

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

Artisan Technology Group
(217) 352-9330 | sales@artisantg.com | artisantg.com

Full-service, independent repair center
with experienced engineers and technicians on staff.

We buy your excess, underutilized, and idle equipment
along with credit for buybacks and trade-ins.

Custom engineering
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

Expert team | Trust guarantee | 100% satisfaction

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

Find the *Keysight / Agilent 16550A* at our website: ***Click HERE***

Help Volume

© 1992-2002 Agilent Technologies. All rights reserved.

Instrument: Agilent Technologies 16550A Logic Analyzer

Agilent Technologies 16550A 100 MHz State/500 MHz Timing Logic Analyzer



The Agilent Technologies 16550A provides timing analysis at 500 MHz in half-channel mode. In state mode, the analyzer can be clocked by an external sample clock at a rate of 100 MHz. Memory depth is 4 KB per channel in full-channel modes or 8 KB per channel in half channel mode.

Getting Started

- "Analyzer Probing Overview" on page 107
- "Setting Up a Measurement" on page 10
- "When Something Goes Wrong" on page 33
- "Error Messages" on page 33

Measurement Examples

- "Making a Basic Timing Measurement" on page 22
- "Making a Basic State Measurement" on page 18
- Advanced Measurement Examples (see the *Measurement Examples* help volume)
- "Interpreting the Data" on page 26

More Features

"Coordinating Measurements" on page 31

Using Inverse Assembly (see the *Listing Display Tool* help volume)

"Using Symbols" on page 110

Using Markers (see the *Markers* help volume)

"Loading and Saving Logic Analyzer Configurations" on page 32

"Testing the Logic Analyzer Hardware" on page 39

Interface Reference

"The Sampling Tab" on page 40

"The Format Tab" on page 55

"The Trigger Tab" on page 75

"The Symbols Tab" on page 119

"Specifications and Characteristics" on page 103

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

Glossary of Terms (see page 131)

Contents

Agilent Technologies 16550A 100 MHz State/500 MHz Timing Logic Analyzer

1 Agilent Technologies 16550A 100 MHz State/500 MHz Timing Logic Analyzer

Setting Up a Measurement 10
Connect the Analyzer to the Target System 10
Define the Type of Measurement 11
Set Up the Bus Labels 13
Define Trigger Conditions 14
Run the Measurement 15
Examine the Data 16

Making a Basic State Measurement 18

Making a Basic Timing Measurement 22

Interpreting the Data 26
Analysis Using Waveform 26
Analysis Using Listing 28

Coordinating Measurements 31

Loading and Saving Logic Analyzer Configurations 32

When Something Goes Wrong 33
Interference with Target System 33
Error Messages 33
Nothing Happens 38
Suspicious Data 38

Testing the Logic Analyzer Hardware 39

Contents

The Sampling Tab 40
Acquisition Depth 40
Setting the Acquisition Mode 41
Performing Clock Setup (State only) 41
Naming the Analyzer 44
Turning the Analyzer Off 45
Sample Period (Timing Only) 45
Trigger Position Control 53

The Format Tab 55

Importing Netlist and ASCII Files 56
Exporting ASCII Files 58
Importing ASCII Files 58
Termination Adapter 60
E5346A High Density Adapter 61
Mapping Connector Names 62
Import the Net List File 62
Verify Net to Label Mapping 63
Select/Create Interface Labels 64
Activity Indicators 64
Assigning Pods to the Analyzers 65
Data On Clocks Display 66
To reorder bits in a label 69
Labels: Mapping Analyzer Channels to Your Target 70
Setting Up the Pod Clock 70
Pod Selection 71
Setting the Pod Threshold 72
State Clock Setup/Hold (State only) 73

Contents

The Trigger Tab 75

Understanding Logic Analyzer Triggering 76
Setting Up a Trigger 78
Inserting and Deleting Sequence Steps 79
Editing Sequence Steps 80
Setting Up Loops and Jumps in the Trigger Sequence 81
Saving and Recalling Trigger Sequences 82
Clearing Part or All of the Trigger 83
Overview of the Trigger Sequence 83
Trigger Functions 84
Working with the User-level Function 91
Defining Resource Terms 94
Tagging Data with Time or State Tags (State Only) 101
Arming Control 101

Specifications and Characteristics 103

What is a Specification 103
What is a Characteristic 103
What is a Calibration Procedure 104
What is a Function Test 104
Agilent Technologies 16550A Logic Analyzer Specifications 104
Agilent Technologies 16550A Logic Analyzer Characteristics 105

Analyzer Probing Overview 107

Using Symbols 110

To load object file symbols 111
To adjust symbol values for relocated code 112
To create user-defined symbols 113
To enter symbolic label values 114
To create an ASCII symbol file 115
To create a readers.ini file 116

Contents

The Symbols Tab 119
Symbols Selector Dialog 121
Symbol File Formats 123
General-Purpose ASCII (GPA) Symbol File Format 124

Glossary

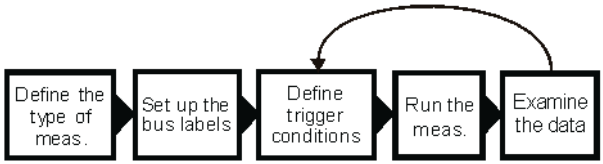
Index

1

Agilent Technologies 16550A 100 MHz State/500 MHz Timing Logic Analyzer

Setting Up a Measurement

After you have connected the logic analyzer probes to your target system, (see page 10) there are five basic steps for any measurement.



1. "Define the Type of Measurement" on page 11
2. "Set Up the Bus Labels" on page 13
3. "Define Trigger Conditions" on page 14
4. "Run the Measurement" on page 15
5. "Examine the Data" on page 16
   Refine measurement by repeating steps 3 - 5.

If you load a configuration file, it will set up the logic analyzer and trigger. For your particular measurement, you may need to change some settings.

See Also

"Making a Basic Timing Measurement" on page 22

"Making a Basic State Measurement" on page 18

Measurement Examples (see the *Measurement Examples* help volume)

Making Basic Measurements for a self-paced tutorial

Connect the Analyzer to the Target System

Before you begin setting up a measurement, you need to physically connect the logic analyzer to your target system. Attach the pods in a way that keeps logically related *channels* together and be sure to

ground each pod. *Analysis probes*, available for most common microprocessors, can simplify the connection process.

The logic analyzer pods carry the signals to the logic analyzer from your target system. Connect the pods either directly to the target system or to an analysis probe. You can attach the pods either directly to a 40-pin header, to a 20-pin header with an adapter, or use the General Purpose Probes to attach to individual channels.

If you are using an analysis probe, Setup Assistant will guide you through the process based on your logic analyzer and the analysis probe.

Step 1: Describe the Measurement (see page 11)

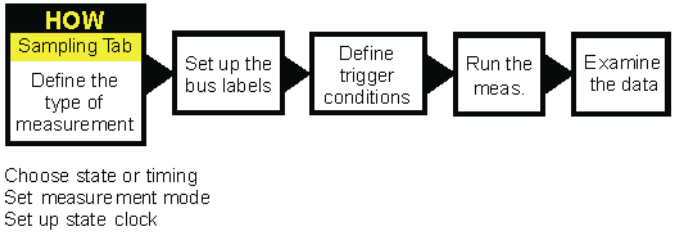
See Also

"Analyzer Probing Overview" on page 107 for more detail on types of probes

Setup Assistant (see the *Setup Assistant* help volume)

Logic Analysis System and Measurement Modules Installation Guide for probe pinout and circuit diagrams.

Define the Type of Measurement



There are two types of measurements: *state measurements* and *timing measurements*. Use the *Sampling* tab to select either type and to specify the details particular to that type.

Choose State or Timing

In a state measurement, the analyzer uses an external clock to determine when to sample. Each time the analyzer receives a state clock pulse, it samples and stores the logic state of the target system.

In a timing measurement, the analyzer is analogous to an oscilloscope. It samples at regular time intervals and displays the information in a waveform similar to the oscilloscope.

Set Measurement Mode

Each measurement type has different measurement modes. In general, there is a trade-off between number of signals and speed.

Because the measurement type and mode affect clocking and trigger options, you *must* set the measurement type first.

Set up State Clock

For state measurements, you must specify a clock to match the clocking arrangement used by your target system. It can be as simple as a single rising edge, or a complex arrangement of up to four signals. If the clock is incorrect, the trace data may indicate a problem where there isn't one. Specify the state clock in Clock Setup.

The equivalent in timing mode of the state clock is the Sample Period. The Sample Period sets the time between logic analyzer samples. For reliable data, the sample period should be no more than half of your clock period. Many engineers prefer setting it to one-fourth of the clock period.

Set up the Trace

The remaining controls finish your description of how you want to capture data. The trigger position determines where the events you specify in the *trigger sequence* will be relative to the majority of the data the logic analyzer captures.

Memory depth is affected by the measurement mode. Some logic analyzers also let you limit how big the acquisition will be with an Acquisiton Depth control.

Step 2: Set Up the Bus Labels (see page 13)

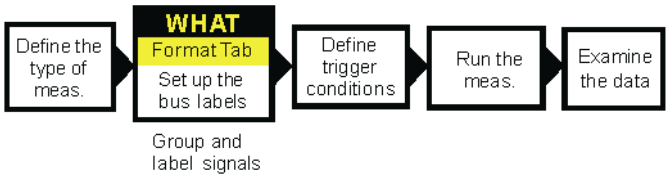
See Also

"The Sampling Tab" on page 40 for information on setting type and assigning pods

"Setting the Acquisition Mode" on page 41 for links to this analyzer's modes

"Performing Clock Setup (State only)" on page 41

Set Up the Bus Labels



The next step is to finish defining the physical connection between the target system and the analzyer. Use the *Format* tab to tell the analyzer what you want to measure on the target system. If you load a configuration file, this step is taken care of for you.

Group and Label Signals

Because the logic analyzer can capture dozens or even hundreds of signals, you need to organize the signals by grouping and labeling *channels*. Labels are used to group these channels into logical signals; for example, "addr bus". These groupings are then used in the trigger tab and the data displays. A label can have up to 32 channels. Each measurement can define 126 labels. Active channels are indicated like so ↕.

Set Threshold Level

The logic analyzer needs to know what threshold level the target system is using. You can set the analyzer to use a Standard or a User

Defined threshold voltage. The logic analyzer requires a minimum voltage swing of 500 mV at the probe tip to recognize changes in logic levels.

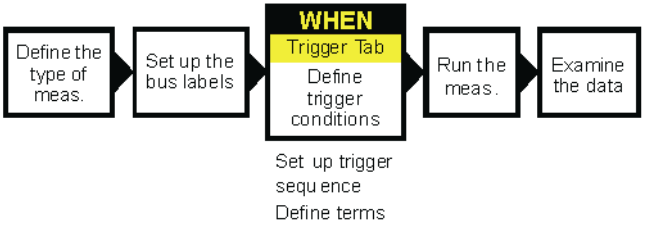
Step 3: Define Trigger Conditions (see page 14)

See Also

"Assigning Bits to a Label" on page 66

"The Format Tab" on page 55

Define Trigger Conditions



The third step is to define the trigger. The trigger settings tell the analyzer when you want to capture data. Controls for this are located under the *Trigger* tab. Configuration files saved from previous measurements automatically define trigger settings.

Set Up a Trigger Sequence

The trigger sequence is like a small program that controls when the logic analyzer stores data. There are trigger functions for the common tasks, or you can set up your own. The logic analyzer starts at the first trigger level, and stays there until the conditions described in that level become true. When that happens, the logic analyzer goes to the next level and follows the instructions there.

Define Terms

Trigger terms are like variables that you use in the trigger sequence. Depending on what analyzer you are using, you can either assign the values directly from within the trigger sequence or from the tabs (pattern, range, edge).

Step 4: Run the Measurement (see page 15)

See Also

"Defining Resource Terms" on page 94

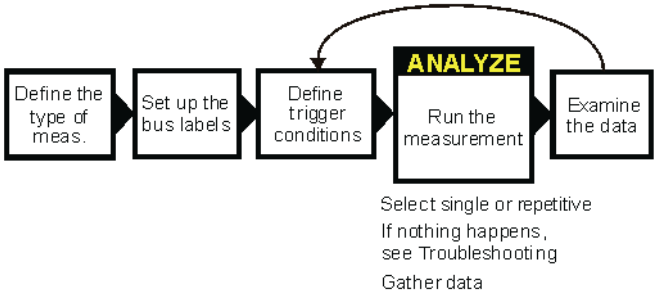
"Understanding Logic Analyzer Triggering" on page 76

"Setting Up a Trigger" on page 78

"The Trigger Tab" on page 75

Measurement Examples (see the *Measurement Examples* help volume)

Run the Measurement



You run the measurement by selecting the Run button. The Run button is labeled either Run, Group Run, or Run All. The difference between the three types is that *Run* starts only the instrument you are using, *Group Run* starts all instruments attached to group run in the Intermodule window, and *Run All* starts all instruments currently placed in the workspace.

Select Single or Repetitive

Runs can be single or repetitive. Single runs gather data until the logic analyzer memory is full, and then stop. Repetitive runs keep repeating the same measurement and are useful for gathering statistics. To stop a run, select the *Stop* button.

NOTE: Repetitive runs on a logic analyzer don't do equivalent time sampling like oscilloscopes do.

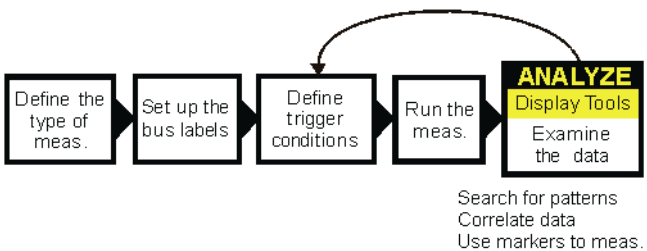
If Nothing Happens...

Analyzers with deep memory take a noticeable amount of time to complete a run. Because data is not displayed until acquisition completes, it may look like nothing is happening. Check the Run Status window to see if the logic analyzer is still running. Messages such as "Waiting in level 1" may indicate you need to refine your trigger. If the status shows as "Stopped", the analyzer either finished the acquisition, or was unable to run. The cause of the problem is listed in the bottom half of the Run Status window, and the messages are explained in more detail in “Error Messages” on page 33.

Step 5: Examine the Data (see page 16)

See Also “When Something Goes Wrong” on page 33

Examine the Data



Data from your measurement can be viewed in various display windows or offline. Some of the things you can do in the display windows are

- Search for patterns
- Display time-correlated data
- Use markers to make measurements and gather statistics

Search for Patterns

You can search displays for certain values, and place markers on them. There are two global markers which keep their place across all measurement views, even across instruments.

Display Correlated Data

There are several tools for correlation. The Intermodule window allows you to specify complex triggering configurations using several instruments. It is also useful for starting acquisitions at the same time. Global markers mark the same events in different displays, so you can switch views without having to reorient yourself. The Compare tool lets you compare two different acquisitions to look for changes.

Use Markers to Make Measurements

The markers can be positioned relative to the beginning, end, trigger, or another marker, as well as set to a specific pattern, state, or time. The *Markers* tab in the Display windows shows the time or state value as you move the markers or take new acquisitions.

See Also

Working with Markers (see the *Markers* help volume)

Using the Chart Display Tool (see the *Chart Display Tool* help volume)

Using the Distribution Display Tool (see the *Distribution Display Tool* help volume)

Using the Listing Display Tool (see the *Listing Display Tool* help volume)

Using the Digital Waveform Display Tool (see the *Waveform Display Tool* help volume)

Using the Compare Analysis Tool (see the *Compare Tool* help volume)

“Interpreting the Data” on page 26

Making a Basic State Measurement

This example uses the circuit board that is supplied with the *Making Basic Measurements* kit as the target system. The kit is supplied with every logic analysis system, or can be ordered from your Agilent Technologies Sales Office.

There are six major steps to making a basic measurement.

Connect the Logic Analyzer to your Target System

1. Connect probes.
   - Connect Pod 1 of the logic analyzer to J1 on the target system.

   The training board has terminations and headers already built in to the system, so you can connect the logic analyzer pod directly to the board.
2. Define the type of measurement On the Agilent Technologies 16700A/B logic analysis system, open a logic analyzer setup window.
   a. In the main window, select the logic analyzer icon.
   b. Choose *Setup...* from the menu.
   c. Select the *Sampling* tab.
   d. If the logic analyzer is not already set for *State*, change the type to *State*.
3. Set up the clock to match the target system's clocking scheme.
   a. In the bottom half of the *Sampling* tab window, choose the correct edges to match your clock. For Pod 1 attached to the training board, the correct clock is the falling edge on J.
      1. Select the *Off* button under J and choose "Falling Edge" from the menu.
4. Group and label bits.
   a. Select the *Format* tab.
   b. Optional - Insert a second label.

1. Select the *Label1* button.
2. Choose *Insert after...*.
3. In the *Enter Label Name* box, select the *OK* button.

c. Optional - Rename *Label1*.

1. Select the *Label1* button.
2. Choose *Rename...*.
3. Enter a new name in the name field.
4. Select the *OK* button to close the *Rename Label* box.

d. Select the bit assignment button.
The bit assignment button is to the right of a label name, and under a pod column.

e. Choose `........********` from the menu.
If none of the choices match your own system, choose *Individual...* and select the individual bits to assign them (*) or ignore them (.).

5. Define trigger events for patterns on buses.

a. Select the *Pattern* subtab in the *Trigger* tab.

b. Optional - Rename *Pattern1*.

1. Select the *Pattern1* field.
2. Enter a new name.

c. Select the appropriate label.

1. Select the label name button.
2. To define the event as a combination of labels, choose *Insert...* To use a different label to define the event, choose *Replace...*
3. In the dialog box, select the label name you want to use and then select the *OK* button.

d. Select the field with *XX* and enter the value you want to trigger on.

e. Optional - Repeat steps a - d for *Pattern2*.

6. Optional - Add additional trigger events to the trigger specification.

The logic analyzer automatically triggers on *Pattern1*, the first trigger event. You can set up more complex triggers by editing the sequence levels and combining trigger events.

a. Select the sequence level number *1* button and choose *Edit...*

b. In the dialog box, select the *Pattern1* button just after Trigger on and choose *Combo...*

c. In the Combination box, select the *Off* option button next to *Pattern2* and choose *On*.

d. To change the trigger to *Pattern1 and Pattern2*, select the *Or* option button to the right of the events and choose *And*.

e. Select the *OK* button.

f. Select the *Close* button.
The analyzer is now set to trigger when it detects both the pattern defined by *Pattern1* and the pattern defined by *Pattern2* on the target system's buses. The trigger sequence windows shows

```
Trigger on "(Pattern1.Pattern2)" 1 time
```

See Also "The Trigger Tab" on page 75

1. Select the *Run* button.
2. Examine the data.

   a. Select the *Window* menu.

   b. Select *Analyzer<A>* in the menu and choose *Listing<1>*.
   Depending on what other instruments are active, there may be more than one *Analyzer<A>*. Choose the one that refers to your analyzer.

   c. To have the listing display appear automatically when you run the logic analyzer, select Options -> Popup on Run -> On in the menu bar of the listing display.

   d. To insert additional labels, select the label name.

See Also **For Connection Information**

Logic Analysis System and Measurement Modules Installation Guide

For Details on the Training Board or More Tutorials

Making Basic Measurements

Examples of Typical Timing Measurements

The "Looking at State Events" group under Hardware Turn-On (see the *Measurement Examples* help volume) measurements.

Firmware Development (see the *Measurement Examples* help volume) measurements.

System Integration (see the *Measurement Examples* help volume) measurements.

For Details on the Logic Analyzer Interface

“The Sampling Tab” on page 40

“The Format Tab” on page 55

“The Trigger Tab” on page 75

Making a Basic Timing Measurement

This example uses the circuit board that is supplied with the *Making Basic Measurements* kit as the target system. The kit is supplied with every logic analysis system, or can be ordered from your Agilent Technologies Sales Office.

There are six major steps to making a basic measurement.

Connect the Logic Analyzer to the Target System

1. Connect probes.
   - Connect Pod 1 of the logic analyzer to J1 on the target system.

   The training board has terminations and headers already built in to the system, so you can connect the logic analyzer pod directly to the board.
2. Define the type of measurement. On the Agilent Technologies 16700A/B logic analysis system, open a logic analyzer setup window.
   a. In the main window, select the logic analyzer icon.
   b. Choose *Setup...* from the menu.
   c. Select the *Sampling* tab.
   d. If the logic analyzer is not already set for *Timing*, change the type to *Timing*.
3. Group and label bits.
   a. Select the *Format* tab.
   b. Optional - Insert a second label.
      1. Select the *Label1* button.
      2. Choose *Insert after...*.
      3. In the *Enter Label Name* box, select the *OK* button.
   c. Optional - Rename *Label1*
      1. Select the *Label1* button.

2. Choose *Rename...*.
3. Enter a new name in the name field.
4. Select the *OK* button to close the *Rename Label* box.

d. Select the bit assignment button.
The bit assignment button is to the right of a label name, and under a pod column.

e. Choose `........********` from the menu.
If none of the choices match your own system, choose *Individual...* and select the individual bits to assign them (*) or ignore them (.).

4. Define trigger events for a bus.

a. Select the *Trigger* tab.

b. Select the *Pattern* subtab.

c. Optional - Rename pattern *Pattern1*.

1. Select the *Pattern1* field.
2. Enter a new name.

d. Select the appropriate label.

1. Select the label name button.
2. To define the event as a combination of labels, choose *Insert...* To use a different label to define the event, choose *Replace...*
3. In the dialog box, select the label name you want to use and then select the *OK* button.

e. Select the field with *XX* and enter the value you want to trigger on.

5. Define trigger events for an edge.

a. Select the *Edge* subtab.

b. Optional - Rename *Edge1*.

1. Select the *Edge1* field.
2. Enter a new name.

c. Select the appropriate label.

1. Select the label name button.
2. To define the event as a combination of labels, choose *Insert...* To use a different label to define the event, choose *Replace...* Edges within an event are always OR'd together, which means only one of the edges on one of the labels needs to occur for the edge event to become true.
3. In the dialog box, select the label name you want to use and then select the *OK* button.

d. Select the edge assignment button (. . . . .) and enter the edge or edges you want to trigger on. Remember, if more than one edge is specified, then when the logic analyzer detects any of the edges the event becomes true.

6. Add the edge event to the trigger specification.

a. Select the sequence level number *1* button and choose *Edit...*

b. In the dialog box, select the *Pattern1* button and choose *Combo...*

c. In the Combination box, select the *Off* option button next to *Edge1* and choose *On*.

d. Select the *Or* option button where the path from *Pattern1* and the path from *Edge1* come together, and choose *And*.

e. Select the *OK* button.
The analyzer is now set to trigger when it detects Edge1 and Pattern1 on the bus. The trigger sequence window shows

```
Trigger on Pattern1.Edge1 occurs 1 times
```

.

The logic analyzer automatically triggers on the first trigger event. You can set up more complex triggers by editing the sequence levels and defining additional trigger events.

See Also

"The Trigger Tab" on page 75

1. Select the *Run* button.
2. Examine the data.

a. Select the *Window* menu.

b. Select *Analyzer<A>* in the menu and choose *Waveform<1>*. Depending on what other instruments are active, there may be more than one *Analyzer<A>*. Choose the one that refers to your analyzer.

c. To have the waveform display appear automatically when you run the logic analyzer, select Options -> Popup on Run -> On in the menu bar of the waveform display.

d. To insert additional labels, or expand overlaid signals, select the label name.

See Also

For Connection Information

Logic Analysis System and Measurement Modules Installation Guide

For Details on the Training Board or More Tutorials

Making Basic Measurements

Examples of Typical Timing Measurements

Hardware Turn-On (see the *Measurement Examples* help volume) measurements.

Firmware Development (see the *Measurement Examples* help volume) measurements.

System Integration (see the *Measurement Examples* help volume) measurements.

For Details on the Logic Analyzer Interface

"The Sampling Tab" on page 40

"The Format Tab" on page 55

"The Trigger Tab" on page 75

Interpreting the Data

After you've acquired a trace with the logic analyzer, you can analyze it in the display tools. The logic analysis system also provides filtering and compare tools for more complex analysis.

The logic analyzer is automatically connected to the Waveform and Listing displays when you set up a measurement. To move to that display,

1. Select the *Window* menu.
2. Select the name of the analyzer whose data you want to view.
3. Choose *Waveform* or *Listing*.
   Source Viewer brings up a Listing display but requires an inverse assembler and an ADDR label.

- "Analysis Using Waveform" on page 26
- "Analysis Using Listing" on page 28

Analysis Using Waveform

Waveform is most useful for *timing* data. If you look at *state* data that uses *store qualification*, you won't be able to easily see where samples were not stored. Timing data, however, is periodic and stores all samples and so works well with Waveform.

Example: Looking for a Missing Pattern

You can easily use the waveform tool to make timing measurements. For example, if you were triggering when a pattern doesn't follow an edge within a certain time (see the *Measurement Examples* help volume), you would probably want to look at your *data set* to see if the pattern ever did occur. This might be the case when you verifying that the system is responding to an interrupt.

After triggering on an instance where the response did not appear quickly enough, you might take these steps in the Waveform display:

1. Find the edge.
   a. Select the *Search* tab.
   b. Select the down arrow after the Label field, and select the label containing the edge.
   c. Select the Value field and enter 1.
   d. Select the *Next* button to locate the edge transition.
2. Place a marker on the edge.

   Select the *Set G1* button. This sets global marker G1 at the location of the edge you just found.
3. Search for the pattern. Searches start at your current location. Since you just set the global marker G1, it indicates where the search starts from.
   a. Select the down arrow after the Label field, and select the label containing the pattern.
   b. Select the Value field and enter the pattern you are searching for.
   c. Select the down arrow after the When field and select *Entering*.
   d. Select the *Next* button to find the next occurrence of that pattern after G1.

   If the logic analysis system cannot find the pattern, a "Value not found" message pops up.
4. Place a marker on the pattern.

   Select the *Set G2* button. This will set global marker G2 at the location of the pattern.
5. Find the time between the edge and the pattern.
   a. Select the *Markers* tab.
   b. In the G2 row, select the down arrow after from, and select *G1*.

   The value after the from field changes to the time between G1 and G2. You can toggle between time and samples by selecting the arrow after the Time or Samples field.

See Also

Using the Digital Waveform Display Tool (see the *Waveform Display Tool*

help volume)

Using the Listing Display Tool (see the *Listing Display Tool* help volume)

Using the Chart Display Tool (see the *Chart Display Tool* help volume)

Using the Distribution Display Tool (see the *Distribution Display Tool* help volume)

Using the Compare Analysis Tool (see the *Compare Tool* help volume)

Using the Pattern Filter Analysis Tool (see the *Pattern Filter Tool* help volume)

Analysis Using Listing

Listing is more useful than Waveform when your target system is running code because it shows the labels as states rather than transitions. Listing is especially useful when you have defined meaningful symbol names for your states. If you have an inverse assembler, you might prefer *Source Viewer* which functions like Listing.

Example: Examining a Subroutine

Listing is the preferred display tool for state measurements. for example, if you were trying to see if a subroutine were exiting abnormally, you might want to measure the number of states between entering and exiting the subroutine. After acquiring data with the logic analyzer, you could examine the *data set* in the Listing display like this:

1. Find the start of the subroutine.

   Assume the subroutine starts at the address 0x58FC.

   a. Select the *Search* tab.

   b. Select the down arrow after the Label field, and select *ADDR*.

   c. Select the Value field, and enter the starting address, 0x58FC.

   d. Select the down arrow after the When field and select *Present*.

   e. Select the *Next* or *Prev* button to move the display to the address.

2. Place a marker on the start of the subroutine.

   Select the *Set G1* button. This sets global marker G1 at the address you just found.

3. Find the end of the subroutine.

   Assume the end of the subroutine is at address 0x58FF. Searches always start at the current location. Since you just set the global marker G1, it indicates where the search starts from.

   a. Select the Value field, and enter 58FF.

   b. Select the *Next* button to find the next occurrence of 0x58FF after the starting address.

4. Place a marker on the end of the subroutine.

   Select the *Set G2* button to set global marker G2 at this position. This lets you refer to G2 when you want to know where the subroutine ends.

5. Find the number of states between the start and end of the subroutine.

   Since you've placed markers at the start and end of the subroutine, all you have to do is find the number of states between those markers.

   a. Select the *Markers* tab.

   b. In the G2 row, select the second down arrow and select *Sample*.

   c. Select the down arrow after from, and select *G1*.

   The value after the from field changes to the number of states between G1 and G2. You can toggle between time and states by selecting the arrow after the Time or Samples field.

Now you know how long the execution stayed in the subroutine, and can also examine the data set between G1 and G2 to look for unusual data.

See Also

Using the Digital Waveform Display Tool (see the *Waveform Display Tool* help volume)

Using the Listing Display Tool (see the *Listing Display Tool* help volume)

Using the Chart Display Tool (see the *Chart Display Tool* help volume)

Using the Distribution Display Tool (see the *Distribution Display Tool*

help volume)

Using the Compare Analysis Tool (see the *Compare Tool* help volume)

Using the Pattern Filter Analysis Tool (see the *Pattern Filter Tool* help volume)

Coordinating Measurements

When you want to coordinate the triggering of two measurements happening on the same logic analyzer, you need to use arming control. The two measurement *machines* are by default controlled by the same run button. However, you can set machine 1 to *arm* machine 2, or the other way around.

If the timing measurement and state measurement are being handled by two different instruments, you can coordinate them by connecting both instruments to the same display, and then setting the correlation in the correlation dialog that appears.



You can tell which instrument is controlling a measurement by looking at the instrument icon. The letter on the icon represents the slot of the *frame* that the instrument is in. Icons with the same letter are controlled by the same instrument.

See Also

Overview - Multiple Instrument Configuration (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Overview - Multiple Machine Configuration (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Loading and Saving Logic Analyzer Configurations

The Agilent Technologies 16550A logic analyzer settings and data can be saved to a configuration file. You can also save any tools connected to the logic analyzer. Later, you can restore your data and settings by loading the configuration file into the logic analyzer.

The Agilent Technologies 16550A logic analyzer can also load configuration files generated for Agilent Technologies 16510B, 16511B, and 16540A models of logic analyzer, but they must first be re-saved as Agilent Technologies 16550A configuration files on the old logic analysis system before being loaded into a Agilent Technologies 16550A on an Agilent Technologies 16700A/B-series logic analysis system. Also, the data cannot be displayed across logic analyzer models, and some setting may change when no comparable setting exists in the Agilent Technologies 16550A logic analyzer.

See Also

Loading Configuration Files (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Saving Configuration Files (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

When Something Goes Wrong

- "Nothing Happens" on page 38
- "Error Messages" on page 33
- "Suspicious Data" on page 38
- "Interference with Target System" on page 33

Interference with Target System

Capacitive Loading on the Target System

Excessive capacitive loading can degrade signals, resulting in suspicious data or even system lockup. All analysis probes add capacitive loading, as can custom probes you design for your target. To reduce loading, remove as many pin protectors, extenders, and adapters as possible.

Careful layout of your target system can minimize loading problems and result in better margins for your design. This is especially important for systems running at frequencies greater than 50 MHz.

Error Messages

"Slow or Missing Clock" on page 34

"Waiting for Trigger" on page 37

"Timer is Off in Sequence Level Where It Is Used" on page 35

"Timer is Specified in Sequence, But Never Started" on page 35

"Measurement Initialization Error" on page 34

"Two Pod Pairs are Needed To Use Both Timers" on page 36

"Maximum of 32 Channels Per Label" on page 34

"Trigger inhibited during timing prestore" on page 36

"Clock Lines Must Be Connected to the Card in Slot X" on page 37

Maximum of 32 Channels Per Label

The logic analyzer can only assign up to 32 channels for each label. If you need more than 32 channels, assign them to two labels and use the labels in conjunction.

See "Adding and Deleting Labels for Terms" on page 100 for how to use more than one label with trigger *terms*.

Measurement Initialization Error

The logic analyzer module failed the internal calibration which it performs when *Run* is selected. An internal calibration failure can indicate either a software or a hardware problem.

Possible Causes

- Hardware failure
- Software failure

Run the Self-Test Utility (see page 39) on the logic analyzer and contact your Agilent Technologies Sales Office for service or software upgrades.

Slow or Missing Clock

The message "Slow or Missing Clock" only appears in *state measurement*s. However, if you have another instrument armed by the state analyzer, a slow or missing clock on the state analyzer will prevent the other instrument from triggering also.

Possible Causes

- Target system is not running properly

  Check that the system is running properly. The logic analyzer and other probing fixtures such as pin extenders can place too much capacitive load on a system.

- Incorrect clock specification

Make sure the target system clock matches the clock specified under *Sampling*.
Also check that the probe's clock channels are attached to the target's clock lines either directly or through an analysis probe.
If you are using an analysis probe, the probe's User's Guide should show the correct connections and settings.

- Bad probe connection

  Check that the probe is securely attached to the clock line and is receiving a signal. The logic analyzer shows activity indicators under the *Sampling* and *Format* tabs.

- Incorrect signal level

  The clock's threshold level is set by the pod threshold. For the logic analyzer's J clock, check the pod threshold of pod 1 of the *master card*.

See Also

"Performing Clock Setup (State only)" on page 41

"Setting the Pod Threshold" on page 72

Timer is Specified in Sequence, But Never Started

This error occurs because you have used a timer term in your *trigger sequence*, but not turned the timer on using the timer controls. Timer controls are available in all sequence levels except the first. You do not need to turn on the timer in the same sequence level, but it does need to be on when it is used in the trigger sequence.

This error message always occurs with the

`Timer is off in sequence level where it is used`

warning.

See "Using Timer Terms" on page 98 for more information on timer control.

Timer is Off in Sequence Level Where It Is Used

Timer controls are available in all sequence levels except the first. You do not need to turn on the timer in the same sequence level, but it does need to be on when it is used in the trigger specification.

Possible Causes

This warning occurs because you have used a *timer term* in your *trigger specification*, but not turned the timer on using the timer controls.

This warning message always occurs with the

`Timer is specified in sequence, but never started`

error message.

See "Using Timer Terms" on page 98 for more information on timer control.

Trigger inhibited during timing prestore

The "trigger inhibited" informational message appears when you have a logic analyzer making a *timing measurement*, and it is set to a slow sample rate. The logic analyzer will fill the designated amount of pre-trigger memory before checking for the trigger condition.

To calculate how long this should take, multiply the sample rate by the percentage of pre-trigger memory and the acquisition depth. For example, if

sample period = 1.0 ms (sample rate = 10^3 samples/sec.)
trigger position = center (percentage of pre-trigger memory = 50%)
acquisition depth = 64K (roughly 64 x 10^3 samples)
then the approximate time is 32 seconds.

Two Pod Pairs are Needed To Use Both Timers

Possible Causes

This message only occurs when all pod pairs are assigned, and the *machine* which is using both timers has only one pod pair.

You can either assign another pod pair to the machine with both timers, or change your *trigger specification* to use only one *timer term*.

See Also

"Using Timer Terms" on page 98

Waiting for Trigger

This message indicates that the specified trigger pattern has not occurred. This may be expected, as when you are waiting to trigger on an unusual event.

Possible Causes

- Misaligned boundaries for addresses

  When the target is a microprocessor that fetches only from long-word aligned addresses, if the trigger is set to look for an opcode fetch at an address that is not properly aligned, the trigger will never be found.

- Trigger set incorrectly

Some strategies you can use when verifying or debugging trigger sequence levels are:

- 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 *store qualifiers* to store "no states" (or only the states you are 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 changed, you can recall the original trigger setup and make changes to it.

See Also

"Branches Taken Stored / Not Stored (State only)" on page 94

"Saving and Recalling Trigger Sequences" on page 82

Clock Lines Must Be Connected to the Card in Slot X

This message occurs if the logic analysis system has detected a slow or missing clock on a logic analyzer *module* that has an expander *card*.

Possible Causes

- Using a clock channel from an expander card

  Reconnect the logic analyzer probes to use a clock channel from the *master card* of the module.

Only the *clock channels* of the master card can be used in the state clock. The master card of a module is identified in the system window.

- Clock properly connected, but slow

  See “Slow or Missing Clock” on page 34.

Nothing Happens

Look for an error message in the *message bar* at the top of the window. Common messages are "slow or missing clock" and "Waiting for trigger".

If the *Run* button briefly grayed-out (and the *Stop* or *Cancel* buttons briefly became active), select the *Window* menu, select the logic analyzer's slot, then choose the Waveform or Listing display.

See Also

“Slow or Missing Clock” on page 34

“Waiting for Trigger” on page 37

Suspicious Data

Intermittent Data Errors

Check for poor connections, incorrect signal levels on the hardware, incorrect logic levels under the logic analyzer's Config tab, or marginal timing for signals.

Unwanted Triggers

If you are using an inverse assembler or a pipeline, triggers can be caused by instructions that were fetched but not executed. To fix, add the prefetch queue or pipeline depth to the trigger address.

The depth of the prefetch queue depends on the processor that you are analyzing, and can be quite deep.

Another solution which is sometimes preferred with very deep prefetch queues is to add writes to dummy variables to your software. Put the instruction just before the area you want to trigger on, then trigger on the actual write to this variable. Although the instruction is prefetched, the analyzer can be set to only trigger when the write is executed.

Testing the Logic Analyzer Hardware

In order to verify that the logic analyzer hardware is operational, run the Self Test utility. The Self Test function of the logic analysis system performs functional tests on both the system and any installed modules.

1. Disconnect all probes of the logic analyzer *module*.
2. If you have any work in progress, save it to a configuration file. (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)
3. Disconnect all loads, adapters, or preprocessors from the probe cable ends.
4. From the system window, select the *System Administration* button.
5. Select the *Admin* tab, then select the *Self Test...* button.
   The system closes all windows before starting up Self Test.
6. Select *Master Frame*.
   If the module is in an expansion frame, select *Expansion Frame*.
7. Select the logic analyzer that you want to test.
8. In the Self Test dialog box, select *Test All*.
   You can also run individual tests by selecting them. Tests that require you to do something must be run this way.

If any test fails, contact your local Agilent Technologies Sales Office or Service Center for assistance.

See Also

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

Agilent Technologies 16550A 100 MHz State/500 MHz Timing Service Guide

The Sampling Tab

The options under *Sampling* tell the analyzer the overall way in which you want to make a measurement. The options include:

- The acquisition mode.
- The data sample rate.
- How much data before and after the trigger.
- How much data to acquire in all.

See Also

"Naming the Analyzer" on page 44

"Turning the Analyzer Off" on page 45

"Setting the Acquisition Mode" on page 41

"Sample Period (Timing Only)" on page 45

"Performing Clock Setup (State only)" on page 41

"Trigger Position Control" on page 53

"Acquisition Depth" on page 40

Acquisition Depth

Acquisition Depth, located under *Sampling* and also under *Trigger Settings*, sets how much data will actually be acquired. While the Agilent Technologies 16550A logic analyzer has a maximum memory depth of 4K samples in State Mode and 8K in Timing Mode, sometimes you may not want to wait for all that memory to fill up.

The numbers shown in the Acquisition Depth menu are approximations. The combination of *count* tags, pod assignments, and acquisition modes affect what choices are available. Also, the values are memory depth *per channel*.

Setting the Acquisition Mode

The measurement type affects all other areas of the logic analyzer setup. It is set under the *Sampling* tab.

If you want the logic analyzer to sample data according to the target system's clock, select *State*. Each time the clock signal becomes valid, the analyzer will sample data from the system under test.

If you want the logic analyzer to sample the target system independently of its clock, select *Timing*. Since in timing mode 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. The Agilent Technologies 16550A logic analyzer can be split into two analyzers, but only one of them may be a timing analyzer.

State Mode and *Timing Mode* offer different options for the acquisition mode field in the top left corner under Controls. The acquisition mode affects the channel width, memory depth, and sample rate.

"State Acquisition Modes" on page 46

"Timing Acquisition Modes" on page 46

"Transitional Timing Acquisition Modes" on page 46

"Comparison of Transitional and Conventional Timing Modes" on page 51

"Glitch Acquisition Mode" on page 52

Performing Clock Setup (State only)

When you select State Mode, the Clock Setup area appears under the State Mode Controls in *Sampling*. The Clock Setup contains three controls and a display area. The clocks you specify here control when the analyzer samples your data. Ideally, the logic analyzer's state clock setup should be identical to the target system's clock. Differences could result in invalid data.

Mode field

The Mode field lets you select among *Master only*, *Master/slave*, and *Demultiplex*. The default is Master only. When you select the others, another control to set the slave clock appears at the bottom of the Clock Setup area. It also enables the Pod Clock field under *Format*.

For more detail on the uses of *Master/slave* and *Demultiplex* clocking, see “Clock Modes (State only)” on page 42.

Advanced Clocking

Advanced clocking allows you to specify clock qualifiers on individual clock edges instead of the group of clock edges. When you select it, the individual clock channels are replaced by *Master Clock...* or *Slave Clock...* Selecting these brings up a dialog that lets you combine edges and qualifiers in more complex Boolean expressions. When you switch from Advanced Clocking to regular clocking, some of the qualifiers are erased.

Clock Channel Specifiers

The *clock channel* specifiers graphically show your clock setup. Edges are ORed ("+") together, and qualifiers are ANDed (".") to all edges. To qualify just one of the edges, switch to Advanced Clocking.

All clock channels for the clock setup must be on the pods of the *master card* of the *module*, but the pods do not need to be part of the state measurement.

See Also

“Clock Modes (State only)” on page 42

Clock Modes (State only)

The Pod Clock field under *Format* appears when a clock mode other than *Master only* is selected in *Sampling*. The Pod Clock field indicates whether a pod's data lines are to be sampled into memory by the master clock, slave clock, or both (demultiplex).

The Pod Clock field and the clocking arrangement are only available in a *state* analyzer.

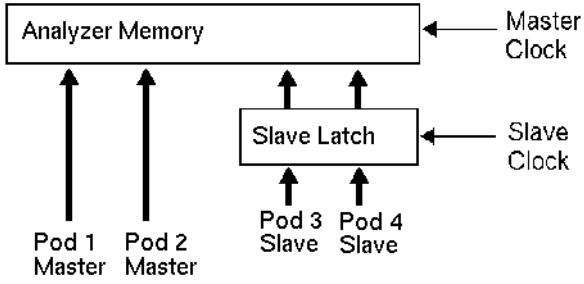
Master

Data on all pods assigned to *Master Clk* is strobed into memory when the status of the clock lines match the clocking arrangement specified for *Master* in the clock setup.

Slave

Data on a pod designated *Slave Clock* is latched when the status of the slave clock inputs meet the requirements of the slave clocking arrangement. Then, followed by a valid master clock, the slave data is strobed into analyzer memory along with the master data.

If multiple slave clocks occur between master clocks, only the data latched by the last slave clock prior to a valid master clock is strobed into analyzer memory.



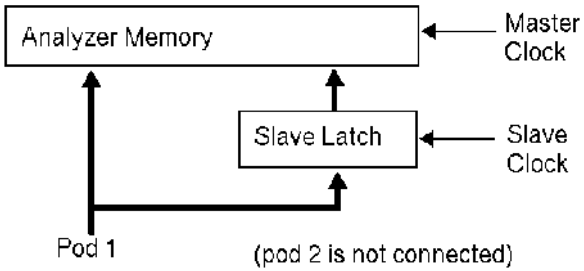
Latching Slave Data

Demultiplex

The demultiplex mode is used to store two different sets of data that occur at different times on the same channels. In demultiplex mode, only one pod of the *pod pair* is used, and that pod is selectable (see page 71). Channel assignments are displayed as *Pod* and *Slave Pod*. Assign slave and master data to separate *labels* for easy recognition of the two sets of data.

Both the master and slave clocks are used in the demultiplex mode. When the analyzer sees a match between the slave clock input and the slave clock arrangement, demux slave data is latched. Then, following a valid master clock, the slave data is strobed into analyzer memory

along with the master data. If multiple slave clocks occur between master clocks, only the data latched by the last slave clock prior to the master clock is strobed into analyzer memory.



Latching Demultiplex Data

Naming the Analyzer

The *Analyzer Name* field under *Sampling* lets you assign a specific name to the analyzer. When you have stored several measurement setups to disk and later reload them, having assigned a specific name to an analyzer can help identify what the setup is for. The analyzer name is also used in the workspace to label the analyzer icons and as part of the title of the analyzer setup window.

There are two analyzers per logic analyzer *module*. To activate the second one, go to the *Workspace* window and drag the second analyzer on to the workspace.

The default names for the analyzers are *Analyzer<N>* and *Analyzer<N2>*, where *N* is the slot of the analyzer module.

To Name an Analyzer

1. Select the *Analyzer Name* field.
2. Enter the new name.

The name now appears below the instrument tool icon in the workspace.

Turning the Analyzer Off

The *On* and *Off* checkboxes under the *Sampling* tab are a shortcut for activating and de-activating the analyzer.

Analyzers come up in the *On* state. If you select the checkbox, the label changes to *Off* and the Analyzer Shutdown Options dialog appears.

Soft

"Soft" shutdowns have the same effect as though you selected *Off* in the Type control of the Pod Assignment dialog under *Format*. The analyzer window remains, but most options are unavailable. Soft shutdowns are easily reversed by selecting the box next to *Off*.

Hard

"Hard" shutdowns have the same effect as deleting the analyzer icon from the workspace. The analyzer window and the windows of its display tools are closed, and the analyzer is removed from the workspace. To turn the analyzer back on, select the analyzer icon in the System window. You will need to restore any complex analysis or display tools.

Sample Period (Timing Only)

Sample Period is used to set the time between data samples. The inverse of sample period is sample rate. Every time a new sample is taken, the analyzer will see updated measurement data. The choices available for sample period depend on the acquisition mode.

If your analyzer is set to *timing*, the Sample Period control is located under the Sampling tab's Timing Mode Controls and under the Trigger tab's Settings sub-tab.

If your analyzer is set to *state*, the Sample Period control is not available. Its functionality is handled by the Clock Setup.

State Acquisition Modes

Full Channel/4K Memory/100 MHz

All pods are available. Memory depth is 4K samples per channel. If time or state count is turned on, the total memory could be split between data acquisition storage and time or state count storage. To maintain the full 4K samples per channel depth, leave one pod pair unassigned. State clock speed is 100 MHz.

Half Channel/8K Memory/100 MHz

Only one pod per pod pair is available, for 17 channels of width and a total memory depth of 8192 bytes. Time and state count is not available in this mode. State clock speed is 100 MHz.

Timing Acquisition Modes

In conventional timing acquisition mode the analyzer stores measurement data at each sampling interval.

Conventional Full Channel 250 MHz

The total memory depth is 4K samples per channel, with data being sampled and stored as often as every 4 ns.

Conventional Half Channel 500 MHz

Only one pod of each pod pair is available. The total memory depth is 8192 samples, with data being sampled and stored as often as every 2 ns.

See Also

"Sample Period (Timing Only)" on page 45

"Pod Selection" on page 71

Transitional Timing Acquisition Modes

In transitional timing modes, the timing analyzer samples data at regular intervals just like it does in conventional timing mode. However, in transitional timing mode, it only stores data when there is a change on the lines assigned to a label within a pod pair.

Data is collected on a *pod pair* basis, even when labels cross pod boundaries. It is still displayed on a label basis, however. Because of this, some pod pairs with less active channels may take longer to fill memory and the end of the acquisition is delayed.

Each time a transition occurs on any of the assigned bits, all bits of the pod pair are stored along with a time tag. If transitions are exceptionally far apart, the timer which generates the time tags may overflow.

NOTE: The timer does not generate a warning when it overflows. If the time between transitions in the display seems too short, it may be due to timer overflow.

"Transitional Full Channel 125-MHz Mode" on page 47

"Transitional Half Channel 250-MHz Mode" on page 48

"Other Transitional Timing Considerations" on page 50

Transitional Full Channel 125-MHz Mode. The total memory depth is 4K samples per channel with a channel width of 34 channels per pod pair. Data is sampled for new transitions every 8 ns. When the timing analyzer runs in the 125-MHz mode, it operates very much like the state analyzer with Count Time turned on. The only exceptions are that the store qualification comes from transition detectors instead of the sequencer and that the analyzer uses an internal clock.

In full-channel transitional timing mode, the analyzer uses half its memory (2 K) to store time tags. Because each pod pair stores transitions at its own rate, you do not have the option of using a free pod to retain full memory as you have in the state mode.

When a transition is detected after a sample with no detected transition, two samples are stored. One sample is a "before transition sample" and the other is an "after transition sample." As long as there are transitions in the subsequent sample, only one sample is stored. When the next sample occurs without a transition, the two stored sample sequences (one before, one after) repeats with the next detected transition.

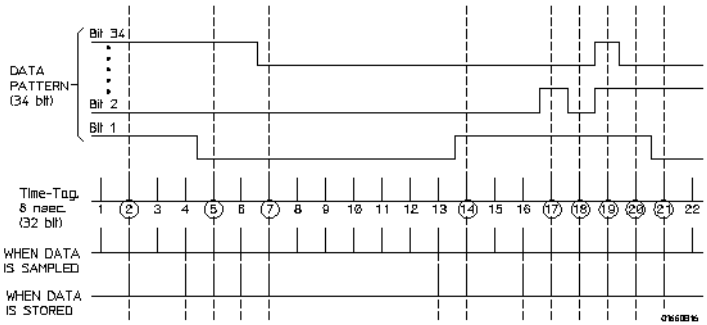
Minimum and Maximum Transitions Stored

Minimum Transitions Stored

Normally, transitions have not occurred at each sample period. This is illustrated below with time-tags 2, 5, 7, and 14. When transitions happen at this rate, two cycles are stored for every transition. Therefore, with 2 K samples of memory, 1 K of transitions are stored. You must subtract one, which is necessary for a starting point, for a minimum of 1023 stored transitions.

Maximum Transitions Stored

If transitions occur at a fast rate, such that there is a transition at each sample point, only one sample is stored for each transition as shown by time tags 17 through 21 below. If this continues for the entire trace, the number of transitions stored is 2 K samples. Again, you must subtract the starting point sample, which then yields a maximum of 2047 stored transitions.



In most cases a transitional timing trace is stored by a mixture of the minimum and maximum cases. Therefore, the actual number of transitions stored will be between 1023 and 2047.

Transitional Half Channel 250-MHz Mode. The total memory depth is 8 K samples with a channel width of 17 channels on one pod. The pod used within the pod pair is selectable. Data is sampled for new transitions every 4 ns. Transitional timing running at 250 MHz is the same as the 125-MHz mode, except that two single-pod data samples (17 bits x 2 = 34 bits) are stored instead of one full pod pair data sample (34 bits). Storing two single-pod data samples occurs because in half channel mode, data is multiplexed into the sequencer pipeline in

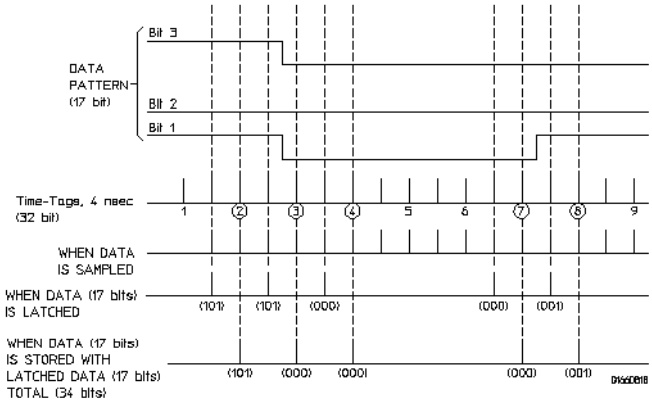
two 17-bit samples. The first 17-bit sample is latched, the next 17-bit sample is sent down the pipeline along with the latched 17-bit sample.

Note that the transition detector still looks at a full 34 bits. This means it is looking at two samples at a time instead of one. In this mode, between 682 and 4094 transitions are stored.

Minimum and Maximum Transitions Stored

Minimum Transitions Stored

The following example shows the kind of data stored from a data stream with transitions that occur at a slow rate (more than 24 ns apart).



The illustration depicts transitions stored in two different ways, depending strictly on chance. Remember that the transition detector only looks at the full 34 bits while the data is stored as two 17 bit samples. So, the transition detector will not see time tag 3 (101/000) as a transition. However, when it compares it to time tags 2 (101/101) or 4 (000/000), it sees a difference and detects them as transitions. For this first set of time tags, the transition detector sees more transitions than are really there. This causes the analyzer to store 6 samples per transition (three 34-bit sample pairs), instead of just two, like in the 125-MHz mode. If all the transitions will be stored in this way throughout the trace, the minimum number of stored transitions is 682 (4096/6).

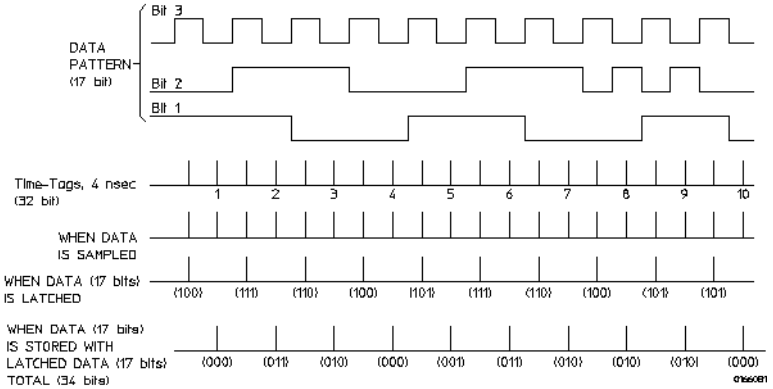
With time tags such as 7 (000/000) and 8 (001/001), transitions can fall between the pairs of samples. When this happens, only one transition is

detected and only 4 samples (two sample pairs) are stored. If all transitions will be stored in this way, 1023 (4096/4) transitions are stored.

From run to run, the actual number of transitions stored for transitions that occur at a slower rate will fall between these two numbers. This storage occurs based on the probability of a transition falling between a sample pair or falling within a sample pair.

Maximum Transitions Stored

The following example shows the case where the transitions are occurring at a 4-ns rate:



In this case, transitions are being detected with each sample. Therefore, they are all being stored. In addition, each sample pair contains a transition. For example, time tag 1 (100/000) contains a transition and is different from time tag 2 (111/011), which also contains a transition. The difference between the two will trigger the transition detector. If this same trend occurs throughout the trace, the analyzer would store (4 K samples - 1) transitions, or 4095. As with the 125-MHz mode, the actual number of transitions stored will fall somewhere between 682 and 4095, depending on the frequency of transitions.

Other Transitional Timing Considerations. Pod Pairs are Independent

In single run mode, each pod pair runs independently. When one pod

pair fills its trace buffer, it will not shut the others down. If there is a pod pair with enabled data lines and with no transitions on its lines, you get a message "Storing transitions after trigger for pods nn/nn." In repetitive run mode, a full pod pair waits 2 seconds, then halts all other pod pairs.

Increasing Duration of Storage

In the 125-MHz mode, a transition on any one of the 34 bits for each sample (if they are all turned on) will cause storage. Reducing the number of bits that are turned on for any one pod pair will more than likely increase data storage time. Separating data lines which contain fast occurring transitions from lines with slow occurring transitions also helps. If separating the data lines, be sure to cross pod pair boundaries. It does not help to move fast lines from pod 1 to pod 2; they must be moved to pod 3, which is a different pod pair. In the 250-MHz mode, a transition on any one of the 17 bits will cause storage.

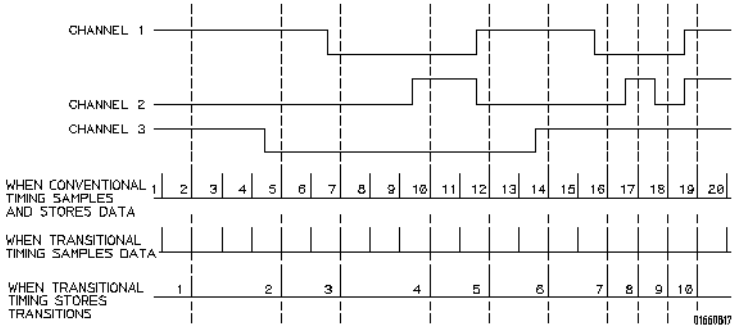
Invalid Data

The analyzer only looks for transitions on data lines that are turned on. Data lines that are turned off store data, but only when one of the lines that is turned on transitions. If the data line is turned on after a run, you would see data, but it is unlikely that every transition that occurred was captured.

Comparison of Transitional and Conventional Timing Modes

In Conventional Timing Acquisition mode, the analyzer stores measurement data at each sampling interval. In Transitional Timing Acquisition mode, the timing analyzer samples data at regular intervals, but only stores data when there is a threshold level transition. Each time a level transition occurs on any of the bits of a pod pair, data on all channels is stored. A time tag is stored with each stored data sample so the measurement can be reconstructed and displayed later.

When using transitional timing, you might consider how many transitions can be stored. The number depends on the mode and frequency of transition occurrence.



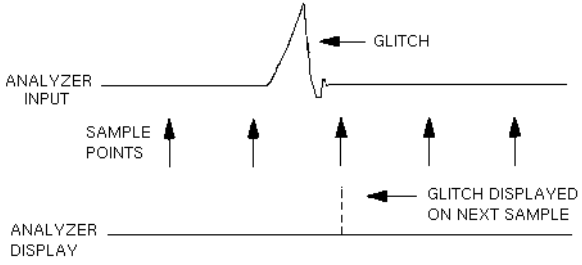
See Also

"Timing Acquisition Modes" on page 46

"Transitional Timing Acquisition Modes" on page 46

Glitch Acquisition Mode

In Glitch Acquisition mode, a glitch is defined as a pulse with a minimum width of 3.5 ns and a maximum width of 8 ns, or the sample period, whichever is larger. As an example, if the sample period is 16 ns, then a glitch is defined as being between 3.5 ns and 16 ns. One advantage of the glitch mode is that if you expand the sample rate, a pulse that is less than the sample rate will still be displayed as a vertical dashed line.



Glitch Half Channel 125 MHz

The total memory depth is split between data storage and glitch storage. Data acquisition memory depth is 2048 per channel. Glitch storage is 2048 per channel. Data is sampled for new transitions at most every 8 ns.

Trigger Position Control

The Trigger Position control, located under *Sampling* and also *Trigger Settings*, determines how much data is stored before and after the *trigger* occurs for all subsequent *acquisitions*. The *trigger point* is placed at a specified position relative to the data in memory. The analyzer triggers differently depending on if it is in *Timing* or *State* mode.

Timing Mode

When a Run is started, the analyzer will not look for a trigger until at least the proper percentage of pretrigger data has been stored. After a trigger has been detected, the specified percentage of posttrigger data is stored before the analyzer halts.

State Mode

When the Run is started, the analyzer immediately looks for the trigger condition. The percentage of pretrigger data determines the *maximum* amount of pretrigger data to save.

The trigger position choices are Start, Center, End, User Defined, or Delay. Delay is only available in timing mode.

- Start

  The trigger position is set at the starting point of available memory. This process results in maximum posttrigger data and minimum pretrigger data. Note that there will be a small amount of pretrigger data stored.

- Center

  The trigger position is set at the center point of available memory. This results in half pretrigger data and half posttrigger data.

- End

  The trigger position is set at the end point of available memory. This results in maximum pretrigger data and minimum posttrigger data. Note that there will be a small amount of posttrigger data stored.

- User Defined

When the trigger position is set to User Defined, a trigger position slider appears. Use this slider to set the trigger position any where between 0% and 100%. As the slider is adjusted, the *% Post Store* indicator shows the amount of data that will be stored after the trigger point.

- Delay

  This option delays the start of data storage until some time after the trigger. The range of the delay time is affected by the sample period, but it could range between 16 ns to 8 ks.

The Format Tab

Under *Format*, you specify the parts of the target system that you want the logic analyzer to look at. You set up labels to match the buses of the target system, and set the threshold level for the signals. For a *state measurement*, you also adjust the setup and hold time.

For advanced measurements, *Format* is also where you assign pods and specify whether to look at channels on the master, slave, or demultiplexed clock.

Common Measurement Controls

"Labels: Mapping Analyzer Channels to Your Target" on page 70

"Setting the Pod Threshold" on page 72

"State Clock Setup/Hold (State only)" on page 73

Advanced Measurement Controls

"Assigning Pods to the Analyzers" on page 65

"Clock Modes (State only)" on page 42

"Setting Up the Pod Clock" on page 70

"Pod Selection" on page 71

See Also

"Data On Clocks Display" on page 66

"Activity Indicators" on page 64

"Assigning Bits to a Label" on page 66

"Inserting and Deleting Labels" on page 67

"Turning Labels On and Off" on page 68

"To reorder bits in a label" on page 69

"Importing Netlist and ASCII Files" on page 56

"Label Polarity" on page 68

Importing Netlist and ASCII Files

Netlist Files

The Netlist Import feature provides a method for importing busses and signals from ASCII netlists created by EDA tools. In order for the feature to work, the device under test must either use the Agilent E5346A high density adapter or the Agilent Technologies Termination Adapter. The adapter must be included as a connector in the netlist.

You can create a Netlist file using an EDA tool. When importing a Netlist file, you can specify which connectors are of interest; connectors that are not specified will be ignored.

Example

```
NET'/Bus1(3)'J1-7
```

Bus1	Four bit bus
(3)	Bit 3
J1-7	Connector J1, pin 7

To Import a Net List

1. Configure the connector names referenced in the Netlist to logic analyzer pods.
2. Specify the Net List file to import.
3. Verify that Nets were correctly mapped to Labels, Pods, and Channels.
4. Select or create labels to appear in the interface.

   Map the Connector Names (see page 62).

   Import the Net List File (see page 62)

   Verify Net to Label Mapping (see page 63)

   Select/Create Interface Labels (see page 64)

ASCII Files

You can create an ASCII file using any Windows, MS-DOS, or UNIX text editor. The ASCII file should contain bus names, pod/clock numbers, and channel definitions.

For Example

```
Label1;A2[15:5];A1[5,2]
```

Label1 Bus Name

A2 and A1 Pod Numbers

[15:5] Channel 15 through Channel 5 ("**********.....")

[5,2] Channel 5 and Channel 2 (".....*..*..")

When setting up the ASCII file a comma (",") separates individual channels, while a colon (":") creates a range of channels.

The following provides an explanation of how to setup and import ASCII files into a logic analysis system.

Setting Up ASCII Files

NOTE: If the analyzer is in state mode with the Clock Setup Mode set to demultiplexer, slave pods will appear with an "S" in front of the pod designation.

For example:

Label1;SA2[5] reads as Label1 maps to Slave Pod A2 Channel 5.

Individual channels

Label1;A2[5] Label1 maps to Pod A2, Channel 5

Multiple channels

Label1;A1[5:2,0] Label1 maps to Pod A1, Channel 5 through Channel 2 and Channel 0.

Individual channels on different pods

Label1;A2[1];A1 [0] Label1 maps to Pod A2, Channel 1 and Pod A1, Channel 0.

Multiple channels on different pods

Label1;A3[15:5]; A2[5];A1[6] Label1 maps to Pod A3, Channel 15 through Channel 5,

Pod A2 Channel 5, and Pod A1 Channel 6.

Clocks

Label1;CK[AK] Label1 maps to Slot A Clock K.

See Also

"Importing ASCII Files" on page 58

"Exporting ASCII Files" on page 58

"Termination Adapter" on page 60

"E5346A High Density Adapter" on page 61

Considerations when creating an Import file.

When updating a label ensure that the label is turned on. Labels that are not active will not be updated.

Other considerations:

- Maximum of 126 labels
- Maximum of 20 characters per label. (The system truncates after 20 characters.)
- Maximum of 32 channels per label.

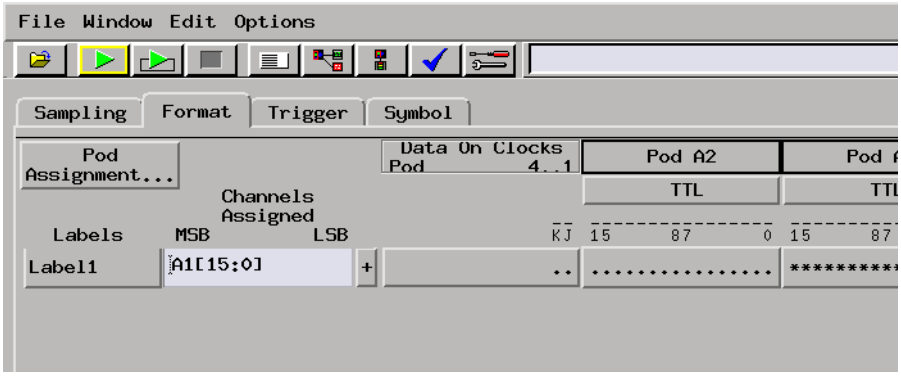
Exporting ASCII Files

1. Select the Format tab.
2. Select File, then select *Export Label...*

Importing ASCII Files

Shown below is the default setup:

```
Label1;A1[15:0]
```



To Import an ASCII file.

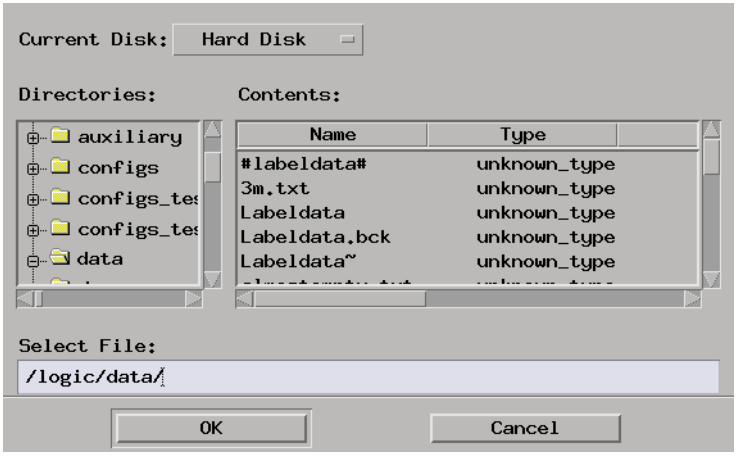
1. Create an ASCII file for importing into the logic analysis system. For example:

   Lab2;A2[15:10,6,3]

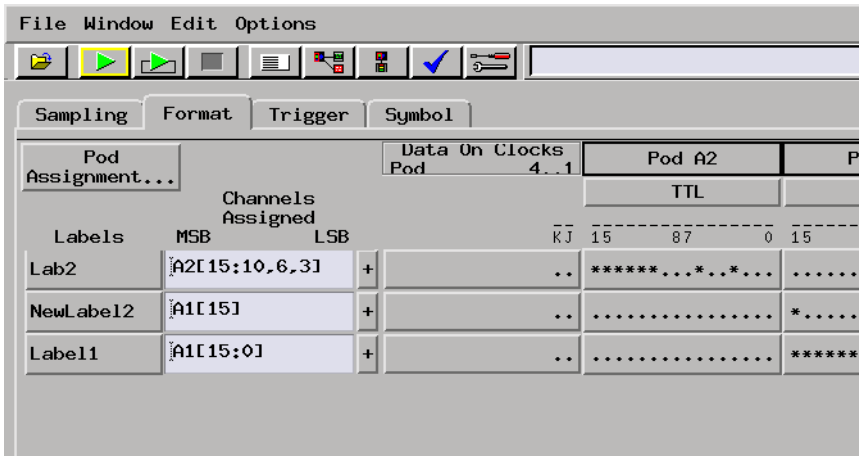
   NewLabel2;A2[15]

   Label1;A1[15:0]

2. Select the Format tab.
3. Select File, then select *Import Labels...*
4. Select the ASCII file you created.

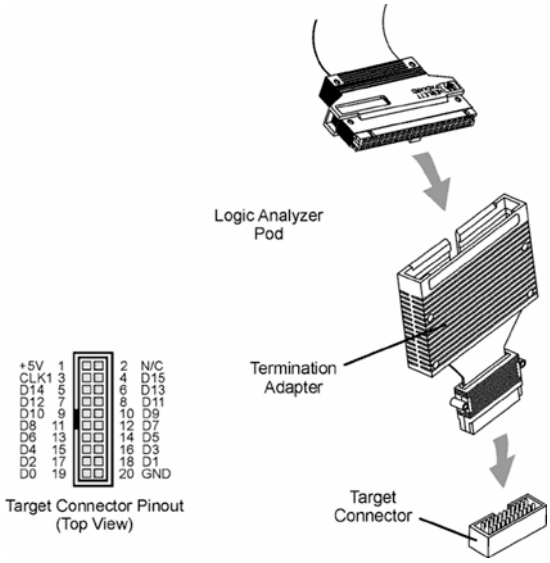


5. Select OK.



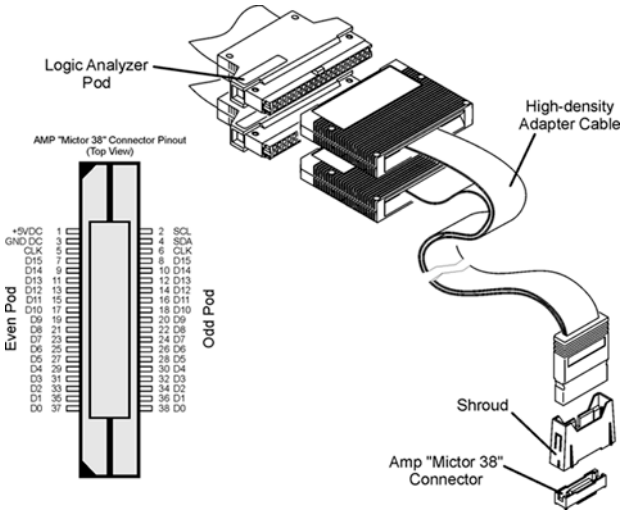
Termination Adapter

The logic analyzer cable must have the proper RC network at its input in order acquire data correctly. The Termination Adapter incorporates the RC network into a convenient package. It also reduces the number of pins required for the header on the target board from 40 pins to 20.



E5346A High Density Adapter

The E5346A high-density adapter provides a convenient and easy way to connect an Agilent logic analyzer to the signals on your target system for packages that are difficult to probe. An Amp "Mictor 38" connector must be installed on your target system board.



Mapping Connector Names

1. Select the Format tab.
2. Select File, then select *Import Netlist*.
3. Select *Next* to go to the Mapping Connector Names dialog.
4. Enter a connector name from the Netlist.
5. Select the type of adapter.
6. Select the logic analyzer pods that are plugged into the adapters.
7. To map more connector names, select *Add Connector* and repeat the steps above.
8. Select *Next*.

Import the Net List File

1. Select *Select Netlist File*.

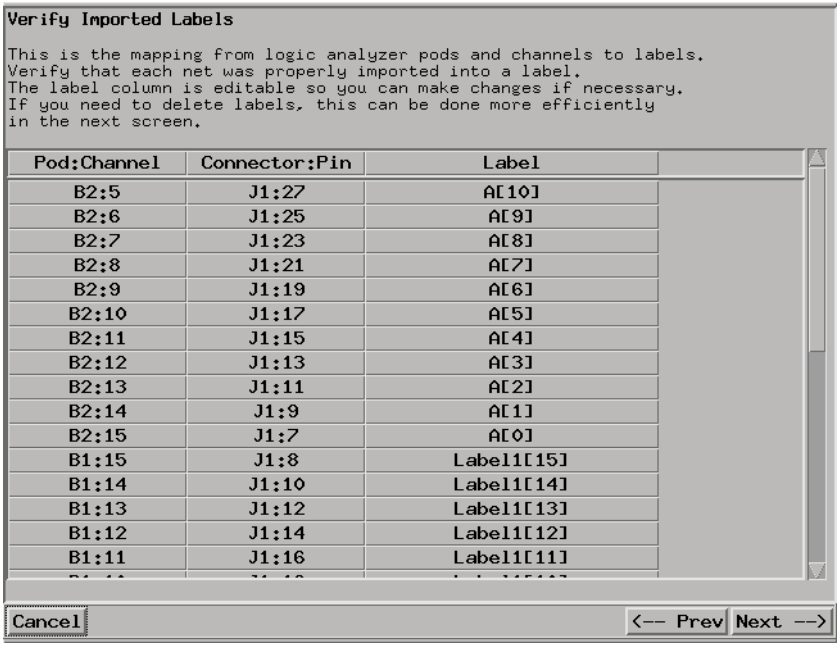
2. Select the file from the File Selection dialog box.
3. Select *OK*
4. Select *Next*

Verify Net to Label Mapping

1. Verify that each net was properly imported into a label.
2. Select *Next*

The list provided, see example below, is the mapping from the logic analyzer pods and channels to labels.

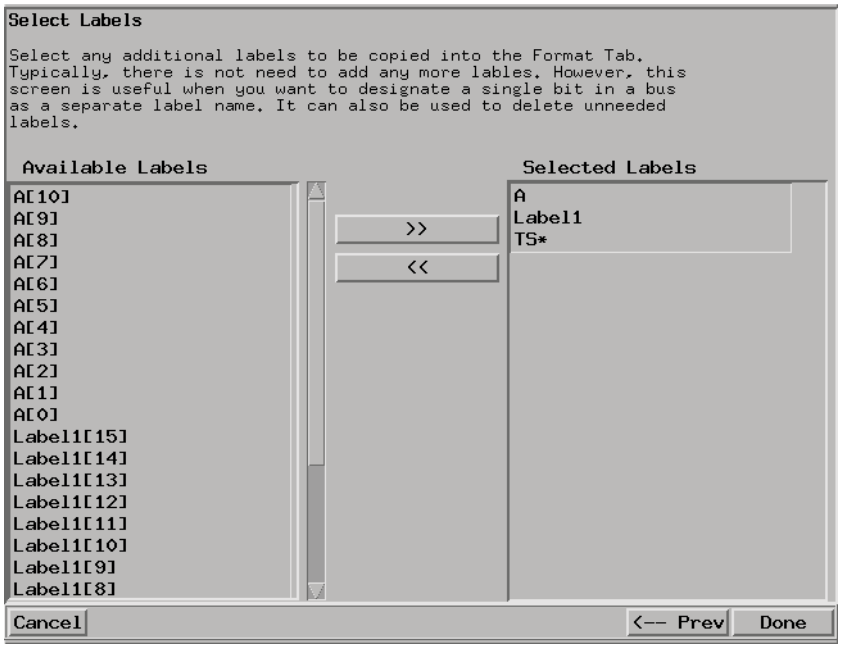
The label column is editable so you can make changes if necessary. If you need to delete labels, this can be done more efficiently in the next dialog box.



Select/Create Interface Labels

Select any additional labels to be copied into the Format tab. Typically there is no need to add any more labels. However, this screen is useful when you want to designate a signal bit in a bus as a separate label name. It can also be used to delete unneeded labels.

Once you have completed editing the labels, select *Done*.



Activity Indicators

Activity indicators are the arrows and dashes associated with the pods. They appear in various places, primarily above the column of bit assignment fields in *Format* and in the Clock Setup area.

When the logic analyzer is properly connected to an active target system, you see arrows in the Activity Indicator displays for each

channel. An active channel looks like .

A dash at the top of the activity indicator display indicates that the signal connected to that channel is electrically high (above the threshold voltage). A dash at the bottom indicates that the signal is electrically low. An arrow indicates that the signal is transitioning.

Activity indicators are not affected by label polarity. (see page 68)

You can use these indicators to check whether there is proper probe connection: *bits* that are stuck high or low may not be properly connected. You can also verify that the signals in your target system are active: bits that are stuck high or low are not crossing the threshold voltage.

Activity indicators are not displayed during an *acquisition*.

See Also

"Setting the Pod Threshold" on page 72

Logic Analysis System and Measurement Modules Installation Guide for details on probing

Assigning Pods to the Analyzers

The *Pod Assignment...* button under *Format* can be used to start the second analyzer on the *module* and to assign pod pairs.

To Assign Pods to an Analyzer

1. In the *Format* tab, select the *Pod Assignment...* button.
   The Pod Assignment window appears.
2. Drag a pod pair and drop it below the analyzer that you want to assign it to.
3. For pods that you do not want to use, *drag and drop* them in the Unassigned Pods area. This preserves memory depth when *count* is turned on.

NOTE: When both analyzers are turned on, pods 1/2 and 5/6 of the master card can not be assigned to the same analyzer.

Pods can only be assigned on a per-pair basis to either of the two analyzers. Each pod pair has two *clock channels*, but only the clock channels of pods on the *master card* can be used in the analyzer's clocking setup. These pods do not need to be assigned to that particular analyzer, however.

To turn on an analyzer that is off, select the *Off* option button and choose *State* or *Timing*. (Only one analyzer at a time can be set to *Timing*.) A second analyzer window appears after a pause for setup.

Data On Clocks Display

The Data On Clocks display column, to the left of the pods' bit reference line, is a display of all clock inputs available as data channels in the present configuration. This includes those clocks on expander *cards*, which cannot be used in the clock setup.

To use a *clock channel* as a *data channel*, assign the clock bit to a *label*. Labels containing clock bits cannot be used in *range terms* where the clock bits span more than one pod pair.

Activity indicators above the clock identifier show signal activity.

See Also

"Assigning Bits to a Label" on page 66

"Activity Indicators" on page 64

Assigning Bits to a Label

The *bits* in a label correspond to the physical logic analyzer probe *channels*. When you run the analyzer, data is gathered on all bits (channels) that are assigned to *labels*. Unassigned bits are inactive.

(*) (asterisk) indicates an assigned bit.
(.) (period) indicates an unassigned bit.
(R) indicates an assigned bit in a reordered label.

To Assign Bits

1. Select the bit assignment label to the right of the label name you want to define.
   Each bit assignment label corresponds to the data pod which is listed above it.
2. Either choose one of the predefined groups, or *Individual*.
3. In *Individual*, select the bits to toggle them between an asterisk and a period.

NOTE: Labels can have a maximum of 32 channels assigned to them.

Bits assigned to a label are numbered from right to left. The least significant assigned bit on the far right is numbered 0. The next assigned bit to the left is numbered 1, and so on. Labels can contain bits that are not consecutive; however, bits are always numbered consecutively within a label. Above each column of bit assignment fields is a bit reference line that shows you the bit numbers and activity indicators.

See Also

"To reorder bits in a label" on page 69

"Activity Indicators" on page 64

Inserting and Deleting Labels

To Insert Additional Labels

1. Select the *label* name button that you want to insert another label next to.
2. Choose *Insert before...* or *Insert after...*.

To Delete Labels

1. Select the button of the label name that you want to delete.
2. Choose *Delete*.

The bit assignments of deleted labels are not saved. You can make a label inactive but not lose its assignment by turning it off (see page 68)

instead.

Turning Labels On and Off

You may want to turn off *labels* that you have created so that the label is not displayed. When a label is turned off, its name and bit assignments are preserved.

To Turn Off a Label

1. Select the button of the label name that you want to turn off.
2. Choose *Label [ON]* to toggle it off.
   If the label is the only one, it cannot be turned off or deleted. If there is more than one label but it is the only one on, turning it off turns the first label back on.

To Turn On a Label

1. Select the button of the label name that you want to turn on.
2. Choose *Label [OFF]* to toggle it on.

To Display a Label that was Off

1. Turn on the label.
2. At the bottom of the window, select the *Apply* button.
   The label's data appears in the display windows.

Label Polarity

The analyzer uses the label polarity to identify patterns when *triggering* and displaying data.

To change the label polarity, select the polarity field, which toggles between positive (+) and negative (-). Positive polarity means that a high voltage is a logical "1". Negative polarity means that a high voltage is a logical "0".

Changing the label polarity after you have set up other parts of the measurement changes these things:

- "1" and "0" values flip in trigger *terms*

- waveforms and bus values (where shown) invert in the Waveform Display tool
- "1" and "0" values flip in the Listing Display tool

Changing the label polarity does not change these things:

- Edge definitions for clock setup and *edge terms*
- Symbol definitions for the logic analyzer

The default polarity for all labels is positive (+). The various display tools, in particular the Waveform Display tool, all show logical values and are affected by polarity changes.

NOTE: Negative logic is rare in circuits. The main exception at this time is RAMBUS.

To reorder bits in a label

In cases where buses in the device under test haven't been probed with consecutive logic analyzer channels, you can reorder the bits in a label.

1. In the Format tab, select the label button whose bits you want to reorder.
2. Choose *Reorder bits*.
3. In the Change Bit Order dialog:
   - To reorder the bits individually, enter the bit that the probe channel should be mapped to.
   - To swap the high and low order bytes or words, select the button *Big Endian to Little Endian* at the bottom of the dialog.
   - To return to sequentially ordered bits, select the button *Default Order* at the bottom of the dialog.
4. Select the *OK* button.

   The label now shows an "R" to indicate that the assigned bit has been reordered.

NOTE: Labels with reordered bits cannot be used as *range terms* or <, <=,>, >= in triggers.

Labels: Mapping Analyzer Channels to Your Target

Labels group and identify related *channels*, such as buses, in a way that is relevant to your system under test.

A single label can include data and clock channels from more than one pod pair, but these cannot be included in *range terms* in the *trigger specification*.

You can define 126 labels per analyzer. Each label can contain up to 32 channels per label.

To Define a Label

1. (Optional) Select the label name button and choose *Rename*
2. Enter a new name and select the *OK* button.
3. Assign bits (see page 66) to the label.
4. If necessary, insert more labels (see page 67) in the list.

See Also

"Assigning Bits to a Label" on page 66

"To reorder bits in a label" on page 69

"Inserting and Deleting Labels" on page 67

"Turning Labels On and Off" on page 68

"Label Polarity" on page 68

Setting Up the Pod Clock

There is one Pod Clock field for each pod in the machine. It only

becomes visible when the Clock Setup under the *Sampling* tab is set to *Master/Slave* or *Demultiplex*. The Pod Clock field is located just below the Pod Threshold in *Format*.

The Pod Clock field is set to *Master Clk* by default. Use the Pod Clock field to indicate if the pod's data is to be strobed into memory by the master clock, slave clock, or both, in the demultiplex mode.

When the Pod Clock is set to *Demultiplex*, only one pod of a *pod pair* is usable. That pod latches data on both the master and slave clocks, so it appears twice in the label area. To select which pod of a pod pair you want to demultiplex, select the Pod Field.

See Also

"Performing Clock Setup (State only)" on page 41

"Clock Modes (State only)" on page 42 for details on slave and demultiplex clocks

"Pod Selection" on page 71

Pod Selection

The Pod field in *Format* identifies which pod of a *pod pair* the settings of the bit assignment field, pod threshold field, and pod clock fields effect. Most of the time it is simply informational. The exceptions are noted below.

Half-Channel Mode

In the half-channel mode, the Pod field becomes a button that you can use to select which pod of a pod pair all pod settings apply to. In the full-channel modes, this field is simply an identifier and is not selectable.

Demultiplex Clock Mode

In the demultiplex clock mode, use the pod field to select which pod of a pod pair is to be used to sample data. In the master or slave clock modes, this field is simply an identifier and is not selectable.

See Also

"Setting the Acquisition Mode" on page 41

"Performing Clock Setup (State only)" on page 41

Setting the Pod Threshold

The pod threshold is a voltage level which the data must cross before the analyzer recognizes it as a change in logic levels. You can specify a threshold level for each pod. The level specified for the master pod that includes the clock is also used for the clock threshold.

Standard

1. In the *Format* tab, select the threshold button located just below the pod name.
2. In the Pod threshold dialog, choose one of the standard threshold options:
   - TTL -- The threshold level is +1.50 volts.
   - LVTTL -- The threshold level is +1.40 volts.
   - SSTL2 -- The threshold level is +1.50 volts.
   - SSTL3 -- The threshold level is +1.25 volts.
   - LVCMOS 1.5v -- The threshold level is +0.75 volts.
   - LVCMOS 1.8v -- The threshold level is +0.90 volts.
   - LVCMOS 2.5v -- The threshold level is +1.25 volts.
   - LVCMOS 3.3v -- The threshold level is +1.65 volts.
   - CMOS 5.0v -- The threshold level is +2.50 volts.
   - ECL -- The threshold level is -1.3 volts.
   - LVPECL -- The threshold level is 2.00 volts.
3. If you don't want the change to apply to all pods, deselect the checked box next to *Apply settings to all pods*.
4. Select the *Close* button.

User Defined

When *User Defined* is selected, the threshold level is selectable from -6.0 volts to +6.0 volts.

NOTE: The logic analyzer requires a minimum voltage swing of 500 mV at the probe tip to recognize changes in logic levels.

NOTE: The threshold voltage specified also applies to the pod's clock input.

State Clock Setup/Hold (State only)

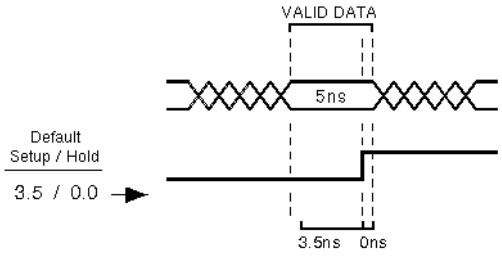
Setup/Hold in Format adjusts the relative position of the clock edge with respect to the time period that data is valid. It is only available when the analyzer is set up for a *state measurement*.

To Change Clock Setup/Hold

1. Select the *Setup/Hold* button.
2. For each pod pair, choose a Setup/Hold selection from the selection list.
3. Select the *OK* button.

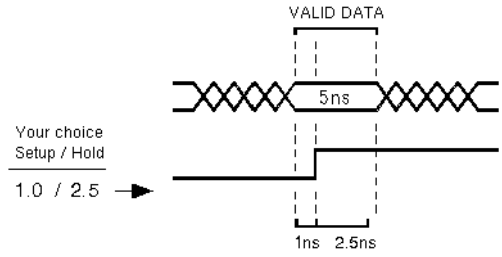
With a single clock edge assigned, the choices range from 3.5 ns Setup/0.0 ns Hold, to 0.0 ns Setup/3.5 ns Hold. With both edges of a single clock assigned, the choices are from 4.0 ns Setup/0.0 ns Hold, to 0.0 ns Setup/4.0 ns Hold. If multiple clocks are assigned, the choices range from 4.5 ns Setup/0.0 ns Hold, to 0.0 ns Setup/4.5 ns Hold.

The relationship of the clock signal and valid data under the default setup and hold is shown in the figure below for a generic logic analyzer.



Default Setup and Hold

If the relationship of the clock signal and valid data is such that the data is valid for 1 ns before the clock occurs and 3 ns after the clock occurs, you will want to use the 1.0 setup and 2.5 hold setting.



Clock Position in Valid Data

The Trigger Tab

The Trigger tab is used to set up a sequence that tells the analyzer when to capture data. The key event is the *trigger*.

The Trigger tab has two main areas:

- On top, tabs of functions and controls to build your trigger
- Beneath the tabs, the current trigger sequence.

Some controls are also located in the logic analyzer window's menu bar.

- "Understanding Logic Analyzer Triggering" on page 76
- "Setting Up a Trigger" on page 78
- "Inserting and Deleting Sequence Steps" on page 79
- "Editing Sequence Steps" on page 80
- "Setting Up Loops and Jumps in the Trigger Sequence" on page 81
- "Saving and Recalling Trigger Sequences" on page 82
- "Clearing Part or All of the Trigger" on page 83

See Also

"Overview of the Trigger Sequence" on page 83

"Trigger Functions" on page 84

"Working with the User-level Function" on page 91

"Defining Resource Terms" on page 94

"Trigger Position Control" on page 53

"Sample Period (Timing Only)" on page 45

"Tagging Data with Time or State Tags (State Only)" on page 101

"Arming Control" on page 101

Understanding Logic Analyzer Triggering

What is a Trigger (see page 76)

What does "Trigger Position" Mean (see page 76)

What can be Used to Specify a Trigger (see page 77)

When to use a Combination, a Branch, or a Level (see page 77)

What is a Trigger

In simplest terms, a trigger is an event that tells the logic analyzer to finish filling its acquisition memory. The memory functions like a conveyor belt: new samples are always coming in, and old samples "falling off" (being overwritten). The logic analyzer has room for 4K samples. When this is full, the only way to fit in new data is to discard the old.

After you specify your trigger sequence and press run, the logic analyzer searches incoming data for events in the trigger sequence. Using the conveyor belt metaphor again, it is like someone tending the conveyor belt who has been told to stop the belt when a certain sample is seen.

The trigger is *not* like an oscilloscope trigger. Logic analyzers trigger only once per run, even when more than one sample matches the trigger event. Logic analyzer trigger events are like special switches to stop the evaluation process and just fill memory.

What does "Trigger Position" Mean

Because the logic analyzer is continually looking at data from your target system after you select *Run*, and because the trigger is a single event, you can arrange to collect data relative to it. It is like the person running the conveyor belt is told to stop the belt when the special sample reaches a certain position.

The default trigger position is in the middle. This means there are

approximately as many samples before the trigger as after. You can also arrange for the trigger to be at the beginning of the "conveyor belt" (acquisition memory), the end, or any percentage along it.

What can be Used to Specify a Trigger

The trigger sequence can be as simple as one event to look for, or a complicated set of branching levels that loop back and jump around. Both types of triggers use a small set of standard resources.

Pattern Pattern terms are things such as `ADDR=0880` or `R/W=1`. Generally they represent values on buses. You can set patterns to match imprecise events, too. See “Using Bit Pattern Terms” on page 95.

Range Range terms match a range of values on a label or bus. For more detail, see “Using Range Terms” on page 96

Timer Timers are started in one sequence level and checked in another. They act like stopwatches. See “Using Timer Terms” on page 98 for more information.

Edge Edge terms are similar to edges in oscilloscopes. They are only available for some types of measurements. Edge terms can check for edges on more than one signal at a time, but not all edges have to occur at the same time. To require that, combine edge terms with *AND*s.

Combinations of Terms To check for more than one type of thing happening *in the same sample*, combine terms within a sequence level using *AND* and *OR*. There are restrictions on exactly which terms can be ANDed and which ORed. The restrictions are covered in “Using Combinations of Terms” on page 99.

When to use a Combination, a Branch, or a Level

To check for simultaneous occurrences, use combinations of terms. All the events described by the terms must happen in the same sample.

To take different actions depending on which events happen in a sample, use *branches* within a sequence level. A branch functions like a set of "if statements" in programming. The sample is checked against

all branches, and the first branch that matches is taken. See "Setting Up Loops and Jumps in the Trigger Sequence" on page 81 for more on branching.

To look for a sequence of events (for example, first look for a memory reference on ADDR, then a certain value on DATA, and when IRQ goes low, trigger) use different sequence levels. When a sample matches the event described in a sequence level, the analyzer goes to the next sequence level and compares the rest of the incoming events. When the logic analyzer reaches the trigger level and finds a sample that matches the trigger event you specify, the logic analyzer triggers.

Setting Up a Trigger

When setting up a *trigger sequence*, you typically trigger first on a simple pattern or edge. From that point, you execute an iterative process of adding or fine-tuning sequence steps until the analyzer consistently triggers at the desired point.

To Set Up a Trigger

1. Define resource terms. (see page 94)
2. Select the most appropriate trigger function and select the Replace button. (see page 78)
3. Select the sequence level number button and choose Edit. (see page 80)
4. If necessary, insert and edit additional sequence steps. (see page 79)

See Also

"Trigger Functions" on page 84

"Working with the User-level Function" on page 91

"Overview of the Trigger Sequence" on page 83

Selecting a Function to Match Trigger Conditions

Review the list of trigger functions and select the one that matches the event you are looking for. In most cases one of the predefined functions provides a good starting point. If none of the predefined trigger

functions match choose *User level* at the end of the list.

For a picture corresponding to the trigger function, select the function from the list. The area to the right shows a picture of the function's effect. The function itself is not inserted into the trigger sequence unless you select the *Replace* or *Insert* button.

See Also

"Trigger Functions" on page 84

Inserting and Deleting Sequence Steps

NOTE:

For *state measurements*, the last level of the *trigger sequence* is always a Store level. It cannot be deleted. For *timing measurements*, the last level must contain the *TRIGGER* action.

To Insert Sequence Steps

1. Select the sequence level that you want to insert other steps around.
2. Select a trigger function from the list in the *Trigger Functions* tab.
3. Choose the *Insert Before* or *Insert After* button.

To Delete Sequence Steps

1. Select the sequence step that you want to delete.
2. Select the *Delete* button.

Trigger Sequence Editing Options

When you select a sequence level, you see a selection menu with choices that allow you to modify the sequence. Choose the option you want from the choices below.

- Edit

  Changes the contents of the sequence step. You can change the resource *terms* or other assignment fields, such as durations and occurrences.

- Copy

  Copies the currently selected step. When you copy a level, the new level contains the same function as the original.

- Replace

  Replaces the currently selected level with the currently highlighted trigger function.

- Delete

  Deletes the level that is currently selected.

- Insert Before / Insert After

  Inserts an additional step before or after the selected step.

- Trigger Level (State only)

  Makes the current step the trigger level.

- Default (State only)

  Only appears in the menu for the last step in a *state* trigger sequence. The last state always stores to fill memory. Default returns the state to *Store anystate*.

Editing Sequence Steps

You can modify the contents of a step in the *trigger sequence* by *editing* it.

To Edit a Sequence Step:

1. Select the sequence level number button and choose *Edit...*.
2. Select a term name to choose a different term.
   You can also choose negations of *terms* and combinations of terms.
3. Set the values of the other fields, such as durations and occurrence counts.
4. Select the *Close* button to close the sequence step editing dialog.

To set the value of a term used in a sequence step, see "Defining Resource Terms" on page 94.

If you are editing a user-level function, refer to:

"Setting Pattern Durations and Occurrence Counts" on page 93 (Timing

Only)

"Using Occurrence Counters" on page 93 (State Only)

"Using Timer Terms" on page 98

"Setting Up Loops and Jumps in the Trigger Sequence" on page 81

Setting Up Loops and Jumps in the Trigger Sequence

To set up loops and jumps in your *trigger sequence*, use *Branches*. Branches are available in most of the *Trigger Functions*. You may need to break down the functions (see page 90) in order to set up branches.

NOTE: If either the < or the > durations are selected, only the primary *Find* or *Trigger on* selection is available. If the *occurs* duration is selected, the secondary (Else on) branch becomes available. If the *Find* field is an edge, only *occurs* is available.

To Set Up a Branch or Loop

1. Select the *Find* button and choose the term that you want to branch on. If this term is found in the incoming data, the analyzer will follow this branch and go to the next trigger sequence level.
2. If you want a secondary branch, select the *Else on* button and choose the term that you want to branch on.
3. Select the *goto level* option button and select the sequence level that you want to branch to.

If the first branch is taken, the analyzer goes to the next level. If the term in the first branch is not found, the analyzer immediately evaluates the *Else on* secondary branching term. The analyzer only *triggers* when the *TRIGGER* primary branch is taken.

If the *Else on* term is found, the secondary branch is taken to the designated sequence level, and the occurrence counter is reset even if the branch loops back to the same level. If the *Else on* term is not

found, the analyzer stays at the same sequence level until one of the two branches is found. If both branches are found true at the same time, the first branch is taken.

See Also

"Breaking Down and Restoring Functions" on page 90

"Branches Taken Stored / Not Stored (State only)" on page 94

Saving and Recalling Trigger Sequences

You can save a *trigger sequence* independently of configuration files within a session by using *Save/Recall*. Recalling the stored trigger sequence can change the trigger arming, memory depth, and trigger position as well as the trigger sequence and term definitions. The trigger sequence specification will not change the acquisition mode (full channel vs. half channel).

The Agilent Technologies 16550A logic analyzer can hold up to 10 trigger sequence specifications per mode per *machine*, for a total of 40 specifications. When you exit your Agilent Technologies 16700A/B *session*, the trigger sequences are cleared. They can be saved across sessions or be shared across logic analyzers as part of a configuration file, however.

To Save a Trigger Sequence

1. In the *Trigger* tab, select the *Save/Recall* subtab.
2. Enter a descriptive name for the trigger sequence in the Title field.
3. Select the *Save* button.
4. Select a memory location to store the trigger sequence in.

To Recall a Trigger Sequence

1. In the *Trigger* tab, select the *Save/Recall* subtab.
2. Select the *Recall* button.
3. Choose the trigger sequence that you want.

If one of the settings in the recalled trigger sequence conflicts with the acquisition mode, it will be set to the closest setting for that mode.

Clearing Part or All of the Trigger

To Clear Part or All of the Trigger:

1. Select the *Trigger* tab.
2. Select *Clear* from the *menu bar*.
3. Choose the option you want from the choices described below:
   - All

     Clears sequence steps, resource *terms*, resource term names, and trigger position back to their default values. Turns on Count Time.
   - Sequence Levels

     Resets the *trigger sequence* to the default sequence for the analyzer acquisition mode.
   - Resource Terms

     Resets all resource term assignment fields, including *labels* and values, back to their default values. The default is usually for all terms to be on the first label, with value *XXXX*.
   - Resource Term Names

     Resets all the resource term names to their default values, *Pattern1* through *Pattern9* and *Patt10*.
   - Save/Recall Memories

     Deletes all saved trigger sequence specifications.

Overview of the Trigger Sequence

The *trigger sequence* is a sequence of steps that control the path that the analyzer takes to find the trigger event. The path taken resembles a

flow chart, with each step in the sequence being an opportunity to direct the analyzer's selection. You can edit the overall trigger sequence by inserting or deleting sequence steps (see page 79).

Each step in the sequence is either a predefined trigger function (see page 84) or a user-level step (see page 91). Both the trigger functions and user-level steps contain variables that you define. The variables, called resource terms (see page 94), represent edges and bit patterns in the data.

When you run the analyzer, it searches for a match between the resource term values and the measurement data. When a match is found, the sequencing continues to the next step, loops back (see page 81) to a previous step, or jumps ahead to another step. Eventually, a path of "true" steps leads to the trigger event.

Each macro uses one or more of the analyzer's internal sequence levels (see page 91). Each user-level step uses one internal sequence level.

See Also

"Understanding Logic Analyzer Triggering" on page 76 for more detail

"Setting Up a Trigger" on page 78 for actual steps

Trigger Functions

Trigger functions provide a simple way to set up the analyzer to *trigger* on common events and conditions. A library of functions is available for both *state* and *timing* measurements.

NOTE: Each trigger function requires at least one internal sequence level (see page 91), and in some cases, multiple levels. The number of levels used by each function is described in the references below.

Timing Trigger Functions

"Basic Timing Trigger Functions" on page 85

"Pattern/Edge Combinations" on page 86

"Time Violations" on page 87

"Timing User Level" on page 85

State Trigger Functions

"Basic State Trigger Functions" on page 88

"Sequence-Dependent Trigger Functions" on page 88

"Time Violations" on page 89

"State User-level" on page 87

See Also

"Setting Up a Trigger" on page 78

"Defining Resource Terms" on page 94

"Breaking Down and Restoring Functions" on page 90

Timing User Level

The *User level* trigger function allows you to create a custom *trigger sequence* using *terms*, a comparison function, and a secondary branch if the comparison function is *occurs*. This trigger function uses one internal sequence level.

See Also

"Working with the User-level Function" on page 91 for more information on the user-defined mode.

Basic Timing Trigger Functions

The following basic trigger functions are found in *Trigger Functions* when the analyzer is in *timing* mode. Each function uses one internal sequence level.

- Find edge.
  This function becomes true when the edge you have designated is seen. It uses one internal sequence level.

- Find anystate n times.
  This function becomes true with the nth state it sees. It uses one internal sequence level. It is equivalent to having the analyzer wait in the sequence level for (n x Sample Period) seconds.

- Find nth occurrence of an edge.
  This function becomes true when it finds the designated occurrence of an edge you have designated. Note that the 500-MHz trigger sequencer may not count edges that occur closer than 2 ns. This function uses one internal sequence level.

- Find pattern present/absent for > duration.
  This function becomes true when it finds a pattern you have designated that has been present or absent for greater than or equal to the set duration. It uses one internal sequence level.

- Find pattern present/absent for < duration.
  This function becomes true when it finds a pattern you have designated that has been present or absent for less than the set duration. It uses five internal sequence levels.

Pattern/Edge Combinations

The following trigger functions are found in *Trigger Function* when the analyzer is in *timing* mode. These predefined functions use a pattern, edge, or a combination of both as the *trigger* element. The functions use either one or two internal sequence levels.

- Find edge AND pattern.
  This function becomes true when a selected edge is seen within the time window defined by a pattern you have designated. It uses one internal sequence level.

- Find pattern occurring too soon after edge.
  This function becomes true when a pattern you have designated is seen occurring within a set duration after a selected edge is seen. It uses two internal sequence levels.

- Find pattern occurring too late after edge.
  This function becomes true when one edge you have selected occurs, and for a designated period after that first edge is seen, a pattern is not seen. It uses two internal sequence levels.

Time Violations

The following trigger functions are found in *Trigger Functions* when the analyzer is in *timing* mode. These trigger functions are specifically tailored to *trigger* on events occurring out of a predefined time range. They use either one or two internal sequence levels.

- Find width violation on a pattern/pulse.
  This function becomes true when the width of a pattern violates minimum and maximum width settings you have designated. It uses one internal sequence level.

- Find 2 edges too close together.
  This function becomes true when a second selected edge is seen occurring within a period you have designated after the occurrence of a first selected edge. It uses two internal sequence levels.

- Find 2 edges too far apart.
  This function becomes true when a second selected edge occurs beyond a period you have designated after the first selected edge. It uses two internal sequence levels.

- Wait t seconds
  This function becomes true after a period you have designated has expired. It uses one internal sequence level.

State User-level

The User-level trigger function allows you to create a custom *trigger sequence* using *terms*, a comparison function, and a jump or loop. This

trigger function uses one internal sequence level.

See Also

"Working with the User-level Function" on page 91 for more information on the user-defined mode.

Basic State Trigger Functions

The following basic trigger functions are found in *Trigger Functions* when the analyzer is in *state* mode. Each macro uses one internal sequence level.

- Find Pattern n times.
  This function becomes true when it sees a pattern you have designated occurring a designated number of times. The pattern may occur consecutively, but does not have to. It uses one internal sequence level.

- Find anystate n times.
  This function becomes true with the nth state it sees. It uses one internal sequence level. It is equivalent to *Wait n external clock states*.

- Find pattern2 occurring immediately after pattern1.
  This function becomes true when the first pattern you have designated is seen immediately followed by a second designated pattern. It uses two internal sequence levels.

- Find pattern n consecutive times.
  This function becomes true when it sees a pattern you have designated occurring a designated number of consecutive times. It uses one internal sequence level.

Sequence-Dependent Trigger Functions

The following trigger functions are found in *Trigger Functions* when the analyzer is in *state* mode. These functions each *trigger* on a particular sequence of events.

- Find too few states between pattern1 and pattern2.
  This function becomes true when a designated pattern1 is seen, followed by a designated pattern2, and with fewer than a selected number of states occurring between the two patterns. It uses four internal sequence levels.

- Find too many states between pattern1 and pattern2.
  This function becomes true when a designated pattern1 is seen, followed by more than a selected number of states, before a designated pattern2. It uses two internal sequence levels.

- Find n-bit serial pattern.
  This function becomes true when a specified serial pattern of n bits is found on the analyzed line. This function uses one internal sequence level for each bit specified in the trigger sequence.

- Find pattern2 n times after pattern1, before pattern3 occurs.
  This function becomes true when it first finds a designated pattern1, followed by a selected number of occurrences of a designated pattern2. In addition, if a designated pattern3 is seen anytime while the sequence is not yet true, the sequence starts over. This includes if pattern2's nth occurrence is at the same time as pattern3, the sequence starts over. It uses two internal sequence levels.

Time Violations

The following trigger functions are found in *Trigger Functions* when the analyzer is in *state* mode. These predefined functions are specifically tailored to *trigger* on events occurring out of a predefined time range. These functions use either one or two internal sequence levels.

- Find pattern2 occurring too soon after pattern1.

This function becomes true when a designated pattern1 is seen, followed by a designated pattern2, and with less than a selected period occurring between the two patterns. It uses two internal sequence levels.

- Find pattern2 occurring too late after pattern1.
  This function becomes true when a designated pattern1 is seen, followed by at least a selected period, before a designated pattern2 occurs. It uses two internal sequence levels.

- Wait n external clock states.
  This function becomes true after a number of user clock states you have designated have occurred. It uses one internal sequence level.

Breaking Down and Restoring Functions

When you break down a trigger function, you gain access to all the resource assignment fields and branching options. You can change these fields to change the structure of the *trigger sequence*. You might need to do this to create a custom trigger sequence or to add jumps.

To Break Down Trigger Functions

1. Select *Modify* in the Trigger window *menu bar*.
2. Choose *Break down functions*. The trigger sequence area changes to show the entire trigger sequence as a series of user-level steps.

The contents of broken down functions are displayed in the long form used in a user-level sequence step. If the function uses two of the analyzer's internal sequence levels, (see page 91) both levels are separated out and displayed in the trigger sequence area.

To Restore Functions

1. Select *Modify* in the Trigger window menu bar.
2. Choose *Restore functions*.

Use *Restore functions* to restore *all* functions to their original

structure. Note that when the functions are restored, all changes are lost and any branching that is part of the original structure is restored.

See Also

"Working with the User-level Function" on page 91 for information on working with functions that are broken down.

How the Internal Sequence Levels Are Used

The analyzer has internal sequence levels that it uses to make up the *trigger sequence*. There are a total of 16 sequence levels available.

The actual number of levels used in a trigger sequence can vary depending on whether you elect to use predefined trigger functions or use the user-defined steps (see page 91) to construct a more custom trigger sequence.

When you use user-defined steps, all of the internal sequence levels are available. Each user-defined sequence level corresponds to one internal level. The only instance where multiple levels are used is when the < duration is assigned.

When you use predefined trigger functions (see page 84), more than one of the internal sequence levels may be required for a single trigger function. Even though some trigger functions use multiple sequence levels, trigger functions are easier to use, and they are the most efficient way to construct a trigger specification.

Working with the User-level Function

NOTE: Before you begin to set up user-level sequence steps, note that in most cases one of the predefined trigger functions (see page 84) will work.

You might need to set up a user-level sequence step to accommodate a condition not covered by the functions, or if you need to set up additional loops and jumps in the sequence. Each user-level sequence step has a "fill-in-the-blanks" type statement. You use resource *terms* to fill in the statement with the appropriate values.

To Set Up a User-Defined Macro

1. In the *Trigger Functions* tab, select *User-level* at the end of the list.
2. Select the *Replace* or *Insert* button.
3. Select the new sequence level number button; choose *Edit*.
   A Trigger Sequence Step window appears.
4. For state measurements, select the *While storing* button; choose the resource term.
   The values represented by this term will be stored in memory while the logic analyzer is evaluating target system data against this step. The default, *anystate*, stores everything. *Nostate* stores nothing.
5. Select the *Trigger on* button or the *Find* button; choose the resource term.
6. In timing measurements, you have the choice of an occurrence counter or duration. To use the secondary branch, you must set this to *occurs*.
7. Set the occurrence field or duration.
8. If you want to use a loop or jump:
   a. Select the *Else on* button; select resource term.
   b. Select the *goto level* option button; choose a sequence level.
9. If you want to use a timer:
   a. In the appropriate sequence level, start the timer by selecting the *Off* option button and choosing *Start*. Timers do not start automatically.
   b. In the *Timer* tab, assign a time value you want to check against.
   c. To check the value, assign the timer to a resource field either on its own, or as part of a combination.

For more information on the functions available in a user-defined step, refer to:

- “Defining Resource Terms” on page 94
- “Setting Pattern Durations and Occurrence Counts” on page 93 (Timing Only)
- “Using Occurrence Counters” on page 93 (State Only)

- "Setting Up Loops and Jumps in the Trigger Sequence" on page 81
- "Using Timer Terms" on page 98

Setting Pattern Durations and Occurrence Counts

(Timing Only)

When a bit pattern is found during a *trigger sequence*, you can influence when the term actually becomes "true" by assigning a time duration or an occurrence count.

The (>/</occurs) control may not be directly accessible if you are editing a trigger function. To reveal it, break down (see page 90) the function and then follow these instructions.

To Set a Pattern Duration

1. Select the (>/</occurs) option button and choose an option.
2. Set the duration or the number of occurrences.

When a greater-than or less-than duration is assigned, the secondary branch (Else on) is not available.

When greater-than (>) is used, the analyzer continues sequence level evaluation only after the resource term has been true for greater than or equal to the amount of duration specified.

When less-than (<) is used, the analyzer continues sequence level evaluation only after the resource term has been true for less than or equal to the amount of duration specified. For each less-than assignment, four internal sequence levels (see page 91) are used.

When occurs is selected, you can set an occurrence count. Use the occurrence counter to delay the trigger sequence evaluation until a resource term has occurred in a designated number of samples. These samples may be interrupted by other values -- they do not need to be consecutive.

Using Occurrence Counters

Use the occurrence counter to delay the *trigger sequence* evaluation until a resource term has occurred in a designated number of samples.

The samples do not need to be consecutive. Whatever positive number you assign to the counter, the pattern must be seen that number of times before the term becomes true.

If the Else branch becomes true before all specified occurrences of the primary (Trigger on, or Find) branch, the Else branch is taken.

Branches Taken Stored / Not Stored (State only). *Branches Taken* sets the analyzer to store, or not to store, the resource *terms* of the branch that the analyzer followed in the *trigger sequence*. The Branches Taken control is located in *Trigger*, under the *Settings* tab.

Use *Branches Taken* if you want to maximize memory usage but have a complex trigger. With Branches Taken set to stored, you can use store qualifiers to not store the preliminary data ("While storing no state") but still reconstruct the events leading up to your trigger.

When the analyzer is set to *Branches Taken Stored*, all branch events (*Find*, *Else On*, or *Trigger*) are stored when they occur.

When the analyzer is set to *Branches Taken Not Stored*, the branch events will only be stored if the states they represent are included in the *While Storing* qualifier.

The *While storing* field specifies what is being stored in the analyzer's memory before the trigger conditions are met. To set this, select the *While storing* button and choose a *pattern* or *range* term for the *store qualification*.

Defining Resource Terms

Resource terms are variables that you can use in defining a *trigger sequence*. The terms available include patterns, edges, ranges, and timers.

When the Agilent Technologies 16550A is configured as a *state analyzer,* up to fourteen resource terms are available: 10 *pattern terms Pattern1 - Pattern9* and *Patt10*, two *range terms*, and two *timer terms*. When configured as a *timing analyzer*, 16 resource terms are available from the 14 terms already mentioned, plus two

edge terms.

When the *module* is set up as two analyzers, each resource term can be used by either of the two analyzers, but not both at the same time.

To Define a Resource Term

1. In the *Trigger* tab, select the appropriate subtab to bring the group of terms forward.
2. Optional - Highlight the term name and enter a new name.
   For *Timer* terms, set the time value and skip the remaining steps.
3. Optional - Select the label button to the right of the term name and choose *Replace*.
4. Select the *label* that you want to use.
5. Optional - add more labels (see page 100) to the term.
6. Optional - Set the numeric base.
7. Select the term value field, to the right of the label name.
8. For pattern and range terms, enter the term value. For edge terms, assign edges or *glitches* to appropriate bits.

See Also

"Using Bit Pattern Terms" on page 95

"Using Edge Terms" on page 97

"Using Range Terms" on page 96

"Using Combinations of Terms" on page 99

"Using Timer Terms" on page 98

"Adding and Deleting Labels for Terms" on page 100

"Numeric Base" on page 100

Using Bit Pattern Terms

Bit pattern resource *terms* can be set to match a numeric value or bit pattern on a group of data channels such as a bus. In order for a pattern to be found by the analyzer, the input data must match all bits of the pattern that are not defined as *Don't Cares (X)*. Patterns can also be

used in a negated form.

To Define a Pattern Term

1. Select the *label* name button to the right of the term name and choose the label that you want to use.
2. If necessary, add more labels (see page 100) to the term.
3. Select the term value field, to the right of the label name.
4. Enter the pattern for each label.
   Depending on the base setting, such as hex or octal, some characters will not be accepted. Don't cares are indicated by an *X*.



Right-click on any of the bit pattern value fields to quickly assign the pattern term to a preset value. *Clear (=X)* sets the value to all *X* (don't cares). *Set (=1)* sets the value to all 1s. *Reset (=0)* sets the value to all 0s.

Using Range Terms

Range terms bracket groups of data values between upper and lower boundaries that you assign. The range term becomes true when the data is numerically between or on the two specified boundaries.

The *labels* used by a range must be contained in a single pod pair, with no clock channels or re-ordered bits allowed.

To Define a Range Term

1. Select the label name button to the right of the term name and choose the label that you want to use.
   Only one label can be used at a time for *range terms*.
2. Select the lower value field, to the right of the label name.
3. Enter the pattern for the low value of the range.
4. Select the upper value field, to the right of the label name.

5. Enter the pattern for the upper value of the range.
   Depending on the base setting, such as Hex or Octal, some characters will not be accepted.





Right-click on either of the value fields to quickly assign the value to a preset value. *Set (=1)* sets the value to all 1s. *Reset (=0)* sets the value to all 0s.

Using Edge Terms

Edge *terms* are only available in *timing* mode. You can set an edge term to match transitions or glitches on one or more channels. When you specify an edge or *glitch* on more than one channel, the analyzer logically *ORs* the edges together. When the analyzer sees a transition that matches any of the ones specified in the edge term, the term becomes true. If you must have both edges occur in the same sample, assign the edges to different terms and combine them (see page 99) with an AND.

The following edge choices are available for each bit:

Rising edge (↑)

Falling edge (↓)

Either rising or falling

Glitch (*)

To Define an Edge Term

1. Select the label name button and choose the *label* that you want to use.
2. Select the edge value button, to the right of the label name.

3. Set the edge or glitch for each channel.
   Don't cares are indicated by a (.).

NOTE: After you close the edge term assignment dialog, you may see *$* indicators in the term value field. This symbol indicates that the value can't be displayed in the selected base. Choose *Binary* base to see the actual assignments.

Using Timer Terms

Timers are like other resource *terms* in that they are either true or false. They are unlike other terms, however, in that they are controlled within the *trigger sequence* and act like a stopwatch. When a timer reaches its assigned count (expires), it becomes true. When a timer expires or stops, its count resets to zero.

Timers can be set to Start, Stop, Pause, or Continue as the analyzer enters a trigger sequence level. The two timers are global, so each sequence level except the first has the ability to control the same timer. As more sequence levels are added, the timer status in the new levels defaults to Off. If a timer is paused in one level, it must be continued in another level before it will be restarted. If the trigger sequence returns to the same sequence level again, the timer will be restarted.

Each Timer term can be used by either of the two analyzers, but not both.

To Assign a Time Value to the Timer Terms

1. Select the *Timer* subtab in the *Trigger* tab.
2. Select the timer value field, next to the timer name.
3. Enter a timer value.
4. Use the up/down arrow buttons to scroll through timer values.

The minimum value a timer can have is 400 ns, which is also the default value.

To Include a Timer in a Trigger Sequence

1. Assign a time value to the timer you want to use.
2. Select the sequence level you want to use it in, and choose *Edit...*.

Timer control is not available in the first sequence level.

3. Select the *Off* option button at the bottom of the edit dialog box and choose *Start*.
   If the other analyzer is already using a timer, it won't appear in this analyzer's edit dialog.
4. Select the event field where you want to use the timer.
5. Choose either *Timer* or *Combo...*.
   Use *Combo* for cases such as an edge occurring after so many nanoseconds.
6. In the other trigger levels, you can *Start*, *Pause*, and *Continue* the timer. When a timer reaches its assigned value, it automatically stops.

See Also "Using Combinations of Terms" on page 99

Using Combinations of Terms

Resource *terms* can be used in combinations. Combinations use the logical AND, NAND, OR, NOR, and XOR functions to combine predefined resource terms.

A combined term is evaluated as a single events. Components that are ANDed together must all occur in the same sample. Ones that are ORed together only require one of the components to happen.

If you intend for the logic analyzer to decide between two actions, use branches instead of combined terms. If you want the logic analyzer to find the terms in sequence, use different levels or a trigger function in the *trigger sequence*.

To Set Up a Combination

1. Edit the sequence step (see page 80) that you want a combination term to appear in.
2. Select the resource term assignment field that you want to make into a combination.
3. Choose *Combo...* from the menu of resource terms.
4. In the Combination dialog, select the *On/Off/Not* option button for each term that you want to use and select *On* or *Not*.

Not selects the logical negation of the term.

5. Select the logical operator buttons and choose a logic operation.
   Not all operations are available at all levels of the combination tree.

NOTE: The combination of terms is now inserted into the term assignment field. If the term is too long to fit, the display is truncated.

Adding and Deleting Labels for Terms

Labels are defined in Format, after which they are available throughout the other analyzer areas and attached display tools.

When you use more than one label to define a trigger *term*, the term conditions must be true on both labels for the term to be true.

To Add a Label to a Resource Term

1. Select the label name button.
2. Choose *Insert*.
3. Select the label that you want to add to the resource term.

To Delete a Label from a Resource Term

1. Select the label name button.
2. Choose *Delete*.

Numeric Base

All *labels* have a numeric base field next to them. The base choices are Binary, Octal, Decimal, Hex, ASCII, Symbol and Twos Complement.

To Change the Numeric Base

1. Select the the base option button.
2. Choose the base that you want.

NOTE: If the numeric base is changed in one window, the base in other windows may not change accordingly. For example, the base assigned to symbols is unique, as is the base assigned in the Listing window.

Tagging Data with Time or State Tags (State Only)

The Count field under *Settings* in *Trigger* accesses a selection menu which is used to stamp the data at each memory location with either a Time tag or a State Count tag. The tags reduce the memory depth by half. To retain the full memory depth when using time or state tags leave one *pod pair* unassigned.

State Count

When the State Count option is selected, numbered tags are placed on all selected data. Pre-trigger data has negative numbers and post-trigger data has positive numbers. You select the data to be tagged when you turn on State Count. A field appears to the right of *States* that lets you define patterns.

State tag numbering can be set either relative to the previous tagged sample or absolute from the *trigger point*. Selecting Absolute or Relative is done by toggling the Absolute/Relative field in the Listing Display window.

Time Count

Time Count places time tags on all data. Pre-trigger data has negative time numbers and post-trigger data has positive time numbers. Time tag numbering is set to be either relative to the previous memory location or absolute from the trigger point. The time tag resolution is 4 ns.

Arming Control

An instrument must be *armed* before it can look for its *trigger*. When you *Run* an instrument, it is armed immediately. When using logic analyzer modules that provide two separate analyzers, you can set one analyzer to arm the other within the same module by selecting *Arming Control...* under *Settings* in *Trigger*.

Selecting the *Arming Control...* button brings up the *Machine Arming Tree* dialog. In this dialog you can set what starts each analyzer, and which one sends a signal to *Arm Out*. *Arm Out* is used to send signals to other instruments in the *frame*, or sent to *Port Out*. *Port Out* can be used to control additional instruments external to the logic analysis system.

To change the source of *Arm In* or the destination of *Port Out*, use the Intermodule dialog (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume).

Setting One Analyzer to Arm the Other

This procedure assumes you have both analyzers turned on. The second analyzer can be turned on by dragging it onto the workspace in the Workspace window.

1. Under the Trigger tab, select the *Settings* subtab.
2. Select the *Arming Control...* button.
3. Select the box of the machine that you want to have wait for its arm signal.
4. In the *armed by* menu, select the other machine.
   The arming signal path forms a tree in the Machine Arming Tree dialog.
5. Set the sequence level number to the trigger sequence level that will wait for the arm signal. It does not have to be the same as the trigger level.
6. Select the *Close* button.

NOTE: If the trigger sequence does not pass through the level containing the *wait for arm* term, the trigger will not wait for the arming signal.

See Also

Overview - Multiple Instrument Configuration (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Overview - Multiple Machine Configuration (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume)

Specifications and Characteristics

NOTE:

Definition of Terms

To understand the difference between specifications (see page 103) and characteristics (see page 103), and what gets a calibration procedure (see page 104) and what gets a function test (see page 104), refer to appropriate links within this note.

"Agilent Technologies 16550A Logic Analyzer Specifications" on page 104

"Agilent Technologies 16550A Logic Analyzer Characteristics" on page 105

What is a Specification

A *Specification* is a numeric value, or range of values, that bounds the performance of a product parameter. The product warranty covers the performance of parameters described by specifications. Products shipped from the factory meet all specifications. Additionally, the products sent to Agilent Technologies Customer Service Centers for calibration and returned to the customer meet all specifications.

Specifications are verified by *Calibration Procedures*.

What is a Characteristic

Characteristics describe product performance that is useful in the application of the product, but that is not covered by the product warranty. Characteristics describe performance that is typical of the majority of a given product, but not subject to the same rigor associated with specifications.

Characteristics are verified by *Function Tests*.

What is a Calibration Procedure

Calibration procedures verify that products or systems operate within the specifications. Parameters covered by specifications have a corresponding calibration procedure. Calibration procedures include both performance tests and system verification procedure. Calibration procedures are traceable and must specify adequate calibration standards.

Calibration procedures verify products meet the specifications by comparing measured parameters against a pass-fail limit. The pass-fail limit is the specification less any required guardband.

The term "calibration" refers to the process of measuring parameters and referencing the measurement to a calibration standard rather than the process of adjusting products for optimal performance, which is referred to as an "operational accuracy calibration".

What is a Function Test

Function tests are quick tests designed to verify basic operation of a product. Function tests include operator's checks and operation verification procedures. An operator's check is normally a fast test used to verify basic operation of a product. An operation verification procedure verifies some, but not all, specifications, and often at a lower confidence level than a calibration procedure.

Agilent Technologies 16550A Logic Analyzer Specifications

The specifications are the performance standards against which the product is tested. These specifications apply only to the Agilent Technologies 16550A 100 MHz State/500 MHz Timing logic analyzer:

```
Maximum State Speed:                100 MHz
Minimum State Clock Pulse Width:*   3.5 ns
```

```
Minimum Glitch Width:                  3.5 ns
Threshold Accuracy:                    +/- (100mV + 3% of threshold setting)
Minimum Master-to-Master Clock Time:* 10.0 ns
Setup/Hold Time:*
  Single Clock, Single Edge:           0.0/3.5 ns through 3.5/0.0 ns,
                                       adjustable in 500-ps increments
  Single Clock, Multiple Edge:         0.0/4.0 ns through 4.0/0.0 ns,
                                       adjustable in 500-ps increments
  Multiple Clock, Multiple Edge:       0.0/4.5 ns through 4.5/0.0 ns,
                                       adjustable in 500-ps increments

* Specified for an input signal VH=-0.9 V, VL=-1.7 V, threshold=-1.3 V,
  slew rate=1 V/ns
```

Agilent Technologies 16550A Logic Analyzer Characteristics

The characteristics are not specifications, but are included as additional information.

```
General information
  - Channel Counts:
    1-card module        96 data, 6 clock
    2-card module        198 data, 6 clock
  - Memory Depth:
    Half Channel           4 Ksamples
    Full Channel           8 Ksamples

Probes
  - Input Resistance:           100 Kohm, +/- 2%
  - Parasitic tip capacitance:  1.5 pF
  - Minimum Voltage Swing:      500 mV peak-to-peak
  - Maximum Voltage:            +/- 40 V peak, CAT I
  - Threshold Range:            +/- 6.0 V, adjustable in 50-mV increments

State Analysis
  - State Clocks:                          6
  - State Clock Qualifiers:                4
                                             Each qualifier can be set to
                                             recognize one of six clock
                                             lines, either high or low.
  - Time Tag Resolution:                   8 ns
  - Maximum Time Count Between States:     34 seconds
  - Maximum State Tag Count:               4.29e9


Timing Analysis
  - Maximum Conventional Timing Rate:
    Half Channel               500 MHz
    Full Channel               250 MHz
  - Maximum Transitional Timing Rate:
    Half Channel               250 MHz
    Full Channel               125 MHz
  - Maximum Timing with Glitch Rate:
    Half Channel               125 MHz
    Full Channel               N/A
  - Sample Period Accuracy:    0.01% of sample period
  - Channel-to-Channel Skew:   2 ns, typical
  - Time Interval Accuracy:    +/-( sample period + channel-to-
                               channel skew + 0.01% of time
                               interval reading )
```

```
Triggering
  - Sequencer Speed:                 125 MHz, maximum
  - State Sequence Levels:           12
  - Timing Sequence Levels:          10
  - Maximum Occurrence Count Value:  1,048,575
  - Pattern Recognizers:             10
  - Range Recognizers:               2
  - Range Width:                     32 bits each
  - Timers:                          2
  - Timer Value Range:               400 ns to 500 seconds
  - Glitch/Edge Recognizers:         2 (timing only)
  - Minimum Detectable Glitch:       3.5 ns

Power Requirements
  All necessary power is supplied through the backplane connector
  of the logic analysis system mainframe.

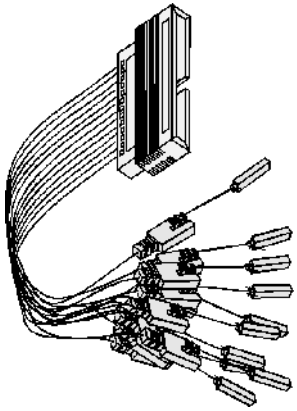
Operating Environment Characteristics
  - Indoor use only.
  - Temperature
    Instrument (except disk and media): 0 to 55 degrees C (+32
                                        to 131 degrees F)
    Probe lead sets and cables:         0 to 65 degrees C (+32 to
                                        149 degrees F)
  - Humidity
    Instrument, probe lead sets, and cables: up to 95% relative humidity at
                                             40 degrees C (+104 degrees F)
  - Altitude      To 4600 m (15,000 ft)
  - Vibration
    Operating:    Random vibration 5 to 500 Hz, 10 minutes per axis,
                  approximately 0.3 g (rms)
    Nonoperating: Random vibration 5 to 500 Hz, 10 minutes per axis,
                  approximately 2.41 g (rms); and swept sine resonant
                  search, 5 to 500 Hz, 0.75 g (0-peak), 5-minute
                  resonant dwell at 4 resonances per axis.
```

Analyzer Probing Overview

The figures below shows a variety of simple probing connections. The specific probe type, number of probes, and location on the target circuit depends on your particular measurement.

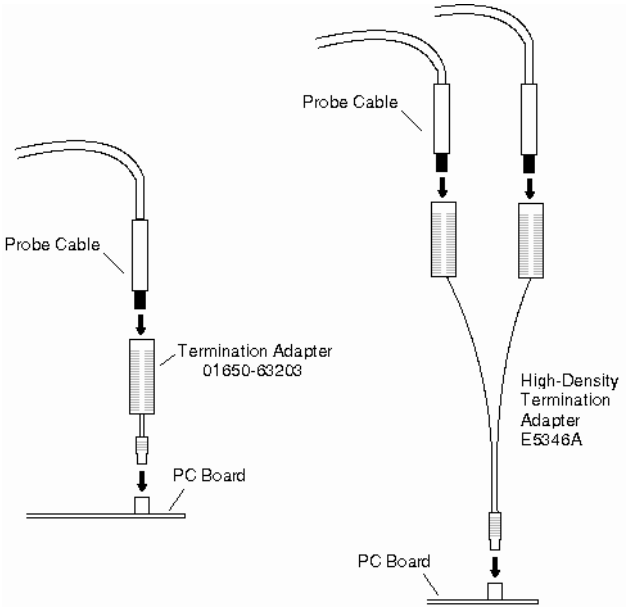
For equivalent circuit diagrams and pinouts, see the description of the probe type in the *Logic Analysis System and Measurement Modules Installation Guide*. If you have misplaced the *Logic Analysis System and Measurement Modules Installation Guide*, you can download the latest version from the Web at <URL: http://www.agilent.com/find/LogicAnalyzer-Manuals/ >

Probe Lead-to-Board Connection



The standard lead set plugs directly into any .1-inch grid with 0.026 to 0.033-inch diameter round pins or 0.025-inch square pins. All probe tips work with the Agilent Technologies 5059-4356 surface mount grabbers and the Agilent Technologies 5959-0288 through-hole grabbers.

Adapter-to-Board Connection



Both the 01650-63203 and the E5346A adapters include termination for the logic analyzer. The 01650-63203 termination adapter plugs into a 2 x 10 pin header with 0.1 inch spacing. The E5346A high-density adapter connects to an AMP "Mictor 38" connector. If possible, use support shrouds around the Mictor connector to relieve strain and improve connections.

Direct Pod-to-Board Connection

If you provide proper termination as part of the target board, you can plug the pod directly into the ©3M 2520-series, or similar alternative connector. Suggested termination is shown in the *Logic Analysis System and Measurement Modules Installation Guide*.

Also use this termination with the Agilent Technologies E5351A high-density, non-terminated adapter.

Pod-to-Analysis Probe Connection

Analysis probes (formerly called preprocessors) are microprocessor-specific interfaces that make it easier to probe buses. Generally,

analysis probes consist of a circuit board that attaches to the microprocessor (possibly through an adapter) and a configuration file. The configuration file sets up the logic analyzer's clocks and labels correctly, and may include an inverse assembler. The circuit board provides access to logical groups of pins through headers designed to connect directly to the logic analyzer.

The easiest way to set up a measurement with an analysis probe is the Setup Assistant. (see the *Setup Assistant* help volume) The Setup Assistant asks you questions about your measurement and then shows you just the information you need to set up the probe correctly. It also loads the proper configuration files.

Using Symbols

You can use symbol names in place of data values when:

- Setting up triggers
- Displaying captured data
- Searching for patterns in Listing displays
- Setting up pattern filters
- Setting up ranges in the System Performance Analyzer

Symbol names can be: variable names, procedure names, function names, source file line numbers, etc.

You can load symbol name definitions into the logic analyzer from a program's object file or from a general-purpose ASCII format symbol file, or you can define symbol names in the logic analyzer.

- "To load object file symbols" on page 111
- "To adjust symbol values for relocated code" on page 112
- "To create user-defined symbols" on page 113
- "To enter symbolic label values" on page 114
- "To create an ASCII symbol file" on page 115
- "To create a readers.ini file" on page 116

See Also

To go to a pattern in the Listing (see the *Listing Display Tool* help volume)

To modify the Source Viewer trace setup (see the *Listing Display Tool* help volume)

To define System Performance Analyzer state interval ranges (see the *System Performance Analyzer* help volume)

To load object file symbols

Object files are created by your compiler/linker or other software development tools.

1. Generate an object file with symbolic information using your software development tools.
2. If your language tools cannot generate object file formats that are supported by the logic analyzer, create an ASCII symbol file (see page 115).
3. Select the *Symbol* tab and then the *Object File* tab.
4. Select the label name you want to load object file symbols for.

   In most cases you will select the label representing the address bus of the processor you are analyzing.
5. Specify the directory to contain the symbol database file (*.ns*) in the field under, *Create Symbol File (.ns) in This Directory*. Select *Browse...* if you wish to find an existing directory name.
6. In the *Load This Object/Symbol File For Label* field, enter the object file name containing the symbols. Select *Browse...* to find the object file and select *Load* in the Browser dialog.

   If your logic analyzer is NFS mounted to a network, you can select object files from other servers.
7. If your program relocates code, see “To adjust symbol values for relocated code” on page 112.

The name of the current object file is saved when a configuration file is saved. The object file will be reloaded when the configuration is loaded.

To reload object file symbols

1. Select the object file/symbol file to reload from the *Object Files with Symbols Loaded For Label* field.
2. Select the *Reload* button.

The values of the object file symbols being used in the trigger sequence or in SPA state-interval ranges will be updated automatically each time

the object file symbols are reloaded.

To delete object file symbol files

1. Select the *Symbol* tab, and then the *Object File* tab.
2. Select the file name you want to delete in the text box labeled, *Object Files with Symbols Loaded For Label*.
3. Select *Unload*.

See Also

"Symbol File Formats" on page 123

To adjust symbol values for relocated code

Use this option to add offset values to the symbols in an object file. You will need this if some of the sections or segments of your code are relocated in memory at run-time. This can occur if your system dynamically loads parts of your code so that the memory addresses that the code is loaded into are not fixed.

To adjust symbol values for a single section of code

1. Select the *Symbol* tab and then the *Object File* tab.
2. In the *Object Files with Symbols Loaded For Label* list, select the file whose symbols you wish to relocate.
3. Select the *Relocate Sections...* button.
4. In the *Section Relocation* dialog, select the field you wish to edit in the section list.
5. Enter the new value for that field and press Enter on your keyboard.
6. Repeat steps 4 through 6 above for any other sections to be relocated.
7. Select *Close*.

To adjust all symbol values

1. Select the *Symbol* tab and then the *Object File* tab.
2. In the *Object Files with Symbols Loaded For Label* list, select the file

whose symbols you wish to relocate.

3. Select the *Relocate Sections...* button.
4. Enter the desired offset in the *Offset all sections by* field. The offset is applied from the linked address or segment.
5. Select *Apply Offset*.
6. Select *Close*.

To create user-defined symbols

1. Under the *Symbol* tab, select the *User Defined* tab.
2. Select the label name you want to define symbols for.
3. At the bottom of the *User Defined* tab, enter a symbol name in the entry field.
4. Select a numeric base.
5. Select *Pattern* or *Range* type for the symbol.
6. Enter values for the pattern or range the symbol will represent.
7. Select *Add*.
8. Repeat steps 3 through 7 for additional symbols.
9. You can edit your list of symbols by replacing or deleting them, if desired.

To replace user-defined symbols

1. Under the *Symbol* tab, select the *User Defined* tab.
2. Select the label you want to replace symbols for.
3. Select the symbol to replace.
4. At the bottom of the *User Defined* tab, modify the symbol name, numeric base, Pattern/Range type, and value, as desired.
5. Select the *Replace* button.
6. Repeat steps 3 through 5 to replace other symbols, if desired.

To delete user-defined symbols

1. Under the *Symbol* tab, select the *User Defined* tab.
2. Select the label you want to delete symbols from.
3. Select the symbol to delete.
4. Select the *Delete* button.
5. Repeat steps 3 and 4 to delete other symbols, if desired.

To load user-defined symbols

If you have already saved a configuration file, and the configuration included user-defined symbols, load the file with its symbols, as follows:

1. In the menu bar of your analyzer window, select *File* and then *Load Configuration...*.
2. In the Load Configuration dialog, select the directory and filename to be loaded.
3. Select the target of the load operation.
4. Select *Load*.

   User-defined symbols that were resident in the logic analyzer when the configuration was saved are now loaded and ready to use.

To enter symbolic label values

When entering label values in the Pattern or Range subtabs of the Trigger tab:

1. Choose the *Symbols* or *Line #s* number base.
2. Select the *Absolute XXXX* button.
3. In the *Symbol Selector* dialog, select the symbol you want to use. All of your symbols for the current label, regardless of type, will be available in the dialog.
   - Use the Search Pattern (see page 122) field to filter the list of symbols

by name. You can use the Recall button to recall a desired Search Pattern.

- Use the Find Symbols of Type selections to filter the symbols by type.

4. Select the symbol you want to use from the list of *Matching Symbols*.
5. If you are using object file symbols, you may need to:
   - Set *Offset By* (see page 122) to compensate for microprocessor prefetches.
   - Set Align to x Byte (see page 123) to trigger on odd-byte boundaries.
6. Select the Beginning, End, or Range of the symbol.
7. Select the *OK* button.

   The name of your symbol now appears as the value of the label.
8. Select the *Cancel* button to exit the *Symbol Selector* dialog without selecting a symbol.

NOTE: When you use symbolic label values in trigger sequences, information about the symbols is saved with the logic analyzer configuration (and in the trigger Save/Recall buffer). If you re-compile your program and reload the logic analyzer configuration (or reload symbols and recall a trigger), the symbol values are automatically re-calculated.

Triggers set up by the Source Viewer now use symbolic values and are re-calculated in the same way.

See Also "Symbols Selector Dialog" on page 121

To create an ASCII symbol file

General-purpose ASCII symbol files are created with text editing/processing tools.

See Also "General-Purpose ASCII (GPA) Symbol File Format" on page 124

To create a readers.ini file

You can change how an ELF/Stabs, Ticoff or Coff/Stabs symbol file is processed by creating a reader.ini file.

1. Create the reader.ini file on your workstation or PC.
2. Copy the file to /logic/symbols/readers.ini on the logic analysis system.

Reader options

C++Demangle

```
1= Turn on C++ Demangling (Default)
0= Turn off C++ Demangling
```

C++DemOptions

```
803= Standard Demangling
203= GNU Demangling      (Default Elf/Stabs)
403= Lucid Demangling
800= Standard Demangling without function parameters
200= GNU Demangling without function parameters
400= Lucid Demangling without function parameters
```

MaxSymbolWidth

```
80= Column width max of a function or variable symbol
    Wider symbols names will be truncated.
    (Default 80 columns)
```

OutSectionSymbolValid

```
0= Symbols whose addresses aren't within the
    defined sections are invalid (Default)
1= Symbols whose addresses aren't within the
    defined sections are valid
```

This option must be specified in the Nsr section of the Readers.ini file:

```
[Nsr]
OutSectionSymbolValid=1
```

ReadElfSection

```
2= Process all globals from ELF section (Default)
   Get size information of local variables
1= Get size information of global and local variables
   Symbols for functions will not be read, and
   only supplemental information for those symbols in
   the Dwarf or stabs section will be read.
0= Do not read the Elf Section
```

If a file only has an ELF section this will have no effect and the ELF

section will be read completely. This can occur if the file was created without a "generate debugger information" flag (usually -g). Using the -g will create a Dwarf or Stabs debug section in addition to the ELF section.

StabsType

```
StabsType=0  Reader will determine stabs type (Default)
StabsType=1  Older style stabs
             (Older style stabs have individual symbol
             tables for each file that was linked into
             the target executable, the indexes of each
             symbol table restart at 0 for each file.)
StabsType=2  Newer style stabs
             (New style stabs have a single symbol table
             where all symbols are merged into a large
             symbol array).
```

ReadOnlyTicoffPage

ReadOnlyTicoffPage tells the ticoff reader to read only the symbols associated with the specified page (as an example 'ReadOnlyTicoffPage=0' reads only page 0 symbols). A value of -1 tells the ticoff readers to read symbols associated with all pages.

```
ReadOnlyTicoffPage=-1 Read all symbols associated will all
                      ticoff pages (Default)
ReadOnlyTicoffPage=p  Read only symbols associated with
                      page 'p' (where p is any integer
                      between 0 and n the last page of
                      the object file).
```

AppendTicoffPage

AppendTicoffPage tells the ticoff reader to append the page number to the symbol value. This assumes that the symbol value is 16-bits wide and that that page number is a low positive number which can be ORed into the upper 16 bits of an address to create a new 32-bit symbol address. For example, if the page is 10 decimal and the symbol address is 0xF100 then the new symbol address will be 0xAF100.

```
AppendTicoffPage=1  Append the ticoff page to the symbol
                    address
AppendTicoffPage=0  Do not append the ticoff page to the
                    symbol address (Default)
```

Examples

Example for Elf/Stabs

```
[ReadersElf]
C
```

```
C
MaxSymbolWidth=60
StabsType=2
```

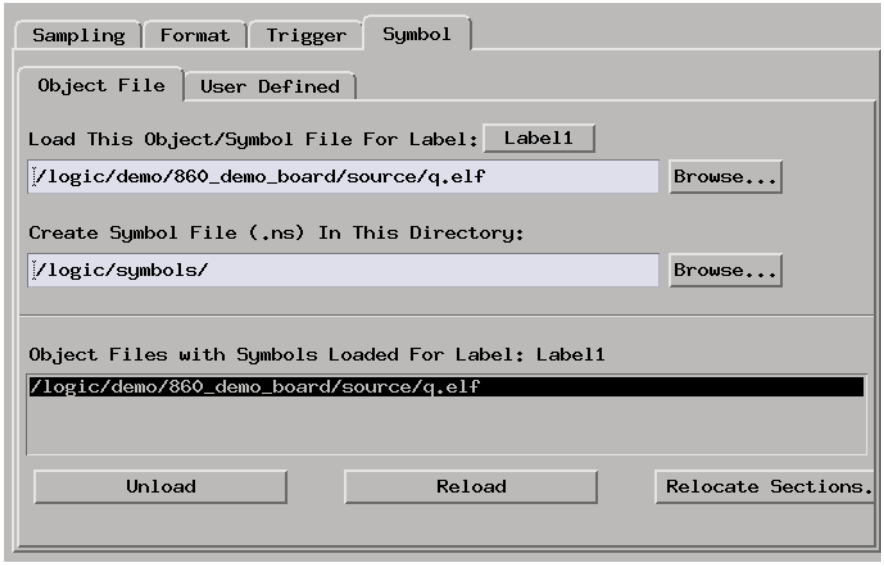
Example for Coff/Stabs (using Ticoff reader)

```
[ReadersTicoff]
C
C
MaxSymbolWidth=60
StabsType=2
```

Example for Ticoff

```
[ReadersTicoff]
C
C
MaxSymbolWidth=60
ReadOnlyTicoffPage=4
AppendTicoffPage=1
```

The Symbols Tab



The Symbols tab lets you load symbol files or define your own symbols. Symbols are names for particular data values on a label.

Two kinds of symbols are available:

- Object File Symbols. These are symbols from your source code and symbols generated by your compiler.
- User-Defined Symbols. These are symbols you create.
- “Symbols Selector Dialog” on page 121
- “Symbol File Formats” on page 123
- “General-Purpose ASCII (GPA) Symbol File Format” on page 124

Multiple files

You can load the same symbol file into several different analyzers, and you can load multiple symbol files into one analyzer. Symbols from all the files you load will appear together in the object file symbol selector that you use to set up resource terms.

Object file versions

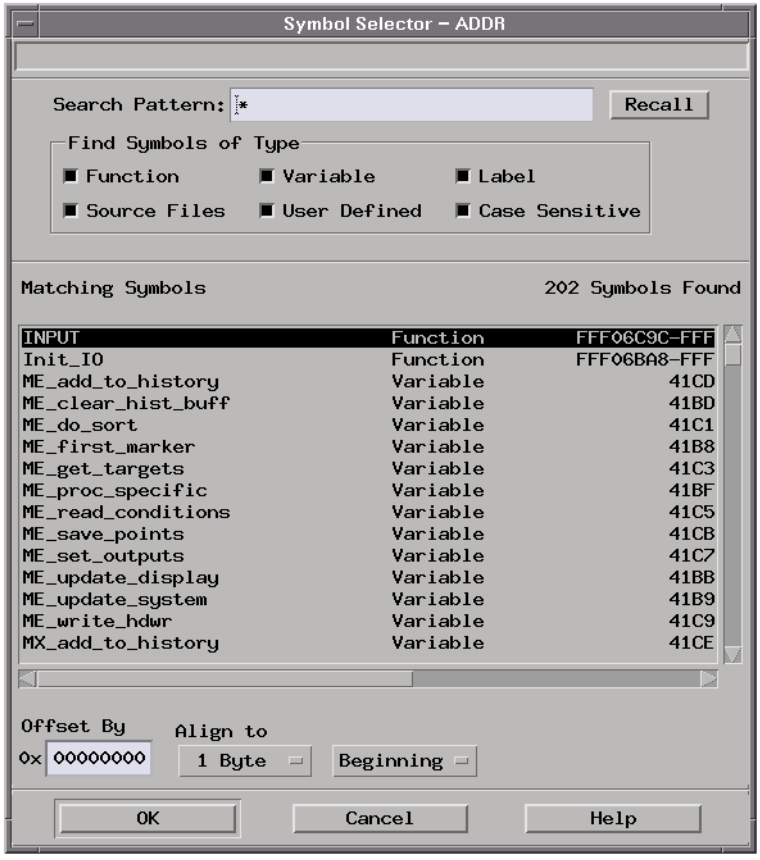
During the load process, a symbol database file with a *.ns* extension will be created by the system. One *.ns* database file will be created for each symbol file you load. Once the *.ns* file is created, the Symbol Utility will use this file as its working symbol database. The next time you need to load symbols into the system, you can load the *.ns* file explicitly, by placing the *.ns* file name in the *Load This Object/Symbol File For Label* field.

If you load an object file that has been loaded previously, the system will compare the time stamps on the *.ns* file and the object file. If the object file is newer, the *.ns* file will be created. If the object file has not been updated since it was last loaded, the existing *.ns* file will be used.

See Also

"Using Symbols" on page 110

Symbols Selector Dialog



Search Pattern: Lets you enter partial symbol names and the asterisk wildcard character (*) to limit the symbols to choose from (see “Search Pattern” on page 122). Use the Recall button to select from previous search patterns.

Find Symbols of Type Lets you limit the types of symbols to choose from.

Matching Symbols Lists the symbols that match the search pattern. You choose a symbol from this list.

Offset By Lets you add an offset value to the starting point of a symbol. This can be useful when compensating for microprocessor prefetches (see "Offset By Option" on page 122).

Align to Lets you mask the lower order bits of a symbol's value. This can be useful for triggering on odd byte boundaries (see "Align to x Byte Option" on page 123).

Beginning/End/ Range When a symbol represents a range of addresses, you can choose the beginning address of the range, the end address of the range, or the whole range.

See Also "To enter symbolic label values" on page 114

Search Pattern

Use this field to locate particular symbols in the symbol databases. To use this field, enter the name of a file or symbol. The system searches the symbol database for symbols that match this name. Symbols that match appear in the list of *Matching Symbols*. You can also use wildcard characters to find symbols.

Asterisk wildcard (*) The asterisk wildcard represents "any characters." When you perform a search on the symbol database using just the asterisk, you will see a list of all symbols contained in the database. The asterisk can also be added to a search word to find all symbols that begin or end with the same letters. For example, to find all of the symbols that begin with the letters "st", select the Search Pattern field and enter "st*".

Offset By Option

The Offset By option allows you to add an offset value to the starting point of the symbol that you want to use. You might do this in order to trigger on a point in a function that is beyond the preamble of the function, or to trigger on a point that is past the prefetch depth of the processor. Setting an offset helps to avoid false triggers in these situations. The offset specified in the Offset By field is applied before the address masking is done by the "Align to x Byte" option.

Example An 80386 processor has a prefetch depth of 16 bytes. Assume functions

func1 and *func2* are adjacent to each other in physical memory, with *func2* following *func1*. In order to trigger on *func2* without getting a false trigger from a prefetch beyond the end of *func1*, you need to add an offset value to your label value. The offset value must be equal to or greater than the prefetch depth of the processor. In this case, you would add an offset of 16 bytes to your label value. You would set the value of the "Offset By" field to 10 hex. Now, when you specify *func2* as your label value, the logic analyzer will trigger on address *func2+10*.

Align to x Byte Option

Most processors do not fetch instructions from memory on byte boundaries. In order to trigger a logic analyzer on a symbol at an odd-numbered address, the address must be masked off. The "Align to x Byte" option allows you to mask off an address.

Example

Assume the symbol "main" occurs at address 100F. The processor being probed is a 68040, which fetches instructions on long-word (4-byte) boundaries. In order to trigger on address 100F, the Align to x Byte option sets the two least-significant address bits to "don't cares". This qualifies any address from 100C through 100F.

Symbol File Formats

The logic analysis system can read symbol files in the following formats:

- OMF96
- OMFx86
- IEEE-695
- ELF/DWARF
- ELF/stabs
- TI COFF

For ELF/DWARF1, ELF/stabs, and ELF/stabs/Mdebug files, C++ symbols are demangled so that they can be displayed in the original

C++ notation. To improve performance for these ELF symbol files, type information is not associated with variables. Hence, some variables (typically a few local static variables) may not have the proper size associated with them. They may show a size of 1 byte and not the correct size of 4 bytes or even more. All other information function ranges, line numbers, global variables and filenames will be accurate. These behaviors may be changed by creating a readers.ini (see page 116) file.

See Also

"To load object file symbols" on page 111

"To create an ASCII symbol file" on page 115

"To create a readers.ini file" on page 116

General-Purpose ASCII (GPA) Symbol File Format

General-purpose ASCII (GPA) format files are loaded into a logic analyzer just like other object files.

If your compiler does not produce object files in a supported format, or if you want to define symbols that are not included in the object file, you can create an ASCII format symbol file.

Typically, ASCII format symbol files are created using text processing tools that convert the symbol table information from a compiler or linker map output file.

Different types of symbols are defined in different records in the GPA file. Record headers are enclosed in square brackets, for example, [VARIABLES]. For a summary of GPA file records and associated symbol definition syntax, refer to the "GPA Record Format Summary" on page 125 that follows.

Each entry in the symbol file must consist of a symbol name followed by an address or address range.

While symbol names can be longer, the logic analyzer only uses the first 16 characters.

The address or address range must be a hexadecimal number. It must appear on the same line as the symbol name, and it must be separated from the symbol name by one or more blank spaces or tabs. Address ranges must be in the following format:

```
beginning address..ending address
```

The following example defines two symbols that correspond to address ranges and one symbol that corresponds to a single address.

```
main      00001000..00001009
test      00001010..0000101F
var1      00001E22      #this is a variable
```

For more detailed descriptions of GPA file records and associated symbol definition syntax, refer to the following topics:

- “SECTIONS” on page 126
- “FUNCTIONS” on page 127
- “VARIABLES” on page 128
- “SOURCE LINES” on page 128
- “START ADDRESS” on page 129
- “Comments” on page 129

GPA Record Format Summary

Format

```
[SECTIONS]
section_name  start..end  attribute

[FUNCTIONS]
func_name  start..end

[VARIABLES]
var_name   start [size]
var_name   start..end

[SOURCE LINES]
File: file_name
line#  address
```

```
[START ADDRESS]
address

#comment text
```

Lines without a preceding header are assumed to be symbol definitions in one of the [VARIABLES] formats.

Example

This is an example GPA file that contains several different kinds of records.

```
[SECTIONS]
prog      00001000..0000101F
data      40002000..40009FFF
common    FFFF0000..FFFF1000

[FUNCTIONS]
main      00001000..00001009
test      00001010..0000101F

[VARIABLES]
total     40002000  4
value     40008000  4

[SOURCE LINES]
File: main.c
10        00001000
11        00001002
14        0000100A
22        0000101E

File: test.c
 5        00001010
 7        00001012
11        0000101A
```

SECTIONS

Use SECTIONS to define symbols for regions of memory, such as sections, segments, or classes.

NOTE: To enable section relocation, section definitions must appear before any other definitions in the file.

NOTE: If you use section definitions in a GPA symbol file, any subsequent function or variable definitions must be within the address ranges of one of the defined sections. Functions and variables that are not within the range are ignored.

Format

```
[SECTIONS]
section_name  start..end  attribute
```

section_name A symbol representing the name of the section.

start The first address of the section, in hexadecimal.

end The last address of the section, in hexadecimal.

attribute (optional) Attribute may be one of the following:

NORMAL (default) - The section is a normal, relocatable section, such as code or data.

NONRELOC - The section contains variables or code that cannot be relocated. In other words, this is an absolute segment.

Example

```
[SECTIONS]
prog            00001000..00001FFF
data            00002000..00003FFF
display_io      00008000..0000801F  NONRELOC
```

FUNCTIONS

Use FUNCTIONS to define symbols for program functions, procedures or subroutines.

Format

```
[FUNCTIONS]
func_name  start..end
```

func_name A symbol representing the function name.

start The first address of the function, in hexadecimal.

end The last address of the function, in hexadecimal.

Example

```
[FUNCTIONS]
main      00001000..00001009
test      00001010..0000101F
```

VARIABLES

You can specify symbols for variables using:

- The address of the variable.
- The address and the size of the variable.
- The range of addresses occupied by the variable.

If you specify only the address of a variable, the size is assumed to be 1 byte.

Format

```
[VARIABLES]
var_name   start [size]
var_name   start..end
```

var_name A symbol representing the variable name.

start The first address of the variable, in hexadecimal.

end The last address of the variable, in hexadecimal.

size (optional) The size of the variable, in bytes, in decimal.

Example

```
[VARIABLES]
subtotal     40002000   4
total        40002004   4
data_array   40003000..4000302F
status_char  40002345
```

SOURCE LINES

Use SOURCE LINES to associate addresses with lines in your source files.

Format

```
[SOURCE LINES]
File: file_name
line#  address
```

file_name The name of a file.

line# The number of a line in the file, in decimal.

address The address of the source line, in hexadecimal.

Example

```
[SOURCE LINES]
File: main.c
10        00001000
11        00001002
14        0000100A
22        0000101E
```

See Also Using the Source Viewer (see the *Listing Display Tool* help volume)

START ADDRESS

Format

```
[START ADDRESS]
address
```

address The address of the program entry point, in hexadecimal.

Example

```
[START ADDRESS]
00001000
```

Comments

Use the # character to include comments in a file. Any text following the # character is ignored. You can put comments on a line alone or on the same line following s symbol entry.

Format

```
#comment text
```

Example

```
#This is a comment
```

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 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 16-state 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 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 0.0.0.0, 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 or higher 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 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 available 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 no-ops 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 post-processing 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.

Index

Symbols

, 93
&> duration trigger field, 93
*, bit assignment, 66
+, label polarity, 68
-, label polarity, 68
., bit unassignment, 66

Numerics

125 MHz acquisition mode, 47
16550A characteristics, 105
16550A logic analyzer, 2
16550A specifications, 104
250 MHz acquisition mode, 48

A

absolute tag, 101
acquisition depth control, 40
acquisition mode, 41
acquisition modes, about transitional timing, 46
acquisition modes, comparison of timing, 51
acquisition modes, overview, 11
acquisition modes, state, 46
acquisition modes, timing, 46
acquisition modes, timing, glitch, 52
acquisitions, interpreting, 26
acquisitions, processing, 26
activity indicator, in Format, 66
advanced clocking checkbox, 41
Align to x Byte option, 123
Align to x Byte option for symbols, 123
altitude characteristics, 105
analyzer 2, turning on, 65
analyzer name field, 44
analyzer name, where used, 44
analyzer probes, general-purpose, 107
analyzer probes, termination adapter, 107
analyzer, arming, 101
analyzer, changing name, 44
analyzer, off button, 45
analyzer, on checkbox, 45
arming analyzer, 101
arming control, 31, 101
arrows, activity indicators, 64
ASCII, 56
ASCII format symbols, 126, 127, 128, 129
ASCII symbol file, 128

B

bad data in measurement, 38
base, numeric, 100
basic state measurements, example, 18
basic timing measurements, example, 22
bit activity indicator, 66
bit assignments, preserving, 68
bit numbering within a label, 66
bit order, changing, 69
bit pattern terms, 95
bit significance, 66
bits, assigning, 66
bits, reordering, 69
branches taken, 94
break down functions, 90
browsing, 122
browsing the symbol database, 122
buses, naming, 70
byte mode, numeric base, 100

C

cable activity indicators, 64
cable power, probe, characteristic, 105
capacitive loading, 33
center trigger position, 53
channel activity indicators, 64
channel counts, characteristic, 105
channel width, 46
channels, assigning to label, 70
channels, maximum per label, 66
channel-to-channel skew, characteristic, 105
clear menu, 83
clear sequence levels, 83
clear sequence/resource/name, 83
clock bits as data channels, 66
clock channel specifiers, 41
clock channels, inputs available as data, 66
clock edges, adjusting, 73
clock lines informational message, 37
clock mode field, 41
clock qualifiers, characteristic, 105
clock setup area, 41
clock threshold level note, 72
clock threshold note, 70
clock time, specification, 104
clocks, data/address same line, 42
clocks, master/slave/demultiplex, 42
clocks, overview, 11
code, assigning address offsets, 112
COFF symbol reader options, 117
combination terms, 99
combo pick, 99
comments, 129
config tab, use, 10
configuration files, loading, 32
configurations, compatibility across models, 32
configurations, storing, 32
connection methods, 10
connections, 10
conventional timing acquisition mode, 51
conventional timing acquisition modes, 46

count state, 101
count states timing function, 85
count time, 101
create labels for busses, 70
custom state function, 87
custom timing macros, 85
custom trigger macros, 82

D

dashes, activity indicators, 64
data channels, using clock bits, 66
data latching note, 42
data on clocks display, 66
data set, interpreting, 26
data, stamping, 101
defaulting the trigger, 83
definition, calibration procedure, 104
definition, characteristic, 103
definition, function test, 104
definition, operational accuracy calibration, 104
definition, specification, 103
delay trigger position, 53
deleting terms, 83
demultiplex clock, 42
demultiplex clocks, 71
demultiplex clocks for pods, 70
depth of memory, characteristic, 105
duration, timing, 50

E

edge and pattern function, 86
edge terms, 97
edge trigger function, 87
edge trigger terms, 97
edit trigger sequence steps, 80
ELF symbol reader options, 116
ELF/DWARF file format, 123
ELF/stabs file format, 123
else branch, 81
end trigger position, 53
enviromental characteristics, 105
error messages, clock lines must be connected to the card, 37
error messages, maximum of 32 channels per label, 34
error messages, measurement initialization error, 34
error messages, slow or missing clock, 34
error messages, timer is off, 35
error messages, timer never started, 35
error messages, two pod pairs are needed to use both timers, 36
error messages, waiting for trigger, 37
errors in data, 38
errors, error messages, 33
example, 122, 123

F

file versions, 111
files, 111
find early event state trigger functions, 89
find edge function, 85
find event state functions, 88
find late event state trigger functions, 89
find n times state functions, 88
find pattern state functions, 88
find sequence state triggers, 88
finding the symbol you want, 122
flowchart, 10
format tab, use, 13, 55
full channel 250 MHz, timing, 46
full channel state mode, 41
full channel timing mode, 41
full channel, timing mode, 47
full channel/4K Memory/100 MHz, 46
functions, 127

G

glitch, 97
glitch mode, 41
glossary of terms, 2
group bit assignment, 66

H

half channel 500 MHz, timing, 46
half channel state mode, 41
half channel timing mode, 41, 71
half channel timing, transitional, 49
half channel timing, transitions, 50
half channel transitional timing mode, 48
half channel/8K Memory/100 MHz, 46
help, symbols, 119
help, trigger, 101
help, trigger sequence, 83
how to set up, 10
humidity characteristics, 105

I

id, naming analyzer, 44
IEEE-695 file format, 123
if branch, 81
import, 56
in ASCII format, 126, 127, 128, 129
in symbol browser, 122
input capacitance, probe, characteristic, 105
input resistance, probe, characteristic, 105
internal sequence, 83
internal sequence levels, 91
invalid data note, 71

L

label polarity, changing, 68

label values, symbolic, 114
labels, 14
labels, activating, 68
labels, add after, 67
labels, add before, 67
labels, adding, 67
labels, assigning bits, 66
labels, defining, 70
labels, deleting, 67
labels, deleting from terms, 100
labels, inserting, 67
labels, inserting on terms, 100
labels, polarity, 68
labels, reordering bits, 69
labels, turning off, 68
labels, turning on, 68
latch, glitch, 94, 97
latch, glitch, characteristic, 105
least significant bit in label, 66
line numbers, 128
load, reducing, 33
loading, 111
loading files including symbols, 111
loading object file symbols, 111
loading user-defined symbols, 114
logic analyzer hangs, 38
logic analyzer probes, 10, 107
logic analyzer triggers, 76
logic analyzer, testing, 39

M

machines available, characteristic, 105
main system help page, 2
masking off addresses of symbols, 123
master clock, 42
master clocks for pods, 70
maximizing memory, 94
maximum pulse width trigger, 87
maximum transitions stored in half-channel mode, 50
maximum transitions stored, full channel, 48
measurement doesntrun', 38
measurement setup, 11
measurement, probing options, 107
measurements, overview of basic state, 18
measurements, overview of timing, 22
memory and trigger, 53
memory depth, characteristic, 105
memory depth, setting, 40
message, trigger inhibited during timing prestore, 36
minimum pulse width trigger, 87
minimum transitions in half-channel mode, 49
minimum transitions stored, full channel, 48
mixed signal, 31
mode and acquisition depth, 41
mode and channel width, 41
mode and sample rate, 41
mode, clocking, 42
most significant bit in label, 66

N

name, id for saving setup, 44
negative logic, note, 68
netlist, 56
note, activity indicators, 64
note, clock, characteristic, 105
note, clocking, 42
note, negative logic, 68
note, pattern recognizer, 105
note, state memory, 46

O

object file symbol files, 111
object file symbols, 122
occurrence counter, characteristic, 105
occurrence counter, state trigger, 93
occurrence counter, timing trigger, 93
occurs field, 93
occurs trigger field, 93
odd-numbered addresses, 123
odd-numbered addresses represented by symbols, 123
offset addresses, assigning, 112
Offset By option of the symbol browser, 122
OMF96 file format, 123
OMFx86 file format, 123
on/off switch, 45
operating environment characteristics, 105
overview, measurement process, 10

P

parasitic tip capacitance, probe, characteristic, 105
pattern after edge function, 86
pattern count, state trigger, 93
pattern duration function, 85
pattern duration, timing, 93
Pattern field, 122
pattern recognizers, characteristic, 105
pattern, state trigger functions, 88
pattern/edge trigger functions, 86
performance verification, 39
period, sample time, 45
pod assignment dialog, 65
pod clocking, demultiplex, 70
pod thresholds, setting, 72
pods, assigning, 65
pods, clocking, 42
pods, selecting, 71
pods, specifying state clock, 70

power through pod cables, characteristic, 105
predefined macros, 84, 91
predefined trigger functions, modifying, 90
prefetch, triggering beyond, 122
probe leads, 107
probes, individual signal, 107
probing options, 10
probing tips, 10
probing, overview, 107
problems making measurements, 33
pulse width function, 87

Q

qualifiers, clocks, characteristic, 105

R

R, bit assignment, 66, 69
radix, numeric base, 100
range recognizers, characteristic, 105
range terms, 96
range trigger terms, 96
range width, characteristic, 105
ranges and reordered bits, 69
rate, sample period, 45
readers.ini file, 116
recalling trigger sequences, 82
relative tag, 101
relocating sections of code, 112
reset bit pattern, 95
restore functions, 90
results, 122
roadmap, 10

S

sample period control, 45
sample period, characteristic, 105
sample rate, setting, 41
sampling tab, use, 40
sampling tab, uses, 11
saving trigger sequences, 82
Search Pattern field, 122
searching the symbol database, 122
sections, 126
selecting macros, 78
self test, 39
sequence levels, characteristic, 105
sequencer, maximum levels, characteristic, 105
set up, trigger sequence, 81
setting threshold, 72
settings, saving, 32
setup/hold field, 73
setup/hold time, specification, 104
shortcut, bit assignment, 66
skew, channel-to-channel, characteristic, 105
slave clock, 42
slave clocks for pods, 70
slow clock message, 34
source line numbers, 128
specifications and characteristics, 103
speed, state/timing, characteristic, 105
Stabs symbol reader options, 117
stamp data, 101
start address, 129
start trigger position, 53
state channel width, 46
state clock pulse width, specification, 104
state clocks, 41
state clocks, characteristic, 105
state clocks, master/slave/both, 70
state clocks, setup/hold, 73
state count, memory, 46
state measurements, basic overview, 18
state memory, 46
state mode, 41
state modes, 46
state modes, characteristic, 105
state speed, characteristic, 105
state tags, 101
state trigger functions, 84, 88, 89
stored/not stored, 94
storing trigger setups, 82
symbol demangling, 116
symbol file formats, 123
symbol file versions, 111
symbol selector dialog, 121
symbolic label values, 114
symbols, 122
symbols, loading object file symbols, 111
symbols, loading user-defined, 114
symbols, outside defined sections, 116
symbols, types and use, 119
symbols, user-defined, 113
symbols, using, 110

T

tab, symbols, 119
tag data, 101
temperature characteristics, 105
terms, 14
then branch, 81
threshold accuracy, specification, 104
threshold logic levels, ECL, 72
threshold logic levels, TTL, 72
threshold range, probe, characteristic, 105
TI COFF file format, 123
time count, 46
time count, characteristic, 105
time interval accuracy, characteristic, 105
time tag resolution, characteristic, 105
time tags, 101

time violation functions, 87
timer error message, 35
timer warning message, 35
timer, characteristics, 105
timers, using, 98
timing acquisition mode, glitch, 52
timing analysis characteristics, 105
timing measurements, basic overview, 22
timing mode, 41
timing modes, 46
timing modes, comparison, 51
timing modes, transitional, 46
timing speed, characteristic, 105
timing trigger functions, 84
timing, memory depth, 46, 51
trace buffer, 71
transitional timing acquisition mode, 51
transitional timing acquisition modes, about, 46
transitional timing half-channel mode, 50
transitional timing mode, 41
transitional timing mode, considerations, 50
transitional timing, full channel mode, 47
transitional timing, half channel, 48
transitional timing, minimum transitions stored, 49
trigger bit patterns, 94
trigger characteristics, 105
trigger edge terms, 94
trigger functions, 83
trigger functions, branching, 81
trigger functions, creating, 91
trigger functions, displaying parts, 90
trigger functions, modifying, 91
trigger functions, pattern/edge, 86
trigger functions, predefined, 84
trigger functions, state, 87, 88, 89
trigger functions, time violation, 87
trigger functions, timing, 85
trigger functions, user-level, 91
trigger functions, using, 78
trigger glitch, 94
trigger inhibited informational message, 36
trigger macros, selecting, 78
trigger pattern functions, 85
trigger pattern terms, 94
trigger position control, 53
trigger range terms, 94
trigger resource terms, 94
trigger resource terms, combination, 99
trigger sequence branches, 81
trigger sequence levels, goto, 81
trigger sequence steps, copying, 79
trigger sequence steps, editing, 80
trigger sequence steps, replacing, 79
trigger sequence, adding steps, 79
trigger sequence, clearing, 83
trigger sequence, customizing, 91
trigger sequence, default, 83
trigger sequence, deleting, 79
trigger sequence, editing, 79
trigger sequence, explanation, 83
trigger sequence, inserting, 79
trigger sequence, specifying, 78
trigger sequences, loading, 82
trigger sequences, saving, 82
trigger sequences, storing, 82
trigger set up, 81
trigger tab, reference, 75
trigger tab, use, 14
trigger terms, adding labels, 100
trigger terms, bit pattern, 95
trigger terms, combination, 99
trigger terms, deleting labels, 100
trigger terms, edge, 97
trigger terms, range, 96
trigger terms, timers, 98
trigger timer terms, 94
trigger variables, 94
trigger, continue timer, 98
trigger, pattern durations, 93
trigger, pause timer, 98
trigger, poststore, 53
trigger, resetting, 83
trigger, setting up, 78
trigger, start timer, 98
trigger, stop timer, 98
trigger, substeps, 14
triggering on a symbol, 122, 123
triggering on a symbol beyond prefetch depth, 122
triggering, about, 76
troubleshooting the logic analyzer, 33

U

unassigned bits, 66
user macros, trigger, 82
user threshold logic level, 72
user-defined macros, 91
user-defined symbols, 113
user-defined symbols, loading, 114
user-defined trigger, 91
user-level function, 91
user-level trigger functions, 81

V

variables, 128
versions, 111
versions of symbol files, 111
vibration characteristics, 105

W

wait state functions, 89
wait time function, 87
warranty, what is covered, 103
what is triggering, 76
while storing, 94
wildcard characters, 122

Index

Publication Number: 5988-9023EN
January 1, 2003



Agilent Technologies

Artisan Technology Group is an independent supplier of quality pre-owned equipment

Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP