

---

# Help Volume

© 1992-99 Hewlett Packard Company. All rights reserved.

---

**HP 16715A 167 MHz State/667  
MHz Timing Logic Analyzer**

---

# HP 16715A 167 MHz State/667 MHz Timing Logic Analyzer



The HP 16715A 167 MHz State/667 MHz Timing logic analyzer offers fast sample rates and deep memory with up to 340 channels. This logic analyzer also marks the debut of VisiTrigger, HP's new patented logic analyzer triggering system.

## Getting Started

- “Analyzer Probing Overview” on page 97
- “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

Measurement Tips and Tricks (see the *Measurement Examples* help volume)

“Arming Control - Multiple Instruments and Analyzers” on page 30

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

Using Symbols (see page 100)

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

“Loading and Saving Logic Analyzer Configurations” on page 32

“Understanding Logic Analyzer Triggering” on page 65

“Testing the Logic Analyzer Hardware” on page 42

### **Interface Reference**

“The Sampling Tab” on page 43

“The Format Tab” on page 51

“The Trigger Tab” on page 62

“The Symbols Tab” on page 100

“Specifications and Characteristics” on page 93

Main System Help (see the *HP 16600A/16700A Logic Analysis System* help volume)

Glossary of Terms (see page 127)



## **HP 16715A 167 MHz State/667 MHz Timing Logic Analyzer**

### **1 HP 16715A 167 MHz State/667 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
Arming Control - Multiple Instruments and Analyzers	30
Loading and Saving Logic Analyzer Configurations	32
When Something Goes Wrong	33
Interference with Target System	33
Error Messages	33
Nothing Happens	34
Suspicious Data	34
Testing the Logic Analyzer Hardware	42

---

# Contents

The Sampling Tab	43
Acquisition Depth	43
Setting the Acquisition Mode	44
Performing Clock Setup (State only)	44
Naming the Analyzer	47
Turning the Analyzer Off	48
Sample Period (Timing Only)	48
Trigger Position Control	49
The Format Tab	51
Activity Indicators	51
Assigning Pods to the Analyzers	52
Data On Clocks Display	53
Labels: Mapping Analyzer Channels to Your Target	57
Setting Up the Pod Clock	57
Pod Selection	58
Setting the Pod Threshold	59
State Clock Setup/Hold (State only)	59
The Trigger Tab	62
E-mail Notify on Trigger	64
What is SMTP	65
Understanding Logic Analyzer Triggering	65
Setting Up a Trigger	68
Inserting and Deleting Sequence Steps	69
Editing Sequence Levels	71
Setting Up Loops and Jumps in the Trigger Sequence	71
Saving and Recalling Trigger Sequences	72
Trigger Function Libraries	73
Clearing Part or All of the Trigger	74
Overview of the Trigger Sequence	75
Trigger Functions	76
Working with Advanced Functions	85
Defining Events	88
Tagging Data with Time or State Tags (State Only)	91

---

# Contents

Specifications and Characteristics	93
What is a Specification	93
What is a Characteristic	93
What is a Calibration Procedure	94
What is a Function Test	94
HP 16715A Logic Analyzer Specifications	94
HP 16715A Logic Analyzer Characteristics	95
Analyzer Probing Overview	97
The Symbols Tab	100
Displaying Data in Symbolic Form	101
Setting Up Object File Symbols	102
To Load Object File Symbols	102
Relocating Sections of Code	104
To Delete Object File Symbol Files	105
Symbol File Formats	105
Creating ASCII Symbol Files	106
Creating a readers.ini File	111
User-Defined Symbols	113
To Create User-Defined Symbols	113
To Replace User-Defined Symbols	113
To Delete User-Defined Symbols	114
To Load User-Defined Symbols	114
Using Symbols In The Logic Analyzer	115
Using Symbols As Trigger Terms	115
Using Symbols as Search Patterns in Listing Displays	116
Using Symbols as Trigger Terms in the Source Viewer	116
Using Symbols as Pattern Filter Terms	117
Using Symbols as Ranges in the Software Performance Analyzer	117
Help - How to Navigate Quickly	120

---

# Contents

Help - System Overview	121
Run/Group Run Function	122
Setting a tool for independent or Group Run	123
Setting Single or Repetitive Run	124
Checking Run Status	124
Demand Driven Data	125

## **Glossary**

## **Index**

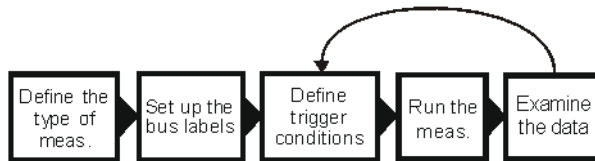


---

HP 16715A 167 MHz State/667 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**

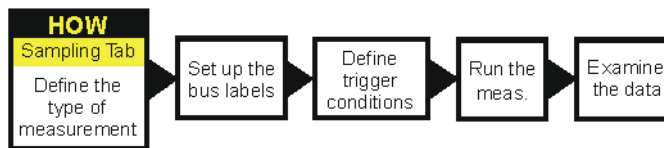
“Analyzer Probing Overview” on page 97 for more detail on types of probes

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



Choose state or timing  
Set measurement mode  
Set up state clock

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.

## Setting Up a Measurement

### 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 Acquisition Depth control.

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

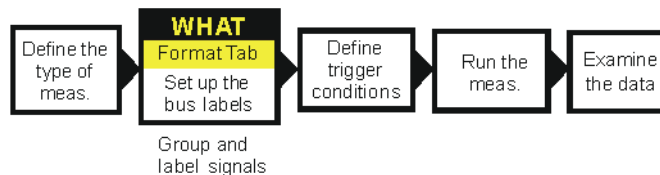
**See Also**

“The Sampling Tab” on page 43 for information on setting type and assigning pods

“Setting the Acquisition Mode” on page 44 for links to this analyzer's modes


“Performing Clock Setup (State only)” on page 44

## Set Up the Bus Labels



The next step is to finish defining the physical connection between the target system and the analyzer. 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 TTL or ECL logic levels,

**Setting Up a Measurement**

or set a different 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)

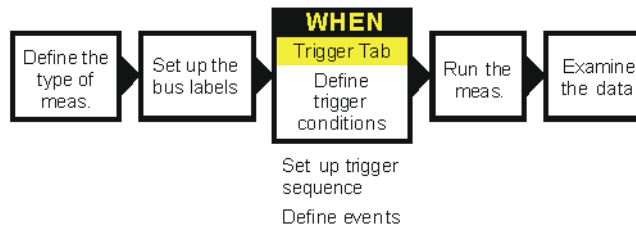
**See Also**

“Assigning Bits to a Label” on page 53

“The Format Tab” on page 51

---

## 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 defined event occurs. When that happens, it goes to the next level and follows the instructions there.

**Define Events**

Trigger events can be used like variables in the trigger sequence. Some

trigger functions only let you insert events that pertain to the description of the function. The advanced functions let you insert any type of event.

Step 4: Run the Measurement (see page 15)

### See Also

“Defining Events” on page 88

“Understanding Logic Analyzer Triggering” on page 65

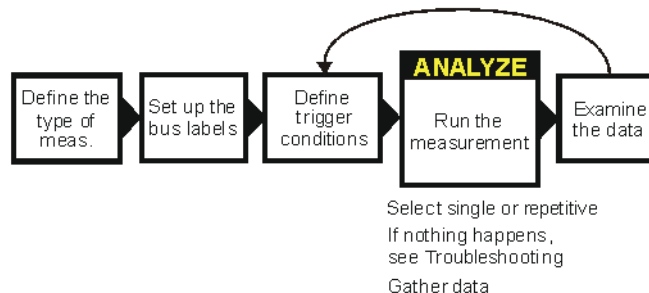
“Setting Up a Trigger” on page 68

“The Trigger Tab” on page 62

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.

**Setting Up a Measurement****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, click *Stop*.

**NOTE:**

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

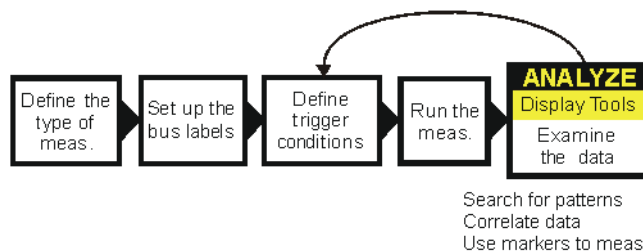
**If Nothing Happens...**

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 HP 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 HP 16600A-series or HP 16700A logic analysis system, open a logic analyzer setup window.
  - a. In the main window, *right-click* the logic analyzer icon.
  - b. Select *Setup...* from the menu.
  - c. Click 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. Click on *Off* under J and choose "Falling Edge" from the menu.
4. Group and label bits.
  - a. Click on the *Format* tab.
  - b. Optional - Insert a second label.
    1. Right-click *Label1*.

2. Select *Insert after...*
3. In the *Enter Label Name* box, click *OK*.
- c. Optional - Rename *Label1*.
  1. Right-click *Label1*.
  2. Select *Rename...*
  3. Enter a new name in the name field.
  4. Click *OK* to close the *Rename Label* box.
- d. Right-click the bit assignment field.

The bit assignment field is the field to the right of a label name, and under a pod column.
- e. Select *. . . . .\*\*\*\*\** from the menu.

If none of the choices match your own system, select *Individual...* and click on the individual bits to assign them (\*) or ignore them (.)
5. Define trigger events for patterns on buses.
  - a. Click the *Trigger* tab.
  - b. Optional - Rename *Pattern1*.
    1. Double-click in the *Pattern1* field.
    2. Enter a new name.
  - c. Select the appropriate label.
    1. Click the field to the far-right of the label name.
    2. To define the event as a combination of labels, click *Insert...* To use a different label to define the event, click *Replace...*
    3. In the dialog box, click the label name you want to use and then click *OK*.
  - d. Click in 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

## Making a Basic State Measurement

and combining trigger events.

- a. Click the *1* box and select *Edit...*
- b. In the dialog box, click the *Pattern1* button just after Trigger on and select *Combo...*
- c. In the Combination box, click *Off* next to *Pattern2* and select *On*.
- d. To change the trigger to *Pattern1 and Pattern2*, click the *Or* box to the right of the events and select *And*.
- e. Click on *OK*.
- f. Click on *Close*.

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 62

1. Click *Run*.
2. Examine the data.
  - a. Click the *Navigate* button.
  - b. Point to *Analyzer<A>* in the menu and select *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, right-click 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 43

"The Format Tab" on page 51

"The Trigger Tab" on page 62

## 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 HP 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 HP 16600A-series or HP 16700A logic analysis system, open a logic analyzer setup window.
  - a. In the main window, *right-click* the logic analyzer icon.
  - b. Select *Setup...* from the menu.
  - c. Click 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. Click the *Format* tab.
  - b. Optional - Insert a second label.
    1. Right-click *Label1*.
    2. Select *Insert after...*
    3. In the *Enter Label Name* box, click *OK*.
  - c. Optional - Rename *Label1*
    1. Right-click *Label1*.

2. Select *Rename...*
  3. Enter a new name in the name field.
  4. Click *OK* to close the *Rename Label* box.
- d. Click the bit assignment field.  
The bit assignment field is the field to the right of a label name, and under a pod column.
- e. Select *. . . . . \*\*\*\*\** from the menu.  
If none of the choices match your own system, select *Individual...* and click on the individual bits to assign them (\*) or ignore them (.)
4. Define trigger events for a bus.
- a. Click the *Trigger* tab.
  - b. Click the *Pattern* tab.
  - c. Optional - Rename pattern *Pattern1*.
    1. Double-click in the *Pattern1* field.
    2. Enter a new name.
  - d. Select the appropriate label.
    1. Click the field to the far-right of the label name.
    2. To define the event as a combination of labels, click *Insert...* To use a different label to define the event, click *Replace...*
    3. In the dialog box, click the label name you want to use and then click *OK*.
  - e. Click in the field with *XX* and enter the value you want to trigger on.
5. Define trigger events for an edge.
- a. Click the *Edge* tab.
  - b. Optional - Rename *Edge1*.
    1. Double-click in the *Edge1* field.
    2. Enter a new name.
  - c. Select the appropriate label.

**Making a Basic Timing Measurement**

1. Click the label field immediately to the right of the label name.
2. To define the event as a combination of labels, click *Insert...* To use a different label to define the event, click *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, click the label name you want to use and then click *OK*.
- d. Click the edge assignment field ( . . . . . ) 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. Right-click the *I* box and select *Edit...*
  - b. In the dialog box, click the *Pattern1* button and select *Combo...*
  - c. In the Combination box, click *Off* next to *Edge1* and select *On*.
  - d. Click the *Or* box where the path from *Pattern1* and the path from *Edge1* come together, and select *And*.
  - e. Click on *OK*.  
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 62

1. Click *Run*.
2. Examine the data.
  - a. Click the *Navigate* button.
  - b. Point to *Analyzer<A>* in the menu and select *Waveform<I>*.



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, right-click 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 43

“The Format Tab” on page 51

“The Trigger Tab” on page 62

## 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. *Right-click* the blue *Navigate* button.
  2. Move the cursor over the name of the analyzer whose data you want to view.
  3. *Click* on *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. Click *Search*.
  - b. Click the down arrow after the Label field, and select the label containing the edge.
  - c. Click the Value field and type 1.
  - d. Click *Next* to locate the edge transition.
2. Place a marker on the edge.

Click *Set G1*. 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. Click the down arrow after the Label field, and select the label containing the pattern.
  - b. Click the Value field and type the pattern you are searching for.
  - c. Click the down arrow after the When field and select *Entering*.
  - d. Click *Next* 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.

Click *Set G2*. This will set global marker G2 at the location of the pattern.
5. Find the time between the edge and the pattern.
  - a. Click *Markers*.
  - b. In the G2 row, click 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 clicking the arrow after the Time or Samples field.

**See Also**

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

**Interpreting the Data**

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. Click *Search*.
- b. Click the down arrow after the Label field, and select *ADDR*.
- c. Click the Value field, and type in the starting address, 0x58FC.
- d. Click the down arrow after the When field and select *Present*.
- e. Click *Next* or *Prev* to move the display to the address.

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

Click *Set G1*. 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. Click the Value field, and enter 58FF.
- b. Click *Next* to find the next occurrence of 0x58FF after the starting address.

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

Click *Set G2* 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. Click *Markers*.
- b. In the G2 row, click the second down arrow and select *Sample*.
- c. Click 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 clicking 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)

## Interpreting the Data

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

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

---

## Arming Control - Multiple Instruments and Analyzers

An instrument must be armed before it can look for its *trigger*. If you have not specified any arming actions, by default the instrument is set to be armed immediately when you *Run* the measurement.

You can configure an analyzer instrument to be armed by either another instrument (different slot or frame), or, by the second analyzer within the same instrument if that second analyzer is turned on.

To configure an analyzer instrument to arm other instruments, it must first be included in the Group Run Arming Tree (see the *HP 16600A/16700A Logic Analysis System* help volume). Next, in the analyzer instrument Setup window, go to *Settings* under *Trigger*, and specify the analyzer which will drive the Arm Out signal. The specified analyzer's *Trigger and fill memory* action becomes *Trigger, arm out, and fill memory*.

To change the source of *Arm In* or the destination of *Arm Out*, use the Intermodule Window (see the *HP 16600A/16700A Logic Analysis System* help volume).

### Setting One Analyzer to Arm the Other

1. Activate the second analyzer.
  - a. In *Format*, click *Pod Assignment...*
  - b. In the Pod Assignment dialog, change the analyzer type from *Off*.  
The system pauses while setting up the second analyzer. When it is done, a setup window for the second analyzer appears.
2. In *Trigger*, click the *Trigger Functions* tab.
3. Select *Wait for second analyzer to trigger* from the function list.

4. Select one of the sequence levels next to where you want to insert *Wait for second analyzer*, then click *Insert before* or *Insert after*.  
If the second analyzer isn't activated, a warning message appears and the level is not inserted.

If you are using Advanced functions, you can also insert an *Analyzer <2> triggers* event directly into a sequence level.

### Setting the Analyzer to Wait for Another Module

1. Set up the Intermodule Arming Tree.
  - a. In the System window, click Intermodule.
  - b. In the Intermodule window, click the analyzer's icon and select an "armed by" option.  
*Independent* will remove the analyzer from a group run. *Group Run* will cause the analyzer to be started simultaneously with other instruments in the group run. The other choices will cause the analyzer to wait for a signal from the specified module.
2. In *Trigger*, click the *Trigger Functions* tab.
3. Select *Wait for arm in* from the function list.
4. Select one of the sequence levels next to where you want to insert *Wait for arm in*, then click *Insert before* or *Insert after*.  
If the analyzer isn't armed by another module and group run is not set to *Group run armed from Port In*, a warning message appears and the level is not inserted.

If you are using Advanced functions, you can also insert an *Arm in from IMB* event directly into a sequence level.

---

#### NOTE:

If the trigger sequence does not pass through the level containing the *wait* term, the logic analyzer will not wait for the arming signal.

---

## Loading and Saving Logic Analyzer Configurations

The HP 16715A logic analyzer settings and data can be saved to a configuration file. The configuration file will include references to any custom trigger libraries you have created, but if the configuration is loaded into an analyzer on a system that does not have the trigger libraries they will not work correctly.

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 HP 16715A logic analyzer can load configurations for HP 16715A, 16716A, and 16717A logic analyzers with no restrictions. It can also load configuration files generated for HP 16550A, 16554A, 16555A, 16555D, 16556A, 16556D, 16557D, 16710A, 16711A, 16712A, and 16600A-series built-in logic analyzer but much of the triggering setup will not transfer. Labels, pod and clock assignments, and measurement mode will still work. Setup-and-hold values are translated by preserving the hold value and adjusting setup accordingly.

---

**NOTE:**

The HP 16600A-series and HP 16700A logic analysis systems can translate configuration files from HP 16500 and HP 16505A logic analysis systems if the measurement module is the same. If the modules are different, first load the configuration file into a module of the same model number on *the new logic analysis system*. Re-save the configuration, then load this configuration into the destination module on the new system.

---

**See Also**

Loading Configuration Files (see the *HP 16600A/16700A Logic Analysis System* help volume)

Saving Configuration Files (see the *HP 16600A/16700A Logic Analysis System* help volume)



## When Something Goes Wrong

- “Nothing Happens” on page 34
  - “Error Messages” on page 33
  - “Suspicious Data” on page 34
  - “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 35
  - “Waiting for Trigger” on page 36
  - “Measurement Initialization Error” on page 35
  - “Maximum of 32 Channels Per Label” on page 35
  - “Trigger inhibited during timing prestore” on page 36
  - “Trigger function initialization failure.” on page 40
  - “Goto action specifies an undefined level” on page 41
  - “No Trigger action found in the trace specification” on page 41
-

## When Something Goes Wrong

“No more Edge/Glitch resources available for this pod pair” on page 37

“No more Pattern resources available for this pod pair” on page 38

“Branch expression is too complex” on page 38

“Trigger Specification is too complex” on page 38

“Timer value checked as an event, but no start action specified” on page 39

“Counter value checked as an event, but no increment action specified” on page 39

“Cannot specify range on label with clock bits that span pod pairs” on page 40

---

## 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 *Run* briefly changed to *Stop* or *Cancel*, click the blue *Navigate* button, click the logic analyzer's slot, then select the Waveform or Listing display.

### See Also

“Slow or Missing Clock” on page 35

“Waiting for Trigger” on page 36

---

## 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.

### **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.

### **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 42) on the logic analyzer and contact your HP Sales Office for service or software upgrades.

### **Slow or Missing Clock**

The message "Slow or Missing Clock" only appears in *state measurements*. 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.

## When Something Goes Wrong

- 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 44

“Setting the Pod Threshold” on page 59

## 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  $64 \times 10^3$  samples)

then the approximate time is 32 seconds.

## 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

“Default Storing (State only)” on page 87

“Saving and Recalling Trigger Sequences” on page 72

### No more Edge/Glitch resources available for this pod pair

This error occurs when you have used more than 2 edges or glitches per *pod pair* in the *trigger specification*.

### Possible Solutions

- Phrase some of the edges as patterns.

For example, if you are looking for a rising edge on a read/write line, you can check for  $R/W = 0$  in one level followed by  $R/W = 1$  in the next level.

- Move some of the edges to another pod pair.

Even if a label spans pod pairs, only the edge resources of the pod pair the specific channel is on are used.

## When Something Goes Wrong

### No more Pattern resources available for this pod pair

This error occurs when you have used up all the pattern resources available. Each *pod pair* has about 28 pattern resources. Some pattern events use more than 1 resource.

#### Possible Solutions

- Keep labels within a pod pair

If a label (bus) spans pod pairs (for example, pods 2 and 3) then when you use the label in a *trigger sequence* it will use up at least one pattern resource on both pod pairs. If you hook up your probes in such a way that the signals are on a single pod pair, you can free up a pattern resource on the other pod pair.

- Move some labels to another pod pair

Each pod pair has its own set of pattern resources. Putting your two most-used labels on different pod pairs can improve your resource usage.

### Branch expression is too complex

The "Branch expression is too complex" message means that the trigger sequence compiler could not allocate enough space to evaluate all *events* in the indicated branch.

Other branches may also be too complex. The trigger sequence compiler stops compiling when it encounters the first fatal error.

#### Possible Causes

- More than 16 *events* OR'd together.

Because of hardware limitations, the trigger sequence compiler can only OR together up to 16 simple events.

### Trigger Specification is too complex

The "Trigger Specification is too complex" message means that the *trigger sequence* compiler could not allocate enough combination resources to evaluate the expression.

The root cause of the problem is that the compiler has run out of the

buses it combines events with.

### Possible Causes

- More than 16 *events* OR'd together.

If you have more than 16 events OR'd together, even if they are in different sequence levels, the compiler may run out of combiners. One way to conserve combiner resources is to change ORs to ANDs. This method uses DeMorgan's Theorem and will not help if there are more ANDs than ORs in the trigger sequence.

- a. Click the trigger sequence level and choose *Breakdown function*.
- b. Change ORs to ANDs.
- c. Click *IF* and select *IF NOT*.

- Resources also being used for store qualification.

If *Default Storing* contains *Store by default Custom*, and the events do not match those of the trigger sequence, the logic analyzer has to split resources between the two. If you can change your default storing slightly to match the trigger sequence events, you may be able to use the trigger sequence as originally entered.

### Timer value checked as an event, but no start action specified

This warning occurs because you have used a timer in your *trigger sequence*, but do not start it with either *Start from reset* or *Resume* in any action. You do not need to start the timer in the same sequence level. The timer will still function if not started, but will not change value.

### Counter value checked as an event, but no increment action specified

This warning occurs because you have used a counter in your *trigger sequence*, but do not have *Counter Increment* as an action. You do not need to increment the counter in the same sequence level. The counter event will still function, but will not change value. The default value for both counters is 0.

## When Something Goes Wrong

### Trigger function initialization failure.

The "trigger function initialization failure" messages mean that you tried to insert a trigger function which required a change in your setup.

#### Possible Causes

- Tried to insert "Wait for arm in" trigger function

A "Wait for arm in" trigger level causes the logic analyzer to wait for a signal from another module or Port In. These signals are passed through the Intermodule Bus. To prevent the logic analyzer from hanging, it must be added to the Group Run Arming Tree. To do this, open the Intermodule window by clicking the Intermodule icon in the System window. In the Intermodule window, click the analyzer icon and select any option except for Independent.

- Tried to insert "Wait for other machine to trigger" function

A "Wait for the other machine to trigger" trigger level causes the logic analyzer to wait for a signal from the other logic analyzer *machine* in the module. If this machine is not on, the current logic analyzer will hang. To turn the other machine on, click *Pod Assignment* under *Format*. Set the type of the other machine to *State* or *Timing*.

- In *Format*, no *labels* have bits assigned to them.

When you insert a trigger function, the logic analyzer sets up a field for you to enter values. The field length is based on the number of bits assigned to the first active label, or the label you specify. If there are no bits assigned to the label, the logic analyzer cannot complete the value field.

#### See Also

Using the Intermodule Window (see the *HP 16600A/16700A Logic Analysis System* help volume)

“Assigning Bits to a Label” on page 53

### Cannot specify range on label with clock bits that span pod pairs

A label that contains clock bits being used as data bits, can only be included in a range term if the clock bits are confined to a single pod pair.



### **Goto action specifies an undefined level**

The "undefined level" messages mean that the trigger sequence contains goto statements that point to non-existent levels. This is detected when the trigger sequence is evaluated. The logic analyzer will not run if there are undefined levels, even if there is no possibility of the goto sequence being called.

#### **Possible Causes**

- The last sequence level calls "goto next"

To check this, click *Overview* under the Trigger tab.

To fix, find the "goto next" statement and change it to point to an existing level.

### **No Trigger action found in the trace specification**

This warning occurs when the trigger sequence you specified does not have at least one *trigger and fill memory* or *trigger and goto* action. The analyzer will still acquire data, but you will need to manually stop it.

## 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 *HP 16600A/16700A Logic Analysis System* help volume)
3. Disconnect all loads, adapters, or preprocessors from the probe cable ends.
4. From the system window, click the *System Admin* icon.
5. Click the *Admin* tab, then *Self Test...*  
The system closes all windows before starting up Self Test.
6. Click *Master Frame*.  
If the module is in an expansion frame, click *Expansion Frame*.
7. Click the logic analyzer that you want to test.
8. In the Self Test dialog box, click *Test All*.  
You can also run individual tests by clicking on them. Tests that require you to do something must be run this way.

If any test fails, contact your local Hewlett-Packard Sales Office or Service Center for assistance.

### **See Also**

Self Test (see the *HP 16600A/16700A Logic Analysis System* help volume)

*HP 16715A 167 MHz State/667 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 47

“Turning the Analyzer Off” on page 48

“Setting the Acquisition Mode” on page 44

“Sample Period (Timing Only)” on page 48

“Performing Clock Setup (State only)” on page 44

“Trigger Position Control” on page 49

“Acquisition Depth” on page 43

---

## Acquisition Depth

*Acquisition Depth*, located under *Sampling* and also under *Trigger Settings*, sets how much data will actually be acquired. While the HP 16715A logic analyzer has a maximum memory depth of 2 M samples in State Mode and 4M 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 HP 16715A 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 Mode” on page 49

“Timing Acquisition Modes” on page 49

---

## 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 45.

### 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...* Clicking 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 45

### 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

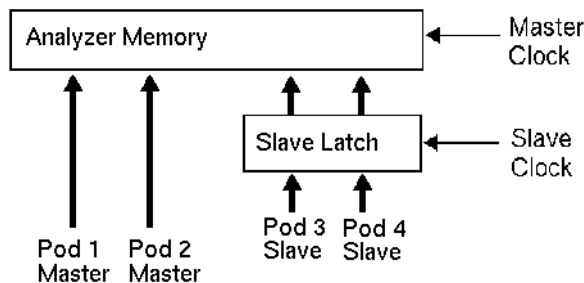
Data on all pods assigned to *Master Clk* is strobed into memory when the status of the clock lines match the clocking arrangement specified for *Master* in the clock setup.

## The Sampling Tab

### 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.

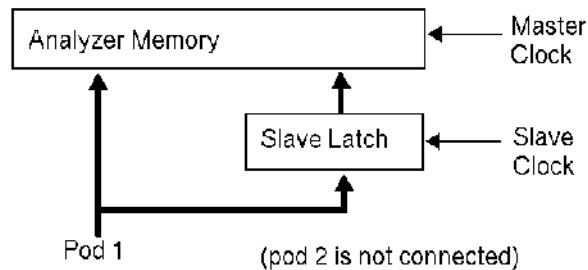


### Latching Slave Data

#### 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 58). 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. Click the *Analyzer Name* field.
2. Type in 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 clicked *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 clicking 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, click 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 Mode

### 167 MHz / 2M State

All pods are available. Memory depth is 2 M samples per *channel*. If time or state *count* is turned on in *Trigger Settings*, the total memory is split between data acquisition storage and time or state count storage. To maintain the full memory depth of 2 M samples per channel, leave one pod pair unassigned. (To unassign a pod pair, click *Pod Assignment* under *Format*, then drag a pod pair to unassigned.) State clock speed matches your target system's clock, up to 167 MHz.

## Timing Acquisition Modes

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

### 333 MHz Full Channel 2M Sample

The total memory depth is 2 M samples per *channel*, with data being sampled and stored as often as every 3 ns. You can set the sample rate to go slower with the Sample Period control.

### 667 MHz Half Channel 4M Sample

Only one pod of each pod pair is available. Channels assigned to unavailable pods are ignored. You can specify which pod to use by toggling the *Pod* field in *Format*. The total memory depth is 4M samples per *channel*. Data is sampled and stored every 1.5 ns; this rate cannot be changed.

#### See Also

“Sample Period (Timing Only)” on page 48

“Pod Selection” on page 58

---

## 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

## The Sampling Tab

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, or User Defined.

- 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.

---

## 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 57

“Setting the Pod Threshold” on page 59

“State Clock Setup/Hold (State only)” on page 59

### Advanced Measurement Controls

“Assigning Pods to the Analyzers” on page 52

“Clock Modes (State only)” on page 45

“Setting Up the Pod Clock” on page 57

“Pod Selection” on page 58

### See Also

“Data On Clocks Display” on page 53

“Activity Indicators” on page 51

“Assigning Bits to a Label” on page 53

“Inserting and Deleting Labels” on page 54

“Turning Labels On and Off” on page 55

“Reordering the Bits in a Label” on page 56

“Label Polarity” on page 55

---


## Activity Indicators

Activity indicators are the arrows and dashes associated with the pods.

---

**The Format Tab**

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 55)

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 59

*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 *Format*, click *Pod Assignment...*  
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 3/4 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, click *Off* and select *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 53

“Activity Indicators” on page 51

## 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.

## The Format Tab

### To Assign Bits

1. Click the bit assignment field to the right of the label name you want to define.  
Each bit assignment field corresponds to the data pod which is listed above it.
2. Either select one of the predefined groups, or *Individual*.
3. In *Individual*, click 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**

“Reordering the Bits in a Label” on page 56

“Activity Indicators” on page 51

## Inserting and Deleting Labels

### To Insert Additional Labels

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

### To Delete Labels

1. Click 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 55) 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. *Right-click* 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. Right-click 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, click *Apply*.  
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:

## The Format Tab

- "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.

## Reordering the Bits in a Label

The bit reorder feature allows the channel order, as it appears in the label, to be assigned independently of the physical order. This feature allows the probe tips for each channel to be physically connected where convenient. The *Reorder* function can then be used to logically rearrange the bits in a label.

---

**NOTE:**

---

Reordered labels can not be used as *range terms* in triggers.

### To Reorder the Bits in a Label

1. Click the label name that you want to reorder.
2. Select *Reorder Bits*.
3. Set the bit order by using one of the following options:
  - To reorder the bits individually, for each channel, type the number of the bit you want to map the channel to. You can also use the *Spin Buttons* to scroll through the list of bits.
  - To arrange the bits sequentially, click the button at the top of the dialog, then select *Default Order*.
  - To swap the high and low order bytes or words, click the button at the



top of the dialog, then select *Big-Endian/Little-Endian*.

4. Click *OK*. The label now shows an "R" to indicate bit assignment.

---

## 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, but this places restrictions on the complexity of the trigger later.

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

### To Define a Label

1. (Optional) *Click* the label name field and select *Rename*
2. Type a new name and click *OK*.
3. Assign bits (see page 53) to the label.
4. If necessary, insert more labels (see page 54) in the list.

### See Also

“Assigning Bits to a Label” on page 53

“Reordering the Bits in a Label” on page 56

“Inserting and Deleting Labels” on page 54

“Turning Labels On and Off” on page 55

“Label Polarity” on page 55

---

## 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

**The Format Tab**

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, click the Pod Field.

**See Also**

“Performing Clock Setup (State only)” on page 44

“Clock Modes (State only)” on page 45 for details on slave and demultiplex clocks

“Pod Selection” on page 58

## 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 44

“Performing Clock Setup (State only)” on page 44

---

## 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.

### To Set the Threshold

1. Under the *Format* tab, click the threshold field.  
The threshold field is located just below the pod name.
2. In the Pod threshold dialog, choose one of the threshold options described below.
3. If you do not want the change to apply to all pods, click the checked box next to *Apply settings to all pods*.
4. Click *Close*.

---

**NOTE:**

The clock threshold level is the same as the level assigned in the Pod Threshold field.

### TTL

The threshold level is +1.5 volts.

### ECL

The threshold level is -1.3 volts.

### USER

When USER is selected, the threshold level is selectable from -6.0 volts to +6.0 volts.

---

## 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

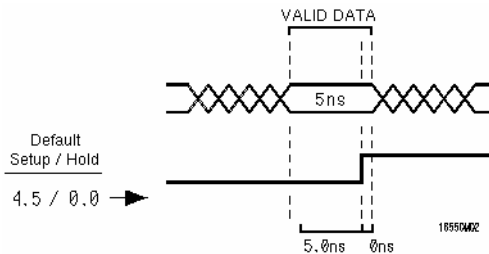
**The Format Tab**

when the analyzer is set up for a *state measurement*.

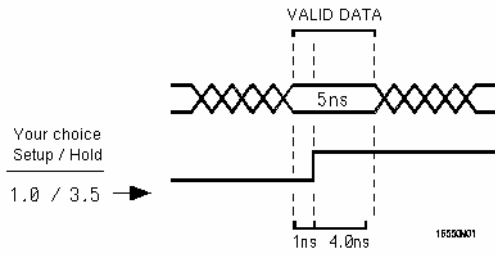
**To Change Clock Setup/Hold**

1. Click the *Setup/Hold* button.
2. For each label, enter a setup and hold. The values are adjustable in 100 ps increments, with a fixed window of 2.5 ns.
3. If you need to adjust *bits* individually,
  - a. Select a label containing the bit.  
If a bit is used in more than one label, this will change its setup and hold for its use in *all* labels.
  - b. Click *Individual bits*.
  - c. Enter the bit number you want to change.
  - d. Adjust the setup and hold.
4. Click *OK*.

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*. In the HP 16715A logic analyzer, you can insert multiple trigger actions. When you insert multiple trigger actions, the trigger marker in the Display windows is placed on the first sample whose evaluation caused a branch through an associated trigger action.

The Trigger tab has two main areas: On top, tabs of functions and controls to build your trigger; and 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 65
- “Setting Up a Trigger” on page 68
- “Inserting and Deleting Sequence Steps” on page 69
- “Editing Sequence Levels” on page 71
- “Setting Up Loops and Jumps in the Trigger Sequence” on page 71
- “Saving and Recalling Trigger Sequences” on page 72
- “Clearing Part or All of the Trigger” on page 74

**See Also**

“Overview of the Trigger Sequence” on page 75

“Trigger Functions” on page 76

“Trigger Function Libraries” on page 73

“Working with Advanced Functions” on page 85

“Defining Events” on page 88

“Trigger Position Control” on page 49

“E-mail Notify on Trigger” on page 64

“Sample Period (Timing Only)” on page 48

“Tagging Data with Time or State Tags (State Only)” on page 91

“Default Storing (State only)” on page 87

“Arming Control - Multiple Instruments and Analyzers” on page 30

## E-mail Notify on Trigger

The *E-mail on Trigger* option is set within the trigger sequence. It directs the logic analyzer to e-mail you with a notice informing you that the analyzer has triggered and has started to fill memory. The automatically generated text is shown as follows:

Example: *system14 : Slot C : Analyzer C has triggered*

Where *system14* is the analysis system IP address or alias you have assigned to it; *Slot C* is the frame slot the module is in; *Analyzer C* identifies the specific analyzer module from others when configured in a multi-module frame configuration. Any text you add in the text entry area of the e-mail setup dialog will appear after the automatically generated text.

It should be noted that if you have specified the analyzer to store all states, and you are using a deep memory analyzer, even though you have been notified of a trigger, the analyzer could be busy filling memory for a period after your e-mail notification.

### Setting Up E-mail on Trigger

The following procedure is performed under the *Trigger - Trigger Functions* tabs, and from the trigger sequence level(s) that include the trigger action. If you have a trigger sequence that includes multiple trigger actions (multi-level branching), and one trigger action includes sending e-mail, then e-mail is sent if any sequence level triggers. In other words, you cannot designate a different e-mail destination for each trigger action in a multi-level branching trigger sequence setup.

1. From a trigger sequence level that includes the trigger command, select the trigger action field and set it to *Trigger, send e-mail, and fill memory*.
2. Select the *Setup* field that appears.
3. Type in the name of your SMTP (see page 65) mail server. Contact your *System Administrator* if you do not know this information.
4. Type in the e-mail address(es) of who you want the message sent *To*:. Use a space to separate multiple addresses.



5. Type in the e-mail address of who the message is being sent *From:*. By default, this is the *login name* followed by @ followed by the *hostname* of your logic analyzer. If the SMTP server has a problem with the default sender address, you may want to specify one that is recognizable by the server. A possible address might be the one specified in the *To:* field.
6. Select *Email on Repetitive Run* if you want mail at each trigger of a repetitive run.
7. Select *OK*.

---

## What is SMTP

SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. A protocol is the special set of rules for communicating the end points in a telecommunication connection as they send signals back and forth.

Protocols exist at several levels in a telecommunication connection. There are hardware telephone protocols. There are protocols between the end points in communicating programs within the same computer or at different locations. Both end points must recognize and observe the protocol.

On the Internet, there are the following TCP/IP protocols:

- TCP (Transmission Control Protocol), which uses a set of rules to exchange messages with other Internet points at the information packet level.
- IP (Internet Protocol), which uses a set of rules to send and receive messages at the Internet address level.
- HTTP, FTP, SMTP and other protocols, each with defined sets of rules to use with other Internet points relative to a defined set of capabilities.

---

## Understanding Logic Analyzer Triggering

What is a Trigger (see page 66)

## E-mail Notify on Trigger

What does "Trigger Position" Mean (see page 66)

What can be Used to Specify a Trigger (see page 66)

When to use a Combination, a Branch, or a Level (see page 67)

### 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 2 M 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.

**Label events** Label events can look for a pattern on a bus, such as  $ADDR < 0880$  or  $R/W = 1$ . They can also look for values in (or out) of a range.

**Timer** Timers are started in one sequence level and checked in another. They act like stopwatches.

**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 *ANDs*.

**Combinations of Events** To check for more than one type of thing happening *in the same sample*, combine events within a sequence level using *AND* and *OR*.

### 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 71 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.

**See Also**

“Setting Up a Trigger” on page 68

“Defining Events” on page 88

“Using Group Events” on page 91

---

## 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 levels until the analyzer consistently triggers at the desired point.

### To Set Up a Trigger

1. Select the most appropriate trigger function and click replace. (see page 69)
2. Define events. (see page 88)
3. If necessary, insert additional sequence levels. (see page 69)

When creating a trigger sequence to describe the sequence of pertinent events on your target system, you need to know if you are looking for

- events happening at the same time,
- different events that might be happening next, or
- a sequence of events.

For events happening at the same time, set up the trigger sequence to look for multiple events in the same level.

For different events that might be happening, the trigger sequence depends on what you want to do based on the events. If you want to take different actions or look for different occurrences afterwards, use separate branches (If-then-else if) within a sequence level. If it is only that different actions could lead to the same problem, group the events using AND and OR within a single branch of a single sequence level.

For a sequence of events, check for each event in a separate sequence

level.

**See Also**

“Trigger Functions” on page 76

“Working with Advanced Functions” on page 85

“Overview of the Trigger Sequence” on page 75

“E-mail Notify on Trigger” on page 64

## 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 click *Replace* or *Insert*.

**See Also**

“Trigger Functions” on page 76

---

## 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. Click the sequence step that you want to insert other steps around.
2. Select a trigger function from the list under *Trigger Functions*.
3. Click *Insert Before* or *Insert After*.

### To Delete Sequence Steps

1. Click the sequence step that you want to delete.

## E-mail Notify on Trigger

2. Select *Delete*.

### Trigger Sequence Editing Options

When you *click* a sequence step, 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 Levels

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

### To Edit a Sequence Level:

1. Click the sequence step to make it editable.  
A yellow box appears around the level, and the editable fields become controls.
2. To replace it with a different function, double-click the desired function in the event list.
3. Click on an *event* to replace it with a different event, or insert additional events.
4. Set the values of the other fields, such as durations and occurrence counts.

### See Also

“Setting Pattern Duration and Occurrence Count” on page 86

“Using Occurrence Counters” on page 87

“Using Timer Terms” on page 90

“Setting Up Loops and Jumps in the Trigger Sequence” on page 71

---

## Setting Up Loops and Jumps in the Trigger Sequence

Using *Goto* statements, you can set up loops or simple jumps in the *trigger sequence*. Loops are useful when you want to trigger after a certain number of iterations. Jumps can be used to implement loops or just to set up parallel branches that look for different types of triggers.

Branches are implemented in the HP 16715A logic analyzer in the Advanced functions. You can also access branches in the other trigger functions by breaking them down (see page 84).

**E-mail Notify on Trigger**

---

**NOTE:**

Use branches within a trigger sequence level when you want to test a single sample for multiple conditions and take different actions based on which is true. Use different sequence levels when you want to test different samples.

---

**To Set Up a Branch or Loop**

1. Set up the events in the first *If* statement. If these events are found in the incoming data, the analyzer will execute the actions in the *then* portion. By default, one of the actions is *Goto next*.
2. To insert an *Else if* branch, click *If* and select *Insert branch after*.

**See Also**

“Breaking Down and Restoring Functions” on page 84

---

## Saving and Recalling Trigger Sequences

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

The HP 16715A logic analyzer provides memory locations to store up to 15 trigger sequences per *machine* for both state and timing. Five of the 15 memory positions are reserved for the last 5 runs. If you think you will want to recall a sequence after several runs, save it in one of the other 10 memory locations.

When you exit your HP 16600A-series or HP 16700A *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. You can also save parts of trigger sequences independently of configuration files by creating a trigger function library. (see page 73)

**To Save a Trigger Sequence**

1. In *Trigger*, click the *Save/Recall* tab.
  2. Click *Save*.
-



3. Select a memory location to store the trigger sequence in.
4. In the Buffer Name dialog, type a descriptive name for the trigger sequence.

### To Recall a Trigger Sequence

1. In *Trigger*, click the *Save/Recall* tab.
2. Click *Recall*.
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. Also, if the trigger sequence uses a trigger function library that does not exist on this mainframe, it will not load correctly.

### See Also

“Loading and Saving Logic Analyzer Configurations” on page 32

“Trigger Function Libraries” on page 73

---

## Trigger Function Libraries

Trigger function libraries allow you to create your own libraries of trigger sequences without having to save and load entire configuration files. They can be copied to other logic analysis systems and loaded into other logic analyzers with the function library capability.

When functions from trigger libraries are inserted into the trigger sequence, they can only be edited by breaking down the function. If a *trigger sequence* or configuration file uses a library that has been deleted, or a function that has been deleted from a library, the logic analyzer replaces the missing function with the default trigger function.

### To Add Functions from Libraries into the Trigger Function List

1. Click *Trigger function libraries...*
2. Select the library from the list.  
Only libraries created in the same acquisition mode are available.

## E-mail Notify on Trigger

3. Click *Load*.

All of the library's functions are added to the list of Trigger Functions. The first function from the library is now selected in the list.

### To Copy Trigger Function Libraries between Systems

1. Connect your logic analysis system to the network. (see the *HP 16600A/16700A Logic Analysis System* help volume)
2. Using a computer on your network, copy `/hplogic/trigger_functions/` to a central location, or directly to other logic analyzers on the network.

---

## Clearing Part or All of the Trigger

### To Clear Part or All of the Trigger:

1. Click *Trigger*.
2. Select *Clear* from the *menu bar*.
3. Choose the option you want from the choices described below:
  - All  
Clears the *trigger sequence*, acquisition depth, and trigger position back to their default values. Turns on Count Time. If the analyzer is set to State, also sets default storing back to store anything.
  - Trigger Sequence  
Resets the *trigger sequence* to the default sequence for the analyzer acquisition mode.
  - Settings  
Returns settings (acquisition depth and trigger position) to default settings. In state measurements, time tags are turned back on. In

timing measurements, the sample period returns to its fastest setting.

- Save/Recall Memories  
Deletes all saved trigger sequence specifications.
  
- Clear Default Store (State only)  
Returns default storing to *Store anything*.

---

## Overview of the Trigger Sequence

The *trigger sequence* is a sequence of steps (the levels) 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 levels (see page 69).

Each step in the sequence is a trigger function. Most trigger functions are tailored to a specific type of measurement. The "Advanced" trigger functions let you work with the bare resources that the analyzer hardware uses. Both types of function are composed of events to evaluate, and actions to take.

When you run the analyzer, it searches for a match between the events and the measurement data. When a match is found, the logic analyzer executes any actions specified. Each function must have as one of its actions a *Goto* or *Trigger and fill memory*. This action controls what level the analyzer evaluates next.

Each trigger function uses one or more of the analyzer's internal sequence levels (see page 85). Each Advanced function uses one internal sequence level.

### See Also

“Understanding Logic Analyzer Triggering” on page 65 for more detail

“Setting Up a Trigger” on page 68 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 85), 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 78

“Pattern/Edge Combinations” on page 79

“Time Violations” on page 79

“Timing User Level” on page 76

### **State Trigger Functions**

“Basic State Trigger Functions” on page 82

“Sequence-Dependent Trigger Functions” on page 82

“Time Violations” on page 83

“State Advanced Functions” on page 80

### **See Also**

“Setting Up a Trigger” on page 68

“Defining Events” on page 88

“Breaking Down and Restoring Functions” on page 84

## **Timing User Level**

The Advanced trigger functions allow you to create a custom *trigger sequence* using events, comparison functions, and up to 4 branches. All Advanced trigger functions use one internal sequence level.

The types of events include labels, timers, flags, and counters.

- Advanced - If/then

This function has only one branch. If the events in the "If" event list are true, it executes the actions after "then".

- **Advanced - 2-way branch**  
This function has two branches, of the form "If - then; else if - then". Each sample, the events in the first "If" branch are checked. If all events are true, then the "then" portion is executed. If they are not true, the events in the "else if" branch are checked. If the "else if" events are true, that "then" portion is executed. If neither branch is true, the logic analyzer remains in this sequence level and repeats the comparison with the next sample.
- **Advanced - 3-way branch**  
This function has three branches, of the form

```
If (events1)
  then (actions1)
Else if (events2)
  then (actions2)
Else if (events3)
  then (action3)
```

The logic analyzer evaluates each sample against the clauses in the order they are specified. The logic analyzer executes the set of actions in the "then" clause associated with the first listed "if" or "else if" clause that becomes true.

- **Advanced - 4-way branch**  
Like the 3-way branch, but with 3 "Else if" clauses.
- **Advanced - pattern1 AND pattern2**  
Searches for two different patterns occurring in the same sample. If you set it to look for more than 1 occurrence, you can specify whether occurrences are consecutive or not. You can also add other events, including labels, to be searched for.

## E-mail Notify on Trigger

- **Advanced - pattern1 OR pattern2**  
Finds either pattern1 or pattern2 or both in a sample. If you set it to look for more than 1 occurrence, you can specify whether the occurrences are consecutive or not. You can also add other events, including labels, to be searched for.

### See Also

“Working with Advanced Functions” on page 85 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.

## E-mail Notify on Trigger

- 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 Advanced Functions

The Advanced trigger functions allow you to create a custom *trigger sequence* using events, comparison functions, and up to 4 branches. All Advanced trigger functions use one internal sequence level.

The types of events include labels, timers, flags, and counters.

- Advanced - If/then  
This function has only one branch. If the events in the "If" event list are true, it executes the actions after "then".
- Advanced - 2-way branch  
This function has two branches, of the form "If - then; else if - then". Each sample, the events in the first "If" branch are checked. If all events are



true, then the "then" portion is executed. If they are not true, the events in the "else if" branch are checked. If the "else if" events are true, that "then" portion is executed. If neither branch is true, the logic analyzer remains in this sequence level and repeats the comparison with the next sample.

- Advanced - 3-way branch  
This function has three branches, of the form

```
If (events1)
  then (actions1)
Else if (events2)
  then (actions2)
Else if (events3)
  then (action3)
```

The logic analyzer evaluates each sample against the clauses in the order they are specified. The logic analyzer executes the set of actions in the "then" clause associated with the first listed "if" or "else if" clause that becomes true.

- Advanced - 4-way branch  
Like the 3-way branch, but with 3 "Else if" clauses.
- Advanced - pattern1 AND pattern2  
Searches for two different patterns occurring in the same sample. If you set it to look for more than 1 occurrence, you can specify whether occurrences are consecutive or not. You can also add other events, including labels, to be searched for.
- Advanced - pattern1 OR pattern2  
Finds either pattern1 or pattern2 or both in a sample. If you set it to look for more than 1 occurrence, you can specify whether the occurrences are consecutive or not. You can also add other events, including labels, to be searched for.

**See Also**

“Working with Advanced Functions” on page 85 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.

## E-mail Notify on Trigger

- 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

In the HP 16715A logic analyzer, you can either expand functions, or break them down. When you expand a function the logic analyzer shows you how the function would be implemented using Advanced functions, but you cannot edit the values. Expanded functions can be compressed back into their original form.

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 add events or branches to the more basic functions. When you break down a trigger function, you can return to the compressed form by selecting *Edit -> Undo* if you have not made any changes.

### To Break Down Trigger Functions

1. *Click* the level number.
2. At the bottom of the menu, select *Break down function*. 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 85) both levels are separated out and displayed in the trigger sequence area.

### To Restore Functions

1. Click the level number.
2. Choose *Compress function*.

### See Also

“Working with Advanced Functions” on page 85 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 85) 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 76), 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 Advanced Functions

---

### NOTE:

Before you begin to set up advanced sequence steps, note that in most cases one of the predefined trigger functions (see page 76) will work.

You might need to set up an advanced sequence step to accommodate a condition not covered by the other functions, or if you need to set up additional loops and jumps in the sequence. Each advanced sequence step has a "fill-in-the-blanks" type statement like the more basic functions, but does not place restrictions on what you can add.

## E-mail Notify on Trigger

### To Set Up an Advanced Function

1. In *Trigger Functions*, select one of the *Advanced* functions at the end of the list.
2. Click *Replace* or *Insert*.
3. Fill in the events as you would for a non-advanced function.
4. To change the event list in a branch,
  - a. click an existing event
  - b. choose the appropriate operation and event type from the list.  
Timers, counters, and flags do not change value unless you insert an action in some level.
  - c. set the event values that you are looking for
  - d. verify that events are properly combined with OR and AND.
5. To change the actions in a branch, click an existing action then choose the appropriate operation and action type. All actions in an action list are executed when a branch is followed.  
If you specify *Goto Next*, make sure there is a next level or your logic analyzer will not run.
6. To insert a branch, click *If* or *Else if* and choose one of the operations from the menu.

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

- “Defining Events” on page 88
- “Setting Pattern Duration and Occurrence Count” on page 86
- “Using Occurrence Counters” on page 87
- “Setting Up Loops and Jumps in the Trigger Sequence” on page 71
- “Using Timer Terms” on page 90

### Setting Pattern Duration and Occurrence Count

(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 easiest way to do this is by using the *Find pattern present for > duration* and *Find pattern present for < duration*.

By breaking down the function (see page 84), you can also access the (present for >/occurs) control. This is also available in the Advanced functions.

### To Set a Pattern Duration in Advanced Functions

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

When greater-than (>) is used, the analyzer continues sequence level evaluation only after the event has been true for greater than or equal to the time specified.

To evaluate a pattern for less-than (<), you need four internal sequence levels (see page 85). Copy the logic and branching used in the *Find pattern present for < duration* function. You can expose the logic by inserting the function, clicking the sequence level number, and selecting *Expand function*.

When occurs is selected, you can set how many times the event must appear, and whether the appearances must be consecutive or not. In functions that do not show the *consecutively/eventually* control, the 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.

**Default Storing (State only).** When the analyzer is set to State, you

**E-mail Notify on Trigger**

can maximize memory by customizing what gets stored in memory. The controls are located in *Trigger*, under the *Default Storing* tab.

If you do not change anything, the logic analyzer is set to *Store anything*, which stores all samples. *Store nothing* prevents samples being stored, and must be overridden with a *Store sample* or *Turn off default storing* in the actions for the individual trigger sequence levels.

*Store custom* lets you filter the data you want to store. You can specify ranges of values that mark data of interest. You can also use the flags or timers to signal to the analyzer when you want to store data. (Be sure to set the flags or other events in the actions for the trigger sequence level.)

The HP 16715A logic analyzer does not use the "Branches taken" feature of earlier logic analyzers. The best way to simulate "Branches taken stored" is by setting up *Store by default nothing*, a *store sample* in each action sequence, and wherever you trigger, *Turn off default storing*.

---

**NOTE:**

When Store Qualification is performed in the 333 MHz State mode, there may be the case where data occupying memory is further disqualified. As a result, you may see a non-contiguous listing of states as well as a reduction of usable memory.

---

---

## Defining Events

Events are defined in the *trigger sequence*. The most common event is a label. Other types of events are timers, counters, and flags. Label events let you specify patterns or ranges on a bus, or (in a *timing measurement*) edges and glitches.

Trigger functions restrict what sort of events you can insert. The instructions below cover the general case for labels. If you need to insert an event that a trigger function does not allow, first search the list for a more appropriate trigger function. If one does not exist, break down the trigger function (see page 84) and then insert the event.



### To Define a Label Event

1. In a trigger sequence level, select a label and choose *Insert* or *Replace*.
2. Select the operator field to the right and specify an operation.  
*Edge* operators are only available in timing measurements, and not in all trigger functions.
3. Specify the value you want to find on the label.  
*X* means you don't care about the values on the specified bits, and are not allowed in ranges. Edges by default look for any rising edges in the label. Click the edge assignment to change it.



*Right-click* on any of the value fields to quickly assign 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.

### Notes on Specific Event Types

**Ranges** The *In range* and *Not in range* operators consider the values you enter as the endpoints to be inside the range. Ranges cannot be set on reordered labels.

**Edges** The *Edge* operator ORs specified edges together. If you need to AND edges, assign them to different events and join those with an AND.

**Flags** Flags can be used to signal between HP 16715A, 16716A, and 16717A modules. One side effect of this is that the 8 flags are shared across all modules in the system. If you have some modules in an HP 16701A expander frame, use flags 1 through 4 to signal from modules in the HP 16701A to modules in the HP 16700A, and flags 5 through 8 for the other direction. If all modules that use flags are installed in a single frame, you can use any of the 8 flags. By default, flags are *Clear*, and show as 0 in the Status tab.

**Timers and Counters** The default values for timers are 0 ns and counters are 0. Their values are changed by inserting appropriate actions. Start timers running with *Timer Start from reset* or *Timer resume* in an action.

**E-mail Notify on Trigger****See Also**

“Using Timer Terms” on page 90

“Using Group Events” on page 91

“Numeric Base” on page 91

“Breaking Down and Restoring Functions” on page 84

**Using Timer Terms**

Timers are like other *events* in that they are either true or false. They are controlled within the *trigger sequence* and act like a stopwatch. Timers start at 0, and can be set to Start, Stop, Pause, or Resume as the analyzer enters a trigger sequence level.

The only restriction timers have is that no timer is available for the first *pod pair* assigned in a configuration. After the first pod pair, there is a timer available for each additional pod pair that is assigned to the measurement.

Timers are not started until you insert a *Start from reset* or *Resume* action in some sequence level. Timers continue doing whatever instruction they last received until either the acquisition completes or they receive a new instruction.

The minimum value you can test a timer for depends on the acquisition mode of the measurement.

**To Include a Timer in a Trigger Sequence**

1. Click an event.
  - If the list mentions EVENTS, select an *Insert EVENT* or *Replace EVENT*, and choose *Timer*.
  - If the list does not mention EVENTS, break down (see page 84) the level, then click an event.
2. Specify which timer, and the time value.
3. Click the sequence level which should start the timer.
4. Click *Trigger* or *Goto*, and select *Insert action -> Timer*, and one of the actions.
5. Insert other timer actions in the same way as Step 4 as appropriate in

other sequence levels.

## Using Group Events

To setup an advanced sequence that evaluates multiple events as a group, instead of separate sequence levels (more common), you must reorganize all desired events in parens (). An example is when you use parens () in a complicated math equation.

Adding parens () for the purpose of grouping trigger events is a more advanced operation that can only be accessed through the "IF" operator in a sequence level. The "IF" operator is only available when you break down a Trigger Function, or, you use one of the advanced functions that contain the "IF" operator.

When the "IF" operator is available, select the "IF" operator field, then select *Group Events*. From the dialog that appears, follow the directions on placing parens () at the beginning and end of events.

## 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. Click the base 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

## E-mail Notify on Trigger

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.

## Specifications and Characteristics

---

**NOTE:**

Definition of Terms To understand the difference between specifications (see page 93) and characteristics (see page 93), and what gets a calibration procedure (see page 94) and what gets a function test (see page 94), refer to appropriate links within this note.

“HP 16715A Logic Analyzer Specifications” on page 94

“HP 16715A Logic Analyzer Characteristics” on page 95

---

### 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 HP 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.

---

## HP 16715A Logic Analyzer Specifications

The specifications are the performance standards against which the product is tested. These specifications apply only to the HP 16715A 167 MHz State/667 MHz Timing logic analyzer:

Maximum State Clock Speed:	167 MHz
Threshold Accuracy:	+/- (65 mV + 1.5% of threshold setting)
Minimum Master-to-Master Clock Time:	5.988 ns
Setup/Hold Time:	

- \*Single Clock, Single Edge: 4.5/-2.0 ns through -2.0/4.5 ns, adjustable in 100-ps increments
- \*Multiple Edges: 5.0/-2.0 ns through -1.5/4.5 ns, adjustable in 100-ps increments
- \* Specified for an input signal  $V_H=-0.9$  V,  $V_L=-1.7$  V, threshold=-1.3 V, slew rate=1 V/ns

---

## HP 16715A Logic Analyzer Characteristics

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

### General information

- Channel Counts:
  - 1-card module 64 data, 4 clock
  - 2-card module 132 data, 4 clock
  - 3-card module 200 data, 4 clock
  - 4-card module 268 data, 4 clock
  - 5-card module 336 data, 4 clock
- Memory Depth:
  - Half Channel 4 M samples per channel
  - Full Channel 2 M samples per channel

Half Channel mode is only available for timing analysis.

### Probes (at end of flying lead set)

- Input Resistance: 100 Kohm, +/- 2%
- Parasitic Tip Capacitance: 1.5 pF
- Minimum Voltage Swing: 500 mV peak-to-peak
- Minimum Input Overdrive: 250 mV
- Maximum Voltage: +/- 40 V peak CAT I
- Threshold Range: +/- 6.0 V, adjustable in 10-mV increments
- Power through pod cables: 1/3 amp per pod

### State Analysis

- Maximum State Clock Speed: 167 MHz
- \*Minimum Setup/Hold Time: 4.5/-2.0 ns through -2.0/4.5 ns, adjustable in 100-ps increments
- Minimum State Clock Width: 1.2 ns
- Minimum Master-to-Master Clock: 5.988 ns
- Minimum Master-to-Slave Clock: 2 ns
- Minimum Slave-to-Slave Clock: 5.988 ns
- State Clocks: 4
- State Clock Qualifiers: 4
- \*\*Time Tag Resolution: 4 ns
- Maximum Time Count Between States: 17 seconds
- \*\*Maximum State Tag Count: 2e32
- Store qualification Default and per sequence level

\* Specified for single-edge, single-clock acquisition.

Multi-edge setup/hold window is 3.0 ns.

\*\* When all pods are being used, time or state tags halve the memory depth.

### Timing Analysis

- Maximum Conventional Timing Rate:
  - Half Channel 667 MHz
  - Full Channel 333 MHz
- Sample Period:
  - Half Channel 1.5 ns
  - Full Channel 3 ns to 1.0 ms
- Sample Period Accuracy: +/- (100 ps + 0.01% of sample period)

# Chapter 1: HP 16715A 167 MHz State/667 MHz Timing Logic Analyzer

## Specifications and Characteristics

- Channel-to-Channel Skew: less than 1.5 ns, typical
- Time Interval Accuracy: +/- ( sample period + channel-to-channel skew + 0.01% of time interval reading )

### Triggering

- Maximum Trigger Sequencer Speed: 167 MHz
- State Sequence Levels: 16
- Timing Sequence Levels: 16
- Sequence Level Branching: Arbitrary 4-way "If/then/else"
- Maximum Occurrence Count Value: 16,777,215
- Pattern Recognizers: 16
- Range Recognizers: 15
- Range Width: 32 bits
- Occurrence Counters: 1 per sequence level
- Global Counters: 2
- Flags: 8, can be used between modules
- Flag set/reset to evaluation: 110 ns, typical
- Timers: (2 x number of cards) - 1
- Timer Value Range: 100 ns to 5497 seconds
- Timer Resolution: 5 ns
- Timer Accuracy: +/- 10 ns + 0.01%
- Timer Reset Latency: 70 ns
- Glitch/Edge Recognizers: 2 per pod pair (timing only)
- Minimum Detectable Glitch: 1.5 ns
- Greater Than Duration: 6 ns to 100 ms in 6-ns increments
- Less Than Duration: 12 ns to 100 ms in 6-ns increments
- Data In to Trigger Out: 150 ns, typical

### Power Requirements

All necessary power is supplied by the backplane connector of the logic analysis system mainframe.

### Operating Environment Characteristics

- Indoor use only.
- Temperature
  - Instrument (except disk and media): 0 to 50 degrees C (+32 to 122 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 80% relative humidity at 40 degrees C (+104 degrees F)
- Altitude
  - Operating, to 4600 m (15,000 ft)
  - Non-operating, to 15,300 m (50,000 ft)
- Vibration
  - Operating: Random vibration 5-500 Hz, 10 minutes per axis, approximately 0.2 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, 0.50 g (0-peak), 5-minute resonant dwell at 4 resonances per axis.

Reliability is enhanced when operating within the following ranges:

- Temperature +20 to 35 degrees C (+68 to 95 degrees F)
- Humidity 20% to 80% non-condensing

### Storage

Store or ship the logic analyzer in environments with the following limits:

- Temperature -40 to +75 degrees C
- Humidity up to 90% at 65 degrees C
- Altitude up to 15,300 meters (50,000 feet)

Protect the module from temperature extremes which cause condensation on the instrument.

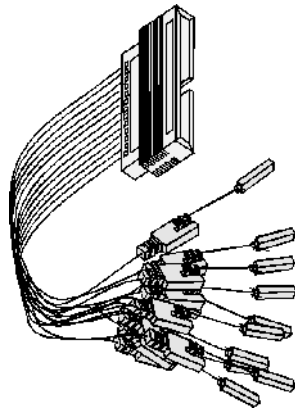


## Analyzer Probing Overview

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.

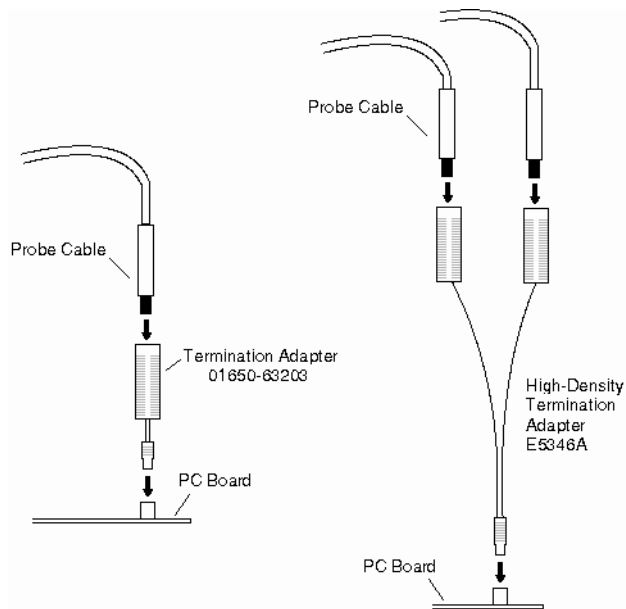
For equivalent circuit diagrams and pinouts, see the description of the probe type in the *Logic Analysis System and Measurement Modules Installation Guide*. If you have misplaced the *Logic Analysis System and Measurement Modules Installation Guide*, you can download the latest version from the Web at <URL: <http://www.hp.com/go/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 HP 5059-4356 surface mount grabbers and the HP 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 HP 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.

## The Symbols Tab

The Symbols tab offers control of the *symbols* capabilities. Symbols represent patterns and ranges of values found on labeled sets of bits. 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.

To load symbols, click one of the following:

- “To Load Object File Symbols” on page 102
- “To Load User-Defined Symbols” on page 114

Symbols are available for all state and timing analyzers. Each label listed in the Format menu can have its own group of symbols associated with it.

- “User-Defined Symbols” on page 113
- “Setting Up Object File Symbols” on page 102
- “Using Symbols In The Logic Analyzer” on page 115
- “Displaying Data in Symbolic Form” on page 101

## Displaying Data in Symbolic Form

You can display data in symbolic form in some of the display tools, such as the Listing display and the Waveform display.

### To View Symbolic Values in a Waveform Display

1. Right-click the label name where you want to display symbolic values.
2. Select *Change attributes...*
3. In the Attribute Dialog:
  - Set ShowValue to *On*.
  - Set Base to *Symbols* or *Line#*.
  - Click *OK*.

The symbolic names for the values now appear in the overlaid bus waveform.

### To View Symbolic Values in a Listing Display

1. Right-click the numeric base of the label where you want to display symbolic values.
2. Set the numeric base to *Symbols* or *Line#*.  
The symbolic names for the values now appear instead of numeric data.

## Setting Up Object File Symbols

Object file symbols can include variable names, procedure or function names, and source file names with line numbers. The linkage between symbol names and address or data values comes from one of two sources:

- Object files that are created by your compiler/linker.
- ASCII symbol files you create with a text editor.

### To use object file symbols

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 106).
3. Load the object file (see page 102) or ASCII symbol file into the logic analyzer.
4. If necessary, relocate sections of your code (see page 104).

### See Also

“Using Symbols In The Logic Analyzer” on page 115

“Symbol File Formats” on page 105

---

## To Load Object File Symbols

1. Select the *Symbol* tab and then the *Object File* tab.
2. 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.
3. Specify the directory to contain the symbol database file (*.ns*) in the field under, *Create Symbol File (.ns) in This Directory*. Click *Browse...* if you wish to find an existing directory name.

4. In the *Load This Object/Symbol File For Label* field, enter the object file name containing the symbols. Click *Browse...* to find the object file and click *Load* in the Browser dialog.  
If your logic analyzer is NFS mounted to a network, you can select object files from other servers.

### **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. Click the *Reload* button.

### **Value update**

The values of the object file symbols being used as terms or as SPA state-interval ranges will be updated automatically each time the object file symbols are reloaded.

### **Configuration file save**

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.

### **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 In The Logic Analyzer” on page 115

“Symbol File Formats” on page 105

---

## Relocating Sections of 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 Relocate a Single Section of Code

1. Click 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, click the field you wish to edit in the section list.
5. Type in 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. Click *Close*.

### To Relocate All Sections of Code

1. Click 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. Type the desired offset in the *Offset all sections by* field. The offset is



applied from the linked address or segment.

5. Click *Apply Offset*.
6. Click *Close*.

---

## To Delete Object File Symbol Files

1. Click 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. Click *Unload*.

---

## 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 111) file.

**See Also**

“Creating ASCII Symbol Files” on page 106 Creating ASCII Symbol Files

“Creating a readers.ini File” on page 111 Creating a readers.ini File

---

## Creating ASCII Symbol Files

If your language tool chain does not produce object files in a supported format, you can create an ASCII symbol file to define symbols. You can also use an ASCII symbol file to define symbols that are not included in your object file.

You can create an ASCII symbol file using any text editor that supports ASCII format text. Each entry in the file you create must be a string of ASCII characters consisting of a symbol name followed by an address or address range. The address or address range must be a hexadecimal number. It must appear on the same line of the text file 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
```

Two formats are available for creating ASCII symbol files:

“Simple Format” on page 106

“Record Header Format” on page 107

---

**NOTE:**

It is possible to generate ASCII symbol files from the symbol or load map output of most language tools.

---

### Simple Format

An ASCII symbol file can be a simple list of name/address pairs.

**Example**

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

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

## Record Header Format

An ASCII symbol file can be divided into records using key words, called *record headers*. The different records allow you to specify different kinds of symbols, with differing characteristics. An ASCII symbol file can contain any of the following kinds of records:

“Start Address” on page 108

“Sections” on page 108

“Functions” on page 108

“Variables” on page 110

“Source Line Numbers” on page 109

“Comments” on page 110

The record headers must be enclosed in square brackets, like this: [HEADER]. If no record header is specified, the lines following are assumed to be symbol definitions in one of the VARIABLES formats:

```
variable    address
variable    start..end
variable    start address    size
```

### Example

Here is an ASCII symbol 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
```

**Setting Up Object File Symbols**

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

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

**Start Address . Format**

```
[START ADDRESS]
address
```

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

**Example**

```
[START ADDRESS]
00001000
```

**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
```

**Sections .** Use SECTIONS to define symbols for regions of memory,

such as sections, segments, or classes.

### 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
```

---

#### NOTE:

If Section definitions are used in an ASCII symbol file, any subsequent Function or Variable definitions must fall within the address ranges of one of the defined Sections. Those Functions and Variables that do not will be ignored by the Symbol Utility.

---

**Source Line Numbers** . 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)

**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 give 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
```

**Comments .** Any text following a # character is ignored by the Symbol

Utility. The # can be used to comment a file. Comments can appear on a line by themselves, or on the same line, following a symbol entry.

### Format

```
#comment text
```

### Example

```
#This is a comment
```

---

## Creating a readers.ini File

You can change how an ELF/Dwarf or ELF/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 /hplgic/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 (Default Elf/Dwarf)  
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)
```

#### 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
```

## Setting Up Object File Symbols

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.

### Dwarf1NoType

1= Use the new fast symbol reader for Dwarf1 (Default)  
0= Use the previous version of the symbol reader

This symbol reader will be slow and may not be able to process all files. In addition the older symbol reader does not do C++ demangling. This symbol reader does process type information, so the sizes of local static and global variables will be processed.

Do not use this option unless you are having trouble reading the symbol file. This option will most likely be deleted in the future.

### Example

```
[ReadersElf]
C
C
ReadElfSection=1
Dwarf1NoType=1
MaxSymbolWidth=60
```



## User-Defined Symbols

“To Create User-Defined Symbols” on page 113

“To Replace User-Defined Symbols” on page 113

“To Delete User-Defined Symbols” on page 114

“To Load User-Defined Symbols” on page 114

---

## 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, type 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. Click *Add*.
8. Repeat steps 3 through 7 for additional symbols.
9. You can edit your list of symbols by using *Replace* (see page 113) and *Delete* (see page 114), if desired.

### See Also

“Using Symbols In The Logic Analyzer” on page 115

---

## 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.
-

## User-Defined Symbols

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

### See Also

“Using Symbols In The Logic Analyzer” on page 115

## Using Symbols In The Logic Analyzer

The ways symbols can be used in the logic analyzer are listed below:

- “Using Symbols As Trigger Terms” on page 115
- “Using Symbols as Search Patterns in Listing Displays” on page 116
- “Using Symbols as Trigger Terms in the Source Viewer” on page 116
- “Using Symbols as Pattern Filter Terms” on page 117
- “Using Symbols as Ranges in the Software Performance Analyzer” on page 117
- “Displaying Data in Symbolic Form” on page 101

---

## Using Symbols As Trigger Terms

You can use either one or both types of symbols as terms within your trigger sequence:

- *Object File Symbols.*
  - *User-Defined Symbols.*
1. At the bottom of the analyzer Trigger window, click the label button next to one of the resource terms, and select *Replace*.
  2. In the Resource selection dialog, select a label to be used in your trigger sequence.  
Use a label that has *symbols* loaded.
  3. Set the numeric base of the trigger term to *Symbols* or *Line #s*.
  4. Click the button to the right of the numeric base field.
  5. In the *Symbol Selector* (see page 117) dialog, select the symbol you want to use.

---

**NOTE:**

The values of object file symbols used as trigger terms are automatically updated when the object file symbols are reloaded (see page 102).

---

**See Also**

Setting Up a Trigger (see the *HP 16600A-Series 100 MHz State/250 MHz Timing Logic Analyzer* help volume)

---

## Using Symbols as Search Patterns in Listing Displays

1. Under the *Search* tab in the Listing display, click *Advanced searching*.
2. In the Goto Pattern dialog, click *Define*.
3. In the Search Pattern dialog, select the *Symbols* numeric base.
4. Select *Pattern*, *Range*, *Not Pattern*, or *Not Range*.
5. Click the button to the right of the numeric base field.
6. In the *Symbol Selector* (see page 117) dialog, select the symbol you want to use.

**See Also**

Go to an Exact Pattern. (see the *Listing Display Tool* help volume)

---

## Using Symbols as Trigger Terms in the Source Viewer

1. In the Source Viewer menu bar, click *Trace*, and select *Trace Setup*.
2. In the Source Line Trigger dialog, select *Symbols* or *Line #s* in the numeric base field.
3. Select *Pattern*, *Range*, *Not Pattern*, or *Not Range*.
4. Click the button to the right of the numeric base field.
5. In the *Symbol Selector* (see page 117) dialog, select the symbol you want to use.

**See Also**

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

## Using Symbols as Pattern Filter Terms

1. Click the numeric base field beside the selected filter term, and select *Symbols* or *Line #s*.
2. Select *Pattern*, *Range*, *Not Pattern*, or *Not Range*.
3. Select *Remove Matching Data* or *Pass Matching Data*, as desired.
4. Click the button to the right of the numeric base field.
5. In the *Symbol Selector* (see page 117) dialog, select the symbol you want to use.

---

## Using Symbols as Ranges in the Software Performance Analyzer

1. In the SPA tool, click *Symbols* in the Define Ranges dialog.
2. In the Symbol Selector (see page 118) dialog, select the symbol or group of symbols you want to use as ranges in your measurement.

### See Also

Defining State Interval Ranges. (see the *System Performance Analyzer* help volume)

## Using the Symbol Selector Dialog

1. 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 118) 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.
2. Click the symbol you want to use in the list of *Matching Symbols*.
3. If you are using object file symbols, you may need to:
  - Set *Offset By* (see page 119) to compensate for microprocessor

prefetches.

- Set Align to x Byte (see page 119) to trigger on odd-byte boundaries.
4. Select the Beginning, End, or Range of the symbol.
  5. Click *OK*.  
The name of your symbol now appears as the value of the resource term.
  6. Click *Cancel* to exit the *Symbol Selector* dialog without selecting a symbol.

## **Using the Symbol Selector Dialog**

1. In the *Symbol Selector* dialog, select the symbol you want to use. All of your symbols, regardless of type, will be available in the dialog.
  - Use the Search Pattern (see page 118) 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.
2. Click and drag to select the symbols you want to use in the list of *Matching Symbols*.
  - Click *Select All* to select all symbols in the list.
  - Click *Unselect All* to unselect all symbols in the list.
3. Click *Add Selected Symbols To Range List* to place the selected symbols into the *Current ranges* list in the Define Ranges dialog.
4. Click *Close* to exit the *Symbol Selector* dialog.

## **Search Pattern**

Use this field to locate particular symbols in the symbol databases. To use this field, type in 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 type "st\*".

### **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.

### **Offset By Option**

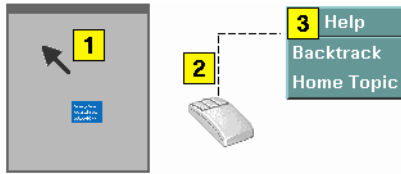
The Offset By option allows you to add an offset value to the starting point of the symbol that you want to use as a term. You might do this in order to trigger on a point in a function that is beyond the preamble of the function, or to trigger on a point that is past the prefetch depth of the processor. 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 trigger term. 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 trigger term. You would set the value of the "Offset By" field to 10 hex. Now, when you specify *func2* as your trigger term, the logic analyzer will trigger on address *func2*+10.

## Help - How to Navigate Quickly

1. Place mouse cursor anywhere in a help window.
2. Press the right mouse button.
3. Select desired destination.



You can also access all navigation and search commands from the help window menu bar.



---

## Help - System Overview

The help system is divided into *System* Help, and *Tool* Help. All help is designed to be task oriented and specific to the window where tasks are performed.

Links, in most cases, are confined to topics in the specific window where help was requested. However, some links do go up to system-level topics from specific tool windows. When this occurs, a second help window will appear. Since the definition of future tools is hard to link to, there are no links going from the system level down to specific tools.

### **System Help**

The system help is accessed through the *Help* field in the menu bar of the main system window. It offers help on system-level topics and operations.

### **Tool Help**

As you add new software and hardware tools to the system, the tool specific help is added. Tool specific help is accessed through the *Help* field in the menu bar of the specific tool windows.

### **Using Help**

In addition to system and tool help, there is help on help called *Using Help*. Using Help shows you how to navigate and search the help systems and to print help topics. Using Help is accessed through the *Help* field in all windows.

### **See Also**

Help - How to Navigate Quickly (see page 120)

## Run/Group Run Function

- Setting a tool for independent or Group Run (see page 123)
- Setting Single or Repetitive Run (see page 124)
- “Checking Run Status” on page 124

### **Understanding Run/Run All/Group Run**

The Run/Run All/Group Run buttons initiate data capture in the instrument tools you have configured. When an instrument tool is connected to analysis or display tools, any of the tools can initiate a run. When two or more instrument tools are configured, you can run them independently or as a group. Two or more instruments running as a group is called an Intermodule measurement.

Use the Intermodule Window (see the *HP 16600A/16700A Logic Analysis System* help volume) to coordinate the run function of multiple instruments as a "Group Run". A common "Group Run" configuration is to run the instrument tools at the same time. A more advanced measurement is to configure one instrument to arm another instrument, each with their own trigger conditions.

- Run appears in the setup dialog and icon menu of an instrument if it is not part of an Intermodule measurement.
- Group Run appears in the setup dialog and icon menu of each tool if two or more instruments are configured for an Intermodule measurement.
- Run All always appears in the System, Workspace and Run Status windows, and initiates a run in all configured instruments, whether they are run independently or are part of a Group Run.

Intermodule measurements are configured between individual instruments. Arming between two machines that belong to one analyzer is configured in the *Arming Info...* dialog found in the *Trigger* window of the analyzer.

### **Understanding Stop/Stop\_All/Cancel**

- Stop will terminate an individual instrument measurement that is running.

(perhaps waiting for a trigger condition)

- Stop All, when selected from the Workspace, will terminate running measurements from all instruments currently on the Workspace.
- Cancel will terminate the processing of trace data from an instrument to an analysis or display tool connected to its output.

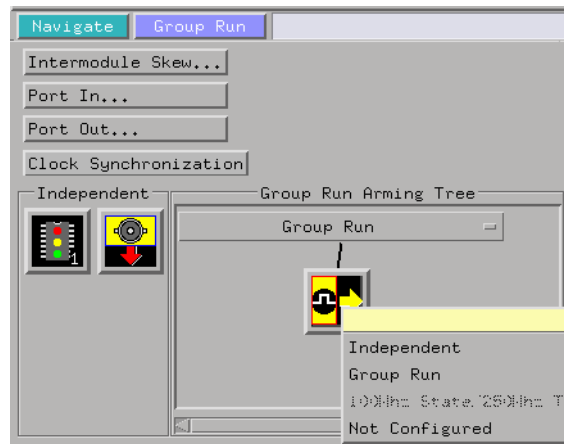
See Also “Demand Driven Data” on page 125

## Setting a tool for independent or Group Run

Use the Intermodule Window to change between Group Run and independent Run.

- Click the Intermodule icon in the System Window, OR
- Use Navigate->System->Intermodule

In the Intermodule window, move instruments between independent Run and Group Run by clicking the icon and selecting the desired arming source. All instruments in "Group Run" will run simultaneously.

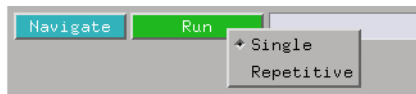


## Setting Single or Repetitive Run

A single measurement will stop after memory is full or a store qualification is met. A repetitive measurement executes successive Single measurements until Stop is selected.

When a single or repetitive measurement is stopped, only data that has been captured to that point is available for viewing.

Select single or repetitive by right-clicking on the Run button in the tool's setup window.



If you have problems displaying trace data when running Repetitive measurements, see “Demand Driven Data” on page 125.

---

## Checking Run Status

The *Run Status* dialog provides status information about the currently configured instruments, and the status of the run with respect to the trigger specification.

To access the *Run Status* dialog:

- The Run Status icon in the System Window, OR
- Navigate->System->Run Status



---

## Demand Driven Data

When an analyzer measurement occurs, acquisition memory is filled with data that is then transferred to the display memory of the analysis or display tools you are using, as needed by those tools. In normal use, this *demand driven data* approach saves time by not transferring unnecessary data.

Since acquisition memory is cleared at the beginning of a measurement, stopping a run may create a discrepancy between acquisition memory and the memory buffer of connected tools. Without a complete trace of acquisition memory, the display memory will appear to have 'holes' in it which appear as filtered data.

This situation will occur in these cases:

- If you stop a repetitive measurement after analyzer data has been cleared and before the measurement is complete.
- If a trigger is not found by the analyzer and the run must be stopped to regain control.

To make sure all of the data in a repetitive run is available for viewing:

- In the workspace, attach a Filter tool to the output of the analyzer.
- In the Filter, select "Pass Matching Data"
- In the filter terms, assure the default pattern of all "Don't Cares" (Xs).

**Run/Group Run Function**

This configuration will always transfer all data from acquisition memory. While this configuration will increase the time of each run, it will guarantee that repetitive run data is available regardless of when it is stopped.

---

## Glossary

**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 the target microprocessor. It provides an interface between the signals of the target microprocessor 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 HP 16600A-series or HP 16700A mainframe. 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** To click an item, position the cursor over the item. Then quickly press and release the *left mouse*

---

# Glossary

*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 HP 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



---

## Glossary

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. Click *Help* 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.

**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** 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** To drag and drop an item, 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.

**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,

---

## Glossary

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 HP 16600A-series or HP 16700A 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 HP 16715A, 16716A, and 16717A logic analyzers.

**held value** A value that is held until the next sample. A held value can exist in multiple data sets.

**immediate mode** In an oscilloscope, the trigger mode that does not require a specific trigger condition such as an edge or a pattern. Use immediate mode when the oscilloscope is armed by another instrument.

**interconnect cable** Short name for *module/probe interconnect cable*.

**intermodule** 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.

**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.

**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.

---

## Glossary

**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 HP 16600A-series or HP 16700A mainframe, 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 HP 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.

---

## Glossary

**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*.

**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.

**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** See *pod pair*

**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.

**point** To point to an item, move the mouse cursor 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 HP 16517A. The primary branch has an optional occurrence count field

---

## Glossary

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** 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.

---

# Glossary

- 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.

**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.

**time-correlated** Time correlated measurements are measurements involving more than one instrument in which all instruments have a common time or trigger reference.

**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.

**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.

**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.

**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.

**trace** See *acquisition*.

**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.

**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.

**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.



---

## Symbols

- \* , bit assignment, 53
- + , label polarity, 55
- , label polarity, 55
- . , bit unassignment, 53

## Numerics

- 167 MHz / 2M state mode, 44
- 167 MHz full channel state mode, 44
- 333 MHz full channel timing mode, 44
- 667 MHz half channel timing mode, 44

## A

- absolute tag, 91
- acquisition depth control, 43
- acquisition mode, 44
- acquisition modes, overview, 11
- acquisition modes, state, 49
- acquisition modes, timing, 49
- acquisitions, interpreting, 26
- acquisitions, processing, 26
- activity indicator, in Format, 53
- advanced clocking checkbox, 44
- advanced function, 85
- Align to x Byte option, 119
- Align to x Byte option for symbols, 119
- altitude characteristics, 95
- analyzer 2, turning on, 52
- analyzer name field, 47
- analyzer name, where used, 47
- analyzer probes, general-purpose, 97
- analyzer probes, termination adapter, 97
- analyzer, arming, 30
- analyzer, changing name, 47
- analyzer, off button, 48
- analyzer, on checkbox, 48

- arm out, specifying, 30
- arming analyzer, 30
- arming control, 30
- arrows, activity indicators, 51
- ASCII format, 106
- ASCII format symbols, 106, 107, 108, 109, 110
- ASCII symbol file, 109

## B

- bad data in measurement, 34
- base, numeric, 91
- basic state measurements, example, 18
- basic timing measurements, example, 22
- beyond prefetch depth, 119
- bit activity indicator, 53
- bit assignments, preserving, 55
- bit numbering within a label, 53
- bit order, changing, 56
- bit significance, 53
- bits, assigning, 53
- bits, reordering, 56
- branches taken, 87
- branches taken, replaced by, 87
- break down functions, 84
- browser dialog, 115
- browser, symbol search, 115
- browsing, 118
- browsing the symbol database, 118
- buses, naming, 57
- byte mode, numeric base, 91

## C

- cable activity indicators, 51
- cable power, probe, characteristic, 95
- cancel, 122
- capacitive loading, 33
- center trigger position, 49
- channel activity indicators, 51

- channel counts, characteristic, 95
- channel width, 49
- channels, assigning to label, 57
- channels, maximum per label, 53
- channel-to-channel skew, characteristic, 95
- clear menu, 74
- clear sequence levels, 74
- clear sequence/resource/name, 74
- clear settings, 74
- clock bits as data channels, 53
- clock bits warning message, 40
- clock channel specifiers, 44
- clock channels, inputs available as data, 53
- clock edges, adjusting, 59
- clock mode field, 44
- clock qualifiers, characteristic, 95
- clock setup area, 44
- clock threshold level note, 59
- clock threshold note, 57
- clock time, specification, 94
- clocks, data/address same line, 45
- clocks, master/slave/demultiplex, 45
- clocks, overview, 11
- code, assigning address offsets, 104
- combination terms, 91
- combo pick, 91
- comments, 110
- config tab, use, 10
- configuration files, loading, 32
- configurations, compatibility across models, 32
- configurations, storing, 32
- connection methods, 10
- connections, 10
- conventional timing acquisition modes, 49
- count state, 91
- count states timing function, 78
- count time, 91

counter 1 value checked as an event, but no increment action specified, 39  
counter 2 value checked as an event, but no increment action specified, 39  
counter warning message, 39  
create labels for busses, 57  
creating a file, 106  
creating ASCII symbol files, 106  
custom state function, 80  
custom timing macros, 76  
custom trigger macros, 72

## D

dashes, activity indicators, 51  
data channels, using clock bits, 53  
data displayed in symbolic form, 101  
data latching note, 45  
data on clocks display, 53  
data set, interpreting, 26  
data, stamping, 91  
default storing, 87  
defaulting the trigger, 74  
definition, calibration procedure, 94  
definition, characteristic, 93  
definition, function test, 94  
definition, operational accuracy calibration, 94  
definition, specification, 93  
deleting terms, 74  
demand driven data, 125  
demultiplex clock, 45  
demultiplex clocks, 58  
demultiplex clocks for pods, 57  
depth of memory, characteristic, 95  
disable/enable run status window, 124  
displaying symbols to represent data, 101

dontstoresample', 87

## E

edge and pattern function, 79  
edge events, 88  
edge trigger function, 79  
edit trigger sequence steps, 71  
ELF/DWARF file format, 105  
ELF/stabs file format, 105  
else branch, 71  
e-mail notify, 64  
email notify, 64  
end trigger position, 49  
enviromental characteristics, 95  
error messages, branch expression is too complex, 38  
error messages, goto action specifies an undefined level, 41  
error messages, maximum of 32 channels per label, 35  
error messages, measurement initialization error, 35  
error messages, no more edge/glitch resources, 37  
error messages, no more pattern resources available, 38  
error messages, slow or missing clock, 35  
error messages, trigger function initialization failure, 40  
error messages, trigger specification is too complex, 38  
error messages, waiting for trigger, 36  
errors in data, 34  
errors, error messages, 33  
events, 14  
example, 107, 118, 119

## F

file formats for ASCII symbols, 106  
file versions, 102  
files, 102  
files, loading user-defined symbol files, 114  
find early event state trigger functions, 83  
find edge function, 78  
find event state functions, 82  
find late event state trigger functions, 83  
find n times state functions, 82  
find pattern state functions, 82  
find sequence state triggers, 82  
finding the symbol you want, 118  
flag 1 - 8 events, 88  
flags, 88  
flowchart, 10  
format tab, use, 13, 51  
full channel, timing, 49  
functions, 108

## G

glossary of terms, 2  
group bit assignment, 53  
group run, 122

## H

half channel timing mode, 58  
half channel, timing, 49  
help, 120, 121  
help, how to use, 121  
help, symbols, 100  
help, trigger, 30  
help, trigger sequence, 75  
how to set up, 10  
HP 16715A 167 MHz State/667 MHz Timing Logic Analyzer, 2  
HP 16715A characteristics, 95  
HP 16715A specifications, 94  
humidity characteristics, 95

---

## I

id, naming analyzer, 47  
IEEE-695 file format, 105  
if branch, 71  
in ASCII format, 108, 109, 110  
in range events, 88  
in symbol browser, 118  
independent run, 122  
input capacitance, probe,  
    characteristic, 95  
input resistance, probe,  
    characteristic, 95  
internal sequence, 75  
internal sequence levels, 85  
invalid data note, 58

## L

label polarity, changing, 55  
labels, 14  
labels, activating, 55  
labels, add after, 54  
labels, add before, 54  
labels, adding, 54  
labels, assigning bits, 53  
labels, defining, 57  
labels, deleting, 54  
labels, inserting, 54  
labels, polarity, 55  
labels, reordering bits, 56  
labels, turning off, 55  
labels, turning on, 55  
least significant bit in label, 53  
line numbers, 109  
load, reducing, 33  
loading, 102  
loading files including symbols, 102  
loading object file symbols, 102  
loading user-defined symbol files,  
    114  
logic analyzer hangs, 34  
logic analyzer probes, 10, 97  
logic analyzer triggers, 65

logic analyzer, testing, 42

## M

machines available, characteristic,  
    95  
mail on trigger, 64  
main system help page, 2  
masking off addresses of symbols,  
    119  
master clock, 45  
master clocks for pods, 57  
maximizing memory, 87  
maximum pulse width trigger, 79  
maximum state speed,  
    specification, 94  
measurement doesn't run, 34  
measurement setup, 11  
measurement, probing options, 97  
measurement, run options, 122  
measurements, overview of basic  
    state, 18  
measurements, overview of timing,  
    22  
memory and trigger, 49  
memory depth, characteristic, 95  
memory depth, setting, 43  
message, trigger inhibited during  
    timing prestore, 36  
minimum master-to-master clock  
    time, 94  
minimum pulse width trigger, 79  
mode and acquisition depth, 44  
mode and channel width, 44  
mode and sample rate, 44  
mode, clocking, 45  
most significant bit in label, 53

## N

name, id for saving setup, 47  
negative logic, note, 55  
no more pattern resources  
    message, 38

not in range events, 88  
note, activity indicators, 51  
note, clock, characteristic, 95  
note, clocking, 45  
note, negative logic, 55  
note, reorder bits, 56  
note, state memory, 49

## O

object file symbol browser, 115  
object file symbol files, 102  
object file symbols, 115, 118  
occurrence counter, characteristic,  
    95  
occurrence counter, state trigger,  
    87  
occurrence counter, timing trigger,  
    86  
occurs field, 87  
occurs trigger field, 86  
odd-numbered addresses, 119  
odd-numbered addresses  
    represented by symbols, 119  
offset, 119  
offset addresses, assigning, 104  
Offset By option of the symbol  
    browser, 119  
OMF96 file format, 105  
OMFx86 file format, 105  
on/off switch, 48  
operating environment  
    characteristics, 95  
options, run, 122  
overview, measurement process,  
    10

## P

pattern after edge function, 79  
pattern count, state trigger, 87  
pattern duration function, 78  
pattern duration, timing, 86  
Pattern field, 118

- 
- pattern, state trigger functions, 82
  - pattern/edge trigger functions, 79
  - performance verification, 42
  - period, sample time, 48
  - pod assignment dialog, 52
  - pod clocking, demultiplex, 57
  - pod thresholds, setting, 59
  - pods, assigning, 52
  - pods, clocking, 45
  - pods, selecting, 58
  - pods, specifying state clock, 57
  - power through pod cables, characteristic, 95
  - predefined macros, 76, 85
  - predefined trigger functions, modifying, 84
  - prefetch, 119
  - present for &> duration trigger field, 86
  - probe leads, 97
  - probes, individual signal, 97
  - probing options, 10
  - probing tips, 10
  - probing, overview, 97
  - problems making measurements, 33
  - pulse width function, 79
  - Q**
  - qualifiers, clocks, characteristic, 95
  - R**
  - R, bit assignment, 53, 56
  - radix, numeric base, 91
  - rate, sample period, 48
  - readers.ini file, 111
  - recalling trigger sequences, 72
  - record headers, 107
  - relative tag, 91
  - relocating sections of code, 104
  - repetitive, 122
  - repetitive data display, 125
  - restore functions, 84
  - results, 118
  - roadmap, 10
  - run, 122
  - run options, 124
  - run status, checking, 124
  - run status, disable window, 124
  - S**
  - sample period control, 48
  - sample period, characteristic, 95
  - sample rate, setting, 44
  - sampling tab, use, 43
  - sampling tab, uses, 11
  - saving trigger sequences, 72
  - Scroll Files field, 115
  - Search Pattern field, 118
  - searching the symbol database, 118
  - sections, 108
  - selecting macros, 69
  - self test, 42
  - sequence levels, characteristic, 95
  - sequencer, maximum levels, characteristic, 95
  - set up, trigger sequence, 71
  - setting threshold, 59
  - settings, saving, 32
  - setup/hold field, 59
  - setup/hold time, specification, 94
  - shortcut, bit assignment, 53
  - simple example, 106
  - simple form, 106
  - single, 122
  - skew, channel-to-channel, characteristic, 95
  - slave clock, 45
  - slave clocks for pods, 57
  - slow clock message, 35
  - SMTP, 65
  - source line numbers, 109
  - specifications and characteristics, 93
  - speed, state/timing, characteristic, 95
  - stamp data, 91
  - start address, 108
  - start trigger position, 49
  - state channel width, 49
  - state clocks, 44
  - state clocks, characteristic, 95
  - state clocks, master/slave/both, 57
  - state clocks, setup/hold, 59
  - state count, memory, 49
  - state measurements, basic overview, 18
  - state memory, 49
  - state modes, 49
  - state modes, characteristic, 95
  - state tags, 91
  - state trigger functions, 76, 82, 83
  - status, 124
  - stop, 122
  - store qualification, 87
  - store sample, 87
  - storing trigger setups, 72
  - symbol demangling, 111
  - symbol file formats, 105
  - symbol file versions, 102
  - symbol selector dialog, 117, 118
  - symbol type, 115
  - symbol types, 115
  - symbols, 118
  - symbols, displaying to represent data, 101
  - symbols, how they are used in the logic analyzer, 115
  - symbols, loading object file symbols, 102
  - symbols, loading user-defined symbol files, 114
  - symbols, setting up, 102
  - symbols, types and use, 100
  - symbols, user-defined details, 113

## T

tab, symbols, 100  
tag data, 91  
temperature characteristics, 95  
then branch, 71  
threshold accuracy, specification, 94  
threshold logic levels, ECL, 59  
threshold logic levels, TTL, 59  
threshold range, probe, characteristic, 95  
TI COFF file format, 105  
time count, 49  
time count, characteristic, 95  
time interval accuracy, characteristic, 95  
time tag resolution, characteristic, 95  
time tags, 91  
time violation functions, 79  
timer 1, 90  
timer 1 value checked as an event, but no start action specified, 39  
timer 2, 90  
timer 2 value checked as an event, but no start action specified, 39  
timer warning message, 39  
timers, using, 90  
timing analysis characteristics, 95  
timing measurements, basic overview, 22  
timing modes, 49  
timing speed, characteristic, 95  
timing trigger functions, 76  
timing, memory depth, 49  
tip capacitance, probe, characteristic, 95  
trace buffer, 58  
trigger characteristics, 95  
trigger events, 88  
trigger flags, uses, 87

trigger function libraries, 73  
trigger functions, 75  
trigger functions, advanced, 85  
trigger functions, branching, 71  
trigger functions, creating, 85  
trigger functions, displaying parts, 84  
trigger functions, modifying, 85  
trigger functions, pattern/edge, 79  
trigger functions, predefined, 76  
trigger functions, state, 80, 82, 83  
trigger functions, time violation, 79  
trigger functions, timing, 76, 78  
trigger functions, using, 68  
trigger inhibited informational message, 36  
trigger macros, selecting, 69  
trigger pattern functions, 78  
trigger position control, 49  
trigger resource terms, combination, 91  
trigger sequence branches, 71  
trigger sequence levels, editing, 71  
trigger sequence levels, goto, 71  
trigger sequence steps, copying, 69  
trigger sequence steps, replacing, 69  
trigger sequence, adding steps, 69  
trigger sequence, clearing, 74  
trigger sequence, customizing, 85  
trigger sequence, default, 74  
trigger sequence, deleting, 69  
trigger sequence, editing, 69  
trigger sequence, explanation, 75  
trigger sequence, inserting, 69  
trigger sequence, specifying, 68  
trigger sequences, loading, 72  
trigger sequences, saving, 72  
trigger sequences, storing, 72  
trigger set up, 71  
trigger tab, reference, 62, 64  
trigger tab, use, 14  
trigger term, 115, 119

trigger terms (events), 88  
trigger terms, combination, 91  
trigger variables, 88  
trigger, pattern durations, 86  
trigger, poststore, 49  
trigger, resetting, 74  
trigger, setting up, 68  
trigger, substeps, 14  
triggering beyond, 119  
triggering on a symbol, 115, 118, 119  
triggering, about, 65  
troubleshooting the logic analyzer, 33

## U

unassigned bits, 53  
user macros, trigger, 72  
User Symbols, 115  
user threshold logic level, 59  
user-defined macros, 85  
user-defined symbol files, loading, 114  
user-defined symbols, details, 113  
user-defined trigger, 85  
user-level trigger functions, 71

## V

variables, 110  
versions, 102  
versions of symbol files, 102  
vibration characteristics, 95

## W

wait for arm in, 40  
wait for arm in trigger function, 30  
wait for other machine to trigger, 40  
wait for second analyzer to trigger function, 30  
wait state functions, 83  
wait time function, 79

---

warning messages, clock bits, 40  
warning messages, counter not  
    incremented, 39  
warning messages, no trigger  
    action found in the trace  
    specification, 41  
warning messages, timer never  
    started, 39  
warranty, what is covered, 93  
what is triggering, 65  
wildcard characters, 118