default storage qualifier. Storage actions can also be used to turn on or off the default storing.

## Chapter 1: Measurement Examples Measurement Tips & Tricks

Sampling Format Trigger Symbol
Trigger Functions Settings Overview Default Storing Status Save/Recall Specify what to store by default. Using store actions in a sequence level overrides the "Default Storing". Store by default Nothing =
At start of acquisition, On - "Default Storing" is Trigger definition
1       FIND PATTERN N TIMES         Find       1         data[0-7]       =         2D       Hex         then       Trigger and fill memory
Insert ACTION     Store     Store sample       Goto     Timer     Don't store sample       Trigger     Counter     Turn on default storing       Flag     Reset occurrence counter     Turn off default storing

#### See also

"Modify trigger functions to build new measurements" on page 307

"Setting up triggers in other logic analyzers" on page 302

# Setting up triggers in other logic analyzers

There are differences in the way that triggers are set up between the Agilent Technologies 16715/16/17/18/19A logic analyzers and other logic analyzers.

Similarities

• In both types of logic analyzers, you are first given the choice of using trigger functions for trigger setup.

Sampling   Format	Trigger Symbol		
Trigger Functions	Pattern   Range   Tir	ner   Settings   Sa	ave/Recall
Find pattern2 occurr		Occurrence 1	Occurrence n
Replace	Insert before	Insert after	Delete
	ng "anystate" 1 occurrence of "a"		
Store "a	nystate"		

• Both types of logic analyzers also have a Settings tab for changing logic analyzer options and a Save/Recall tab for saving trigger setups.

# • Patterns, edges, ranges, and timers are set up under their own tabs in the older logic analyzers.

Sampling Format	Trigger	Symbol
Trigger Functions	Pattern	Range Timer Settings Save/Recall
a Label1	Hex	2456
b Label1	Hex	
c Label1	Hex	

Sampling Format Trigger	Symbol
Trigger Functions   Pattern	Range Timer Settings Save/Recall
range1 Label1 Hex	- 0000 ffff
range2 Label1 Hex	- 0000 ffff

Sampling   Format	Trigger	Symbol	1			
Trigger Functions	Pattern	Range	Timer	Settings	Save/Recall	
timer1 400ns	.▲ ∀					
timer2 400ns	A					
P						

All editing of the trigger setup happens in these tabs and in the trigger sequence level button menus.

While storing "anystate" 1   TRIGGER on "a" 1 time Edit4
Insert before 2 Insert after
Copy Delete Replace
Trigger Level
Trigger sequence step #1
User level - custom combinations, loops
While storing anystate
Trigger on a occurs 1 imes
Else on no state goto level 1 -
Timer control is not available in first macro
Close

See also "Setting up 16715/16/17/18/19A triggers" on page 300

# Use trigger functions for easy measurement set up

Many common measurement setups are provided with logic analyzers. These setups are called *trigger functions*, and you can use them for quick measurement setup.

You can use different trigger functions at different sequence step levels to combine them into a single measurement.

# To access trigger<br/>functions1. In the Trigger tab of a logic analyzer's Setup window, select the Trigger<br/>Functions tab.

2. Select a trigger function, and select the Replace, Insert Before, or Insert After button to move it to the trigger definition below.

Expander Card (Maste	er: E)		
Slot E HP 16717A 333MHz State/2GHz TimingZoom			
Listing	n<1> pom<1>		
	Run M		_
		7	
Trigger Functions Set	tings Overview Statu	· · · · · · · · · · · · · · · · · · ·	
General Timing		Irigger	function libraries
Find pattern absent f Run until user stop Find 2 ed <u>ges too clos</u> Find 2 edges too far Find pattern occurrin	e together apart	edge 1	edge 2
Replace 📐	Insert before	Insert after	Delete

3. In the trigger definition, specify the appropriate values and options, and select the Close button.

Trigger definition
1 FIND 2 EDGES TOO CLOSE TOGETHER
Find <u>R/*W</u> Edge <u>↑</u>
followed by AT2 Edge
occurring within 60 ns
then Trigger and fill memory

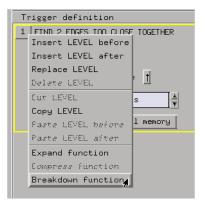
See also

"Modify trigger functions to build new measurements" on page 307

# Modify trigger functions to build new measurements

Break the trigger function down to see the measurement in terms of the logic analyzer resources:

1. In Trigger window, break down the function from the Modify menu



Note that in a timing analyzer, the sample period and occurrence counts can be used to measure time.

Trigger definition
1 If R/*W Edge 1
occurs $1 \xrightarrow{A}$ time
then Goto 2
2 If not AT2 Edge ↑ occurs 9 times eventually
then Goto 1
Else if AT2 Edge †
then Trigger and fill memory

Once broken-down, you can modify a trigger function.

	Know how processor execution affects measurements
Instruction Cache	When instruction caches are turned ON, a complete view of processor execution cannot be viewed at the pins of the processor.
	You can solve this problem by turning OFF instruction caches or by instrumenting your code (adding instructions that cause activity that can be viewed at the pins of the processor).
Chip Selects, MMUs, Paged Memory	When MMUs or paged memory moves code to different locations in memory, or when chip selects or reassigned address lines appear to change code addresses, symbol values are no longer accurate.
	The symbol tools require that a one-to-one mapping exists between physical and logical addresses.
	When chip selects reassign address lines, you may be able include them in the ADDR label specification to re-create a one-to-one mapping.
Word Alignment	When processors fetch multiple instructions (for example, byte instructions in a word fetch, or burst mode fetches), not all instruction addresses appear on the address bus. So, if you wanted to trigger on an instruction address, it might never be seen on the address bus.
	You can work around this problem by aligning symbols to the word (or burst) boundary or by manually setting the lower address bits to 0 or X (don't care) in the trigger specification.
	Note that the workaround could result in a trigger on an unexecuted instruction (for example, if the previous instruction causes an execution branch or jump).
Unexecuted Prefetches	In processors that have prefetch queues and/or instruction pipelines, some fetched instructions are not executed. And, you could trigger on an instruction that isn't executed.
	Most inverse assemblers will flag unexecuted instructions with "nu" or "-". Enhanced inverse assemblers let you filter away unexecuted instructions from view.

To prevent false triggering, you can add an offset to addresses in the trigger specification (where a capture of the offset address indicates execution at the previous address). Another (less practical) way to prevent false triggering is to add NOPs to code to account for prefetch depth and pipelines.

# Getting the most out of trace memory

	Your strategy for capturing the right amount of data depends on the amount of trace memory your logic analyzer has.
Using deep memory analyzers	If trace memory is deep, you can capture all execution and use the filtering tools to only display relevant data.
Using filter tools	One strategy when using deep memory analyzers is to use the filtering and display tools to look at different aspects of the captured data.
	For example, if you use the deep memory analyzer to capture all execution, you can use the Filter tool to isolate writes to a certain variable and the Chart display to track the values that were written. Or, you can use the Distribution display to see how the values vary.

Using storage If the data you're interested in appears infrequently and you are not qualifiers in the able to capture enough of it when all states are stored, you can use measurement storage qualifiers to store only the data you're interested in.

Using the context The context store feature stores events of interest plus the context store feature found in data before and after these events. These events plus their context some analyzers may occur far apart in time. There might be no way to capture a series of these events, even with very deep analyzers, unless they have context store.

# If the trigger doesn't occur as expected

You've set up a trigger specification that you believe will lead to a trigger, and when you run the measurement, the trigger doesn't occur. Or, you've set up a trigger specification that will only lead to a trigger in a error condition.

How do you tell what the logic analyzer is doing when the trigger doesn't occur? How do you know which parts of your trigger specification the logic analyzer has or hasn't seen?

There are a few strategies you can use when verifying or debugging sequence level programming:

- Look at the run status message line or open the Run Status window. It will tell you what level of the sequence the logic analyzer is in.
- Stop the measurement and look at the data that was captured. This is particularly useful when you use storage qualifiers to store "no states" (or only the states you're interested in) and the branches taken are stored.
- Save the trigger setup; then, simplify it to see what part of the sequence does get captured. When you learn what needs to be modified, you can recall the original trigger setup and make changes to it.

**absolute** Denotes the time period or count of states between a captured state and the trigger state. An absolute count of -10 indicates the state was captured ten states before the trigger state was captured.

**acquisition** Denotes one complete cycle of data gathering by a measurement module. For example, if you are using an analyzer with 128K memory depth, one complete acquisition will capture and store 128K states in acquisition memory.

**analysis probe** A probe connected to a microprocessor or standard bus in the device under test. An analysis probe provides an interface between the signals of the microprocessor or standard bus and the inputs of the logic analyzer. Also called a *preprocessor*.

**analyzer 1** In a logic analyzer with two *machines*, refers to the machine that is on by default. The default name is *Analyzer*<*N*>, where N is the slot letter.

**analyzer 2** In a logic analyzer with two *machines*, refers to the machine that is off by default. The default name is *Analyzer*<*N2*>, where N is the slot letter.

arming An instrument tool must be

armed before it can search for its trigger condition. Typically, instruments are armed immediately when *Run* or *Group Run* is selected. You can set up one instrument to arm another using the *Intermodule Window*. In these setups, the second instrument cannot search for its trigger condition until it receives the arming signal from the first instrument. In some analyzer instruments, you can set up one analyzer *machine* to arm the other analyzer machine in the *Trigger Window*.

**asterisk (\*)** See *edge terms*, *glitch*, and *labels*.

**bits** Bits represent the physical logic analyzer channels. A bit is a *channel* that has or can be assigned to a *label*. A bit is also a position in a label.

**card** This refers to a single instrument intended for use in the Agilent Technologies 16600A-series or 16700A/B-series mainframes. One card fills one slot in the mainframe. A module may comprise a single card or multiple cards cabled together.

**channel** The entire signal path from the probe tip, through the cable and module, up to the label grouping.

click When using a mouse as the

pointing device, to click an item, position the cursor over the item. Then quickly press and release the *left mouse button*.

**clock channel** A logic analyzer *channel* that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a *data channel*, except in the Agilent Technologies 16517A.

**context record** A context record is a small segment of analyzer memory that stores an event of interest along with the states that immediately preceded it and the states that immediately followed it.

**context store** If your analyzer can perform context store measurements, you will see a button labeled Context Store under the Trigger tab. Typical context store measurements are used to capture writes to a variable or calls to a subroutine, along with the activity preceding and following the events. A context store measurement divides analyzer memory into a series of context records. If you have a 64K analyzer memory and select a 16state context, the analyzer memory is divided into 4K 16-state context records. If you have a 64K analyzer memory and select a 64-state context, the analyzer memory will be

divided into 1K 64-state records.

**count** The count function records periods of time or numbers of state transactions between states stored in memory. You can set up the analyzer count function to count occurrences of a selected event during the trace, such as counting how many times a variable is read between each of the writes to the variable. The analyzer can also be set up to count elapsed time, such as counting the time spent executing within a particular function during a run of your target program.

**cross triggering** Using intermodule capabilities to have measurement modules trigger each other. For example, you can have an external instrument arm a logic analyzer, which subsequently triggers an oscilloscope when it finds the trigger state.

**data channel** A *channel* that carries data. Data channels cannot be used to clock logic analyzers.

**data field** A data field in the pattern generator is the data value associated with a single label within a particular data vector.

**data set** A data set is made up of all labels and data stored in memory of any single analyzer machine or

instrument tool. Multiple data sets can be displayed together when sourced into a single display tool. The Filter tool is used to pass on partial data sets to analysis or display tools.

#### debug mode See monitor.

**delay** The delay function sets the horizontal position of the waveform on the screen for the oscilloscope and timing analyzer. Delay time is measured from the trigger point in seconds or states.

**demo mode** An emulation control session which is not connected to a real target system. All windows can be viewed, but the data displayed is simulated. To start demo mode, select *Start User Session* from the Emulation Control Interface and enter the demo name in the *Processor Probe LAN Name* field. Select the *Help* button in the *Start User Session* window for details.

**deskewing** To cancel or nullify the effects of differences between two different internal delay paths for a signal. Deskewing is normally done by routing a single test signal to the inputs of two different modules, then adjusting the Intermodule Skew so that both modules recognize the signal at the same time.

**device under test** The system under test, which contains the circuitry you are probing. Also known as a *target system*.

**don't care** For *terms*, a "don't care" means that the state of the signal (high or low) is not relevant to the measurement. The analyzer ignores the state of this signal when determining whether a match occurs on an input label. "Don't care" signals are still sampled and their values can be displayed with the rest of the data. Don't cares are represented by the *X* character in numeric values and the dot (.) in timing edge specifications.

**dot (.)** See *edge terms*, *glitch*, *labels*, and *don't care*.

**double-click** When using a mouse as the pointing device, to double-click an item, position the cursor over the item, and then quickly press and release the *left mouse button* twice.

**drag and drop** Using a Mouse: Position the cursor over the item, and then press and hold the *left mouse button*. While holding the left mouse button down, move the mouse to drag the item to a new location. When the item is positioned where you want it, release the mouse button.

Using the Touchscreen: Position your finger over the item, then press and hold finger to the screen. While holding the finger down, slide the finger along the screen dragging the item to a new location. When the item is positioned where you want it, release your finger.

**edge mode** In an oscilloscope, this is the trigger mode that causes a trigger based on a single channel edge, either rising or falling.

**edge terms** Logic analyzer trigger resources that allow detection of transitions on a signal. An edge term can be set to detect a rising edge, falling edge, or either edge. Some logic analyzers can also detect no edge or a *glitch* on an input signal. Edges are specified by selecting arrows. The dot (.) ignores the bit. The asterisk (\*) specifies a glitch on the bit.

**emulation module** A module within the logic analysis system mainframe that provides an emulation connection to the debug port of a microprocessor. An E5901A emulation module is used with a target interface module (TIM) or an analysis probe. An E5901B emulation module is used with an E5900A emulation probe. **emulation probe** The stand-alone equivalent of an *emulation module*. Most of the tasks which can be performed using an emulation module can also be performed using an emulation probe connected to your logic analysis system via a LAN.

**emulator** An *emulation module* or an *emulation probe*.

**Ethernet address** See *link-level address*.

**events** Events are the things you are looking for in your target system. In the logic analyzer interface, they take a single line. Examples of events are *Label1* = *XX* and *Timer 1* > 400 *ns*.

**filter expression** The filter expression is the logical *OR* combination of all of the filter terms. States in your data that match the filter expression can be filtered out or passed through the Pattern Filter.

**filter term** A variable that you define in order to specify which states to filter out or pass through. Filter terms are logically OR'ed together to create the filter expression.

**Format** The selections under the logic analyzer *Format* tab tell the

logic analyzer what data you want to collect, such as which channels represent buses (labels) and what logic threshold your signals use.

**frame** The Agilent Technologies 16600A-series or 16700A/B-series logic analysis system mainframe. See also *logic analysis system*.

**gateway address** An IP address entered in integer dot notation. The default gateway address is 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.

**glitch** A glitch occurs when two or more transitions cross the logic threshold between consecutive timing analyzer samples. You can specify glitch detection by choosing the asterisk (\*) for *edge terms* under the timing analyzer Trigger tab.

**grouped event** A grouped event is a list of *events* that you have grouped, and optionally named. It can be reused in other trigger sequence levels. Only available in Agilent Technologies 16715A, 16716A, and 16717A logic analyzers.

held value A value that is held until

the next sample. A held value can exist in multiple data sets.

**immediate mode** In an oscilloscope, the trigger mode that does not require a specific trigger condition such as an edge or a pattern. Use immediate mode when the oscilloscope is armed by another instrument.

**interconnect cable** Short name for *module/probe interconnect cable*.

**intermodule bus** The intermodule bus (IMB) is a bus in the frame that allows the measurement modules to communicate with each other. Using the IMB, you can set up one instrument to *arm* another. Data acquired by instruments using the IMB is time-correlated.

**intermodule** Intermodule is a term used when multiple instrument tools are connected together for the purpose of one instrument arming another. In such a configuration, an arming tree is developed and the group run function is designated to start all instrument tools. Multiple instrument configurations are done in the Intermodule window.

**internet address** Also called Internet Protocol address or IP address. A 32-bit network address. It

is usually represented as decimal numbers separated by periods; for example, 192.35.12.6. Ask your LAN administrator if you need an internet address.

**labels** Labels are used to group and identify logic analyzer channels. A label consists of a name and an associated bit or group of bits. Labels are created in the Format tab.

**line numbers** A line number (Line #s) is a special use of *symbols*. Line numbers represent lines in your source file, typically lines that have no unique symbols defined to represent them.

**link-level address** Also referred to as the Ethernet address, this is 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.

**local session** A local session is when you run the logic analysis system using the local display connected to the product hardware.

**logic analysis system** The Agilent Technologies 16600A-series or 16700A/B-series mainframes, and all tools designed to work with it. Usually used to mean the specific system and tools you are working with right now.

**machine** Some logic analyzers allow you to set up two measurements at the same time. Each measurement is handled by a different machine. This is represented in the Workspace window by two icons, differentiated by a 1 and a 2 in the upper right-hand corner of the icon. Logic analyzer resources such as pods and trigger terms cannot be shared by the machines.

**markers** Markers are the green and yellow lines in the display that are labeled x, o, G1, and G2. Use them to measure time intervals or sample intervals. Markers are assigned to patterns in order to find patterns or track sequences of states in the data. The x and o markers are local to the immediate display, while G1 and G2 are global between time correlated displays.

**master card** In a module, the master card controls the data acquisition or output. The logic analysis system references the module by the slot in which the master card is plugged. For example, a 5-card Agilent Technologies 16555D

would be referred to as *Slot C: machine* because the master card is in slot C of the mainframe. The other cards of the module are called *expansion cards*.

**menu bar** The menu bar is located at the top of all windows. Use it to select *File* operations, tool or system *Options*, and tool or system level *Help*.

**message bar** The message bar displays mouse button functions for the window area or field directly beneath the mouse cursor. Use the mouse and message bar together to prompt yourself to functions and shortcuts.

#### module/probe interconnect cable

The module/probe interconnect cable connects an E5901B emulation module to an E5900B emulation probe. It provides power and a serial connection. A LAN connection is also required to use the emulation probe.

**module** An instrument that uses a single timebase in its operation. Modules can have from one to five cards functioning as a single instrument. When a module has more than one card, system window will show the instrument icon in the slot of the *master card*.

**monitor** When using the Emulation Control Interface, running the monitor means the processor is in debug mode (that is, executing the debug exception) instead of executing the user program.

**panning** The action of moving the waveform along the timebase by varying the delay value in the Delay field. This action allows you to control the portion of acquisition memory that will be displayed on the screen.

**pattern mode** In an oscilloscope, the trigger mode that allows you to set the oscilloscope to trigger on a specified combination of input signal levels.

**pattern terms** Logic analyzer resources that represent single states to be found on labeled sets of bits; for example, an address on the address bus or a status on the status lines.

**period (.)** See *edge terms*, *glitch*, *labels*, and *don't care*.

**pod pair** A group of two pods containing 16 channels each, used to physically connect data and clock signals from the unit under test to the analyzer. Pods are assigned by pairs in the analyzer interface. The number of pod pairs avalaible is determined

by the channel width of the instrument.

pod See pod pair

**point** To point to an item, move the mouse cursor over the item, or position your finger over the item.

preprocessor See analysis probe.

**primary branch** The primary branch is indicated in the *Trigger sequence step* dialog box as either the *Then find* or *Trigger on* selection. The destination of the primary branch is always the next state in the sequence, except for the Agilent Technologies 16517A. The primary branch has an optional occurrence count field that can be used to count a number of occurrences of the branch condition. See also *secondary branch*.

**probe** A device to connect the various instruments of the logic analysis system to the target system. There are many types of probes and the one you should use depends on the instrument and your data requirements. As a verb, "to probe" means to attach a probe to the target system.

**processor probe** See *emulation probe*.

**range terms** Logic analyzer resources that represent ranges of values to be found on labeled sets of bits. For example, range terms could identify a range of addresses to be found on the address bus or a range of data values to be found on the data bus. In the trigger sequence, range terms are considered to be true when any value within the range occurs.

**relative** Denotes time period or count of states between the current state and the previous state.

**remote display** A remote display is a display other than the one connected to the product hardware. Remote displays must be identified to the network through an address location.

**remote session** A remote session is when you run the logic analyzer using a display that is located away from the product hardware.

**right-click** When using a mouse for a pointing device, to right-click an item, position the cursor over the item, and then quickly press and release the *right mouse button*.

**sample** A data sample is a portion of a *data set*, sometimes just one point. When an instrument samples the target system, it is taking a single

measurement as part of its data acquisition cycle.

**Sampling** Use the selections under the logic analyzer Sampling tab to tell the logic analyzer how you want to make measurements, such as State vs. Timing.

**secondary branch** The secondary branch is indicated in the *Trigger sequence step* dialog box as the *Else on* selection. The destination of the secondary branch can be specified as any other active sequence state. See also *primary branch*.

**session** A session begins when you start a *local session* or *remote session* from the session manager, and ends when you select *Exit* from the main window. Exiting a session returns all tools to their initial configurations.

**skew** Skew is the difference in channel delays between measurement channels. Typically, skew between modules is caused by differences in designs of measurement channels, and differences in characteristics of the electronic components within those channels. You should adjust measurement modules to eliminate as much skew as possible so that it does not affect the accuracy of your measurements.

**state measurement** In a state measurement, the logic analyzer is clocked by a signal from the system under test. Each time the clock signal becomes valid, the analyzer samples data from the system under test. Since the analyzer is clocked by the system, state measurements are *synchronous* with the test system.

**store qualification** Store qualification is only available in a *state measurement*, not *timing measurements*. Store qualification allows you to specify the type of information (all samples, no samples, or selected states) to be stored in memory. Use store qualification to prevent memory from being filled with unwanted activity such as noops or wait-loops. To set up store qualification, use the *While storing* field in a logic analyzer trigger sequence dialog.

**subnet mask** A subnet mask blocks out part of an IP address so that 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. Ask your LAN administrator if you need a the subnet mask for your network.

**symbols** Symbols represent patterns and ranges of values found on labeled sets of bits. Two kinds of symbols are available:

- 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.
- User-defined symbols Symbols you create.

Symbols can be used as *pattern* and *range* terms for:

- Searches in the listing display.
- Triggering in logic analyzers and in the source correlation trigger setup.
- Qualifying data in the filter tool and system performance analysis tool set.

**system administrator** The system administrator is a person who manages your system, taking care of such tasks as adding peripheral devices, adding new users, and doing system backup. In general, the system administrator is the person you go to with questions about implementing your software. **target system** The system under test, which contains the microprocessor you are probing.

**terms** Terms are variables that can be used in trigger sequences. A term can be a single value on a label or set of labels, any value within a range of values on a label or set of labels, or a glitch or edge transition on bits within a label or set of labels.

**TIM** A TIM (Target Interface Module) makes connections between the cable from the emulation module or emulation probe and the cable to the debug port on the system under test.

**time-correlated** Time correlated measurements are measurements involving more than one instrument in which all instruments have a common time or trigger reference.

timer terms Logic analyzer resources that are used to measure the time the trigger sequence remains within one sequence step, or a set of sequence steps. Timers can be used to detect when a condition lasts too long or not long enough. They can be used to measure pulse duration, or duration of a wait loop. A single timer term can be used to delay trigger until a period of time after detection of a significant event.

timing measurement In a timing measurement, the logic analyzer samples data at regular intervals according to a clock signal internal to the timing analyzer. Since the analyzer is clocked by a signal that is not related to the system under test, timing measurements capture traces of electrical activity over time. These measurements are *asynchronous* with the test system.

**tool icon** Tool icons that appear in the workspace are representations of the hardware and software tools selected from the toolbox. If they are placed directly over a current measurement, the tools automatically connect to that measurement. If they are placed on an open area of the main window, you must connect them to a measurement using the mouse.

**toolbox** The Toolbox is located on the left side of the main window. It is used to display the available hardware and software tools. As you add new tools to your system, their icons will appear in the Toolbox.

**tools** A tool is a stand-alone piece of functionality. A tool can be an instrument that acquires data, a display for viewing data, or a postprocessing analysis helper. Tools are represented as icons in the main window of the interface. trace See acquisition.

**trigger sequence** A trigger sequence is a sequence of events that you specify. The logic analyzer compares this sequence with the samples it is collecting to determine when to *trigger*.

**trigger specification** A trigger specification is a set of conditions that must be true before the instrument triggers.

**trigger** Trigger is an event that occurs immediately after the instrument recognizes a match between the incoming data and the trigger specification. Once trigger occurs, the instrument completes its *acquisition*, including any store qualification that may be specified.

**workspace** The workspace is the large area under the message bar and to the right of the toolbox. The workspace is where you place the different instrument, display, and analysis tools. Once in the workspace, the tool icons graphically represent a complete picture of the measurements.

**zooming** In the oscilloscope or timing analyzer, to expand and contract the waveform along the time base by varying the value in the s/Div

field. This action allows you to select specific portions of a particular waveform in acquisition memory that will be displayed on the screen. You can view any portion of the waveform record in acquisition memory.

#### **Numerics**

16715A triggers, setting up, 300 16716A triggers, setting up, 300 16717A triggers, setting up, 300 16718A triggers, setting up, 300 16719A triggers, setting up, 300

#### A

accessing memory, 137 accumulate mode in SPA tool set. 88 activity, bus, 174 address alignment, 199 address lines, reassigned, 308 address offset, 199 address range, execution leaves, 28 aligning symbols, 308 alignment, address, 199 analog domain, 11 analysis probe (standard bus), 174 application software developers, 2 arm, 271, 277 automate emulation probe commands, 140

#### B

bandwidth, evaluating, 131 boot code, 152 boot code, downloading, 144 boot code, testing, 143 bottleneck, performance, 283 break down trigger functions, 307 breakpoints, capturing execution between, 223 breakpoints, stop processor execution using, 149 buckets, time, 88 bus activity, 174 bus arbitration sequences, 287 bus contention, 40, 72 bus mode in SPA tool set, 88 bus occupation, 131

bus occupation, simulating, 287 bus stability, 88 bus transactions, 88

#### $\mathbf{C}$

cache (instruction) ON, 175 cache, instruction, 175 cache, secondary, 116 callers of a function, 206 chip selects, 308 code (boot), downloading, 144 code download, initializing registers before, 137 code instrumentation, 175, 308 communications channel, 40, 72 Compare tool, 51 component stress conditions, 51 conformance to specifications, 51 conformance to specifications. measuring, 50 consecutive sequence of events. 105 context store, 206 context store on variable accesses. 236 control signals, 32 correlated data, 266 correlated displays, 45, 165, 174, 258, 262, 271, 294 correlated source viewer, 236 corruption, stack or heap, 251 coverage measurements, 283 critical defects, isolating, 288 cross-domain measurements, 257 crosstalk, 11 cycles, extra, 116 cycles, monitor, 223

#### D

data from remote device, 59 data request interrupt, 116 data setup specifications, 36 data strobe, 88 debug port, 137 debugger breakpoints, 226 debugger breakpoints, capturing execution between. 223 debugger, third party, 137 deep memory analyzers, 309 defect isolation, 283 download boot code, 144 download, initializing registers before, 137 DRAM row/column address strobes, 20 driver development, 155 driver execution, 165 driver writers, 2 droop, 11

## Е

edges (signal), 16 edges too close, 20 edges too far, 20 emulation control tool set, 137 emulation probe as test tool, 140 end of packet, 177 enhanced inverse assemblers, 308 entry and exit, function, 214 event occurrence rates, 192 event, store N samples, 100 events (SPA), viewing, 88 events, Nth occurrence, 96 events, sequence of, 105 exception, routine that causes, 206 execution (processor), how measurements are affected by, 308 execution (processor), start/stop, 147 execution (processor), stop using breakpoints, 149 execution leaves address range, 28 execution order, 203

# Index

execution time, 186 execution time, function, 214 execution time, software, 218 execution, driver, 165 execution, window of, 210 exercising the microprocessor, 136 extra cycles, 116

#### F

failure or halt, capturing execution up to, 171 fall time, 11 filter, 111 filter tools, 309 Find Packet trigger function, 177 firmware developers. 2 firmware development, 2, 143 flip-flops, 56 frequency of interrupts, 183 function entry and exit, 214 function execution time, 214, 218 function statement, 210 function, callers of, 206 functions (trigger), modifying, 307 functions, trigger, 305

#### G

glitch/edge resource, 36 glitches, 45 glitches (signal), 16 global markers, 214 global variable accesses, context store, 236 ground bounce, 11 guarded memory, 255

#### H

handshake violation, 67 hardware breakpoints, 149 hardware designers, 2 hardware turn-on, 2, 10 heap corruption, 251 heap overflow, 255 heap usage, 251 holding off trigger, 32

#### I

I/O accesses, 140 I/O read, context store, 236 initializing registers, 137 initializing registers before download, 137 instability, bus, 88 instruction cache, 308 instruction cache ON, 175 instruction pipelines, 308 instrumenting code, 175, 308 intermodule skew. 266 interrupt exceptions, 206 interrupt frequency and type, 183 interrupt latency, 186 interrupt loading, 192 interrupt response time, 59 interrupt routine, 116 interrupt sequence simulation, 191 interrupt service routines, 183 interrupt, data request, 116 invalid pointers, 283 inverse assemblers, 308 isolating defects, 283

# J

jitter, 81

## K

known-good circuitry, 51

## L

latches, 56 latency, interrupt, 186 loop (program) exit, 111

#### M

main system help, 2 markers, 20, 271 markers, global, 214 measurements, how processor execution affects, 308 measurements, parametric, 11 measuring software performance, 287 measuring time, 28 memory (trace), 100, 120, 210 memory accesses, 140 memory activity, 283 memory elements, 56 memory refresh routine, 120 memory select line, 36 memory usage, 250 memory write cycles, 120 memory, accessing, 137 memory, consecutive reads, 116 memory, guarded or non-existent, 255 memory, specific write to, 96 memory, stack or heap, 251 memory, trace, 309 memory, writes to consecutive locations, 105 microprocessor on-chip debug circuitry, 137 microprocessor, exercising the, 136 MMUs, 308 modifying trigger functions, 307 monitor cycles, omitting from trace. 223 monitor loop, 111 monitor variable values, 231 multi-processor systems, 266

#### Ν

network protocol decoder inverse assembler, 177

# Index

network switching systems, 177 noise, 11 non-consecutive sequence of events, 105 non-existent memory, 255 NOPs, adding to code, 308 Nth occurrence of an event, 96 Nth transition, 24 NULL pointer de-references, 229

#### 0

occurrence rate of events, 192 offset, address, 199 offsets, adding to trigger spec. addresses, 308 omitting monitor cycles, 223 on-chip (microprocessor) debug circuitry, 137 OS calls, 175 oscilloscope measurements, 11 out of range, variable, 240 overshoot, 11

#### Р

packet data, 177 packet data, triggering on, 177 paged memory, 308 parallel, serial data to, 160 parametric measurements, 11 pattern generation, 75 pattern generator, 191, 287 pattern generator, starting on source line, 262 pattern resource, 32, 36 pattern stops, 28 pattern, stable, 16 patterns (signal), 16 performance bottleneck, 283 peripheral usage, 283 physical layer protocol, 177 pipelines (instruction), 308 PLD control signal violation, 67 pointers, invalid, 283 prefetches, unexecuted, 308 processor execution, how measurements are affected by, 308 processor execution, start/stop, 147 processor execution, stop using breakpoints, 149 profile, system, 282 program coverage measurements, 283 program execution, window of, 210 program flow, 125 program loop exit, 111 program messages, 175 protocol violation, 116 pulse, 271 pulse width, 11 pulse width specifications, 63

#### R

Real-Time OS (RTOS), 175 reassigned address lines, 308 reference buffer. 51 refresh (memory) routine, 120 register accesses, 140 registers, initializing, 137 registers, initializing before download, 137 ringing, 11 rise, 11 rise time, 11 root cause of noise, crosstalk, or ground bounce, 11 root cause of performance bottleneck. 283 routine executes before another, 203 routine that causes task or exception, 206 routine, memory refresh, 120

#### S

scope measurements, 11 searching for states with markers, 214 secondary cache, 116 sensors, frequency of data acquired from, 192 sequence of events, 105 serial data to parallel, 160 serial pattern, 155 setting up 16715A triggers, 300 setting up 16716A triggers, 300 setting up 16717A triggers, 300 setting up 16718A triggers, 300 setting up 16719A triggers, 300 setting up triggers, 302 setup or hold violation. 56 setup or hold violation, capturing software execution at, 289 signal edges, 16 signal inactivity, 28 signal parameters, 11 signal patterns, 16 simulate bus occupation, 287 simulating interrupt sequences, 191 software breakpoints, 149 software code analysis, 198 software developers, 2 software development, 2, 198 software execution and standard bus execution, 266 software execution time, 218 software execution, capturing at oscilloscope trigger, 258 software performance, measuring, 287 source line, stopping execution at, 226 source line, trace about, 199 source line, trigger oscilloscope on, 294 source viewer, 206

# Index

SPA displays, State Overview, 283 SPA displays, Time Overview, 192 SPA events, viewing, 88 specifications, conformance to, 51 specifications, measuring conformance, 50 stability, bus, 88 stable pattern, 16 stack corruption, 251 stack frame, 203 stack overflow, 255 stack usage, 251, 283 standard bus, 174 standard bus execution and software execution, 266 start of packet, 177 start processor execution, 147 startup execution, 152 state events, 96 state machine, 271 State Overview SPA display, 283 stimulus generation, 75 stop processor execution, 147 stop processor execution using breakpoints, 149 storage qualifiers, 309 store N samples, 100 stress conditions (component), 51 strobe signals, 88 subroutine exits prematurely, 116 subroutine, time critical, 116 symbols, aligning, 308 system crash, capturing execution up to, 171 system integration, 2, 257 system integrators, 2 system profile, 282

#### Т

task, routine that causes, 206 test tool, emulation probe, 140 third-party debugger, 137 throughput bottlenecks, 283 time dispersion, 81 Time Overview SPA display, 192 time, interrupt execution, 186 time, measuring, 28 time, software execution, 218 time-correlated data, 266 time-correlated displays, 45, 165, 174, 258, 262, 271, 294 timing machine, 271 timing violation, 271 TimingZoom display, 45 tips and tricks, 300 too close, events that occur. 116 too far, events that occur, 116 trace memory, 100, 120, 210, 309 trace, omitting monitor cycles, 223 transactions, bus, 88 trigger doesntoccur', 309 trigger functions, 302, 305 trigger functions, modifying, 307 trigger, holding off, 32 triggering on network packet data, 177 triggers, setting up, 302 triggers, setting up 16715A, 300 triggers, setting up 16716A, 300 triggers, setting up 16717A, 300 triggers, setting up 16718A, 300 triggers, setting up 16719A, 300 type of interrupts, 183

#### U

unexecuted prefetches, 308

#### V

variable accesses, context store, 236 variable corruption, 240 variable equals value, break execution when, 245 variable out of range, 240 variable values, monitoring, 231 variable, writers of, 236 variables, analyzing, 229 view SPA events, 88 VisiTrigger capabilities, 177

#### W

window of program execution, 210 word alignment, 308 write cycles, 120 writers of a variable, 236