Help Volume

© 1997-2001 Agilent Technologies. All rights reserved.

# Analysis: Pattern Filter Tool

# Using the Pattern Filter Tool

Use the Pattern Filter tool to filter a data stream before you view it or store it to a file. You can specify which states in the data you want to pass through the filter, or which states you want to filter out.

- “Filtering States Out of the Data” on page 6
- “Passing Only Desired States Through the Filter” on page 7
- “Passing All States Through the Filter” on page 8
- “Loading & Saving Filter Configurations” on page 17
- “Printing the Pattern Filter Window” on page 18

**See Also**

“Working With Filter Terms” on page 9

The Symbols Tab (see page 19)

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

Glossary of Terms (see page 41)

# Contents

**Using the Pattern Filter Tool**

**1 Using the Pattern Filter Tool**

Filtering States Out of the Data 6

Passing Only Desired States Through the Filter 7

Passing All States Through the Filter 8

Working With Filter Terms 9
Adding and Deleting Filter Terms 9
Defining and Changing Filter Term Values 10
Adding Labels to a Filter Term 11
Deleting Labels from a Filter Term 12
Replacing Labels in a Filter Term 13
Changing Term Names 14
Turning Filter Terms On and Off 15
Creating Powerful Filter Expressions 15

Loading & Saving Filter Configurations 17

Printing the Pattern Filter Window 18

The Symbols Tab 19

Displaying Data in Symbolic Form 20

Setting Up Object File Symbols 21
To Load Object File Symbols 21
Relocating Sections of Code 23
To Delete Object File Symbol Files 24
Symbol File Formats 24
Creating ASCII Symbol Files 25
Creating a readers.ini File 30

Using Symbols In The Logic Analyzer 33

Using Symbols As Trigger Terms 33
Using Symbols as Search Patterns in Listing Displays 34
Using Symbols as Trigger Terms in the Source Viewer 34
Using Symbols as Pattern Filter Terms 34
Using Symbols as Ranges in the Software Performance Analyzer 35

User-Defined Symbols 38

To Create User-Defined Symbols 38
To Replace User-Defined Symbols 38
To Delete User-Defined Symbols 39
To Load User-Defined Symbols 39

**Glossary**

**Index**

1

# Using the Pattern Filter Tool

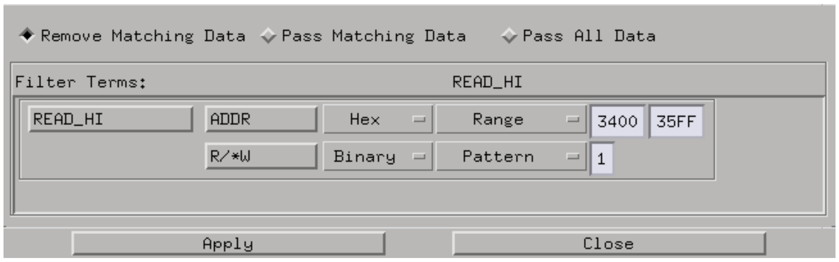
## Filtering States Out of the Data

1. Select *Remove Matching Data*.
2. In the *Filter Terms* area, define a filter term (see page 10) to match the condition that you want to filter on.
   All of the labels in a filter term are logically *AND'ed* together.
3. If necessary, add more terms (see page 9) to the expression.
   At the top of the Pattern Filter window is a *filter expression* that shows each of the terms that are logically *OR'ed* together.
4. Select the *Run* icon to acquire and filter data, or if data has already been acquired, select *Apply* to filter it.

All states in the data that match the filter expression are filtered out of the data.

**Example**

This setup filters out states that contain a *READ* from a particular range of memory addresses. This condition is indicated by a value of 1 for label *R/*W* and a memory address in the range 3400-35FF hex. The filter term *READ_HI* is defined to match this condition.


[✓ Remove Matching Data] Pass Matching Data Pass All Data

Filter Terms: READ_HI

| | | | | | |
|---|---|---|---|---|---|
| READ_HI | ADDR | Hex | Range | 3400 | 35FF |
| | R/*W | Binary | Pattern | 1 | |

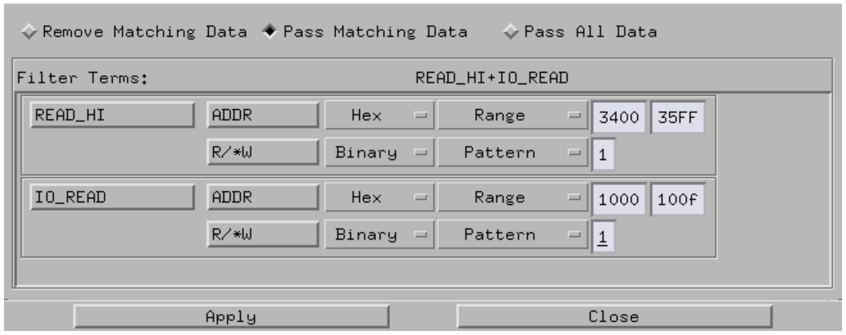
## Passing Only Desired States Through the Filter

1. Select *Pass Matching Data*.
2. In the *Filter Terms* area, define a filter term (see page 10) to match the condition you want to filter on.
   All of the labels in a filter term are logically *AND'ed* together.
3. If necessary, add more terms (see page 9) to the expression.
   At the top of the Pattern Filter window is a *filter expression* that shows each of the terms that are logically *OR'ed* together.
4. Select the *Run* icon to acquire and filter data, or if data has already been acquired, Select *Apply* to filter it.

Only those states in the data stream that match the filter expression are passed through. The rest are filtered out.

**Example**

This setup passes through only those states that contain a *READ* from one of two memory address ranges, 3400-35FF hex, or 1000-100F hex. A *READ* is indicated as pattern *1* on label *R/*W*. The filter term *READ_HI* is set up to match a read from one of the address ranges. The filter term *IO_READ* is set up to match a read from the other.


Remove Matching Data [✓ Pass Matching Data] Pass All Data

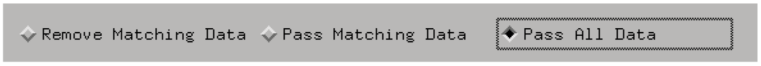
Filter Terms: READ_HI+IO_READ

| | | | | | |
|---|---|---|---|---|---|
| READ_HI | ADDR | Hex | Range | 3400 | 35FF |
| | R/*W | Binary | Pattern | 1 | |
| IO_READ | ADDR | Hex | Range | 1000 | 100f |
| | R/*W | Binary | Pattern | 1 | |

## Passing All States Through the Filter

1. Select *Pass All Data*.
2. Select *Run* to acquire and filter data, or if data has already been acquired, select *Apply* to filter it.

You can also turn off some or all of the terms (see page 15) in order to pass all of the data.



## Working With Filter Terms

Filter terms are used to define a filter expression that determines how data is filtered. The filter terms are logically *OR'ed* together to create the filter expression, which is shown at the top of the Pattern Filter window.

Each filter term is made up of data patterns and ranges for one or more labels. All of the labels associated with a particular filter term are logically *AND'ed* together to create the filter term.

- "Adding and Deleting Filter Terms" on page 9
- "Defining and Changing Filter Term Values" on page 10
- "Adding Labels to a Filter Term" on page 11
- "Deleting Labels from a Filter Term" on page 12
- "Replacing Labels in a Filter Term" on page 13
- "Changing Term Names" on page 14
- "Turning Filter Terms On and Off" on page 15
- "Creating Powerful Filter Expressions" on page 15

## Adding and Deleting Filter Terms

**To Add a Filter Term:**

1. Select one of the terms in the *Filter Terms* area.
2. Select *Insert term before* or *Insert term after*.

**To Delete a Filter Term:**

1. Select the term that you want to delete.
2. Select *Delete term*.

**To Delete All Terms:**

1. Select any of the terms in the *Filter Terms* area.
2. Select *Delete all terms*.
   One term will remain. The pattern for this term will be all *Don't Cares*, represented by *X's*.

**See Also**

"Defining and Changing Filter Term Values" on page 10

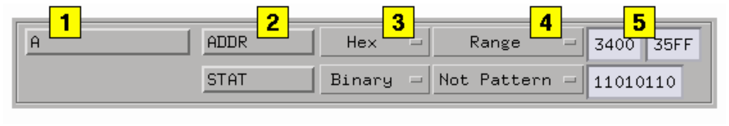
"Adding Labels to a Filter Term" on page 11

"Deleting Labels from a Filter Term" on page 12

## Defining and Changing Filter Term Values

The value of a filter term consists of the logical *AND* of each of the label patterns or ranges associated with the term. To change the value of the filter term:

1. If desired, change the term name (see page 14).
2. Add the desired labels (see page 11) to the term.
3. For each label, set the numeric base.
   If you choose *Symbol* base, see The Symbols Tab (see page 19).
4. Select a pattern or range mode (see page 11).
5. Type the pattern or range value.



**See Also**

"Adding and Deleting Filter Terms" on page 9

"Adding Labels to a Filter Term" on page 11

"Deleting Labels from a Filter Term" on page 12

### Pattern and Range Modes

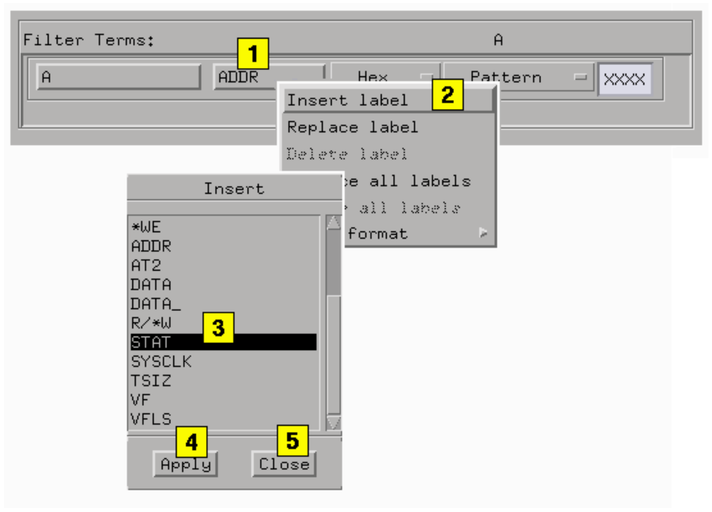
Four modes are available for each label assigned to a filter term: *Pattern*, *Range*, *Not Pattern* and *Not Range*.

*Not Pattern* selects all patterns except the defined pattern. For example, if the defined pattern is *11* binary and the mode is set to *Not Pattern*, this is the same as defining three filter terms: one for *00*, one for *01*, and one for *10*, and setting them all to *Pattern*. (Filter terms are ORed together.)

*Not Range* selects the values outside the range you define. If the range for a four-bit label is *0 through 8* hex, and the mode is set to *Not Range*, this is the same as setting the mode to *Range* and range value as *9 through F* hex.

## Adding Labels to a Filter Term

1. Select one of the labels that is already assigned to the term.
   If no labels are present, make sure that the filter is connected to an instrument tool, and that there are labels defined in the instrument tool windows.
2. Select *Insert label*.
3. In the *Label Selection* dialog, select the label name to be inserted.
   To select multiple labels, highlight the desired labels.
4. Select *Apply*.
5. Select *Close* to close the *Label Selection* dialog.



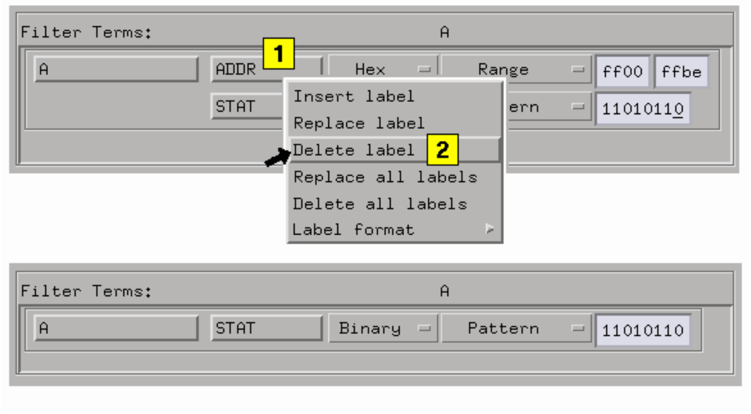
**See Also**

"Adding and Deleting Filter Terms" on page 9

"Deleting Labels from a Filter Term" on page 12

## Deleting Labels from a Filter Term

1. Choose the label name you want to delete.
2. Select *Delete label*.
   The Filter Terms dialog will show your deleted label gone.
3. To delete all of the labels from a term, select *Delete all labels*.
   At least one label name will remain, with a pattern of all *Don't Cares*, represented by *X's*.



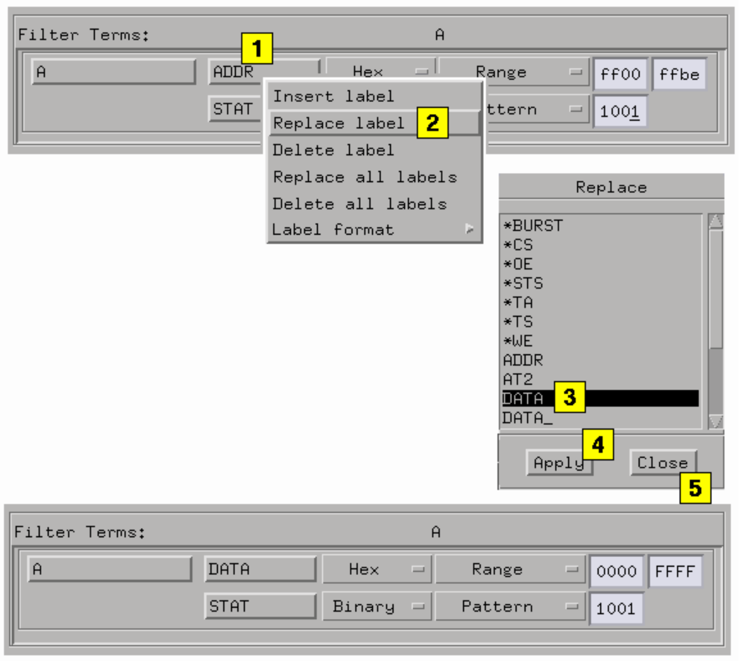
**See Also**

"Adding and Deleting Filter Terms" on page 9

"Adding Labels to a Filter Term" on page 11

## Replacing Labels in a Filter Term

1. Choose the label name you want to replace.
2. Select *Replace label*.
3. In the *Label Selection* dialog, select the name of the replacement label.
4. Select *Apply*.
5. Select *Close* to close the *Label Selection* dialog
   Your replacement label will have default values, which you can change.



**See Also**

"Adding and Deleting Filter Terms" on page 9

"Adding Labels to a Filter Term" on page 11

"Deleting Labels from a Filter Term" on page 12

## Changing Term Names

1. Choose the term whose name you want to change.
2. Select *Rename term*.
3. In the Rename Term dialog, enter in the new name.
4. Select *OK*.

**See Also**

"Adding and Deleting Filter Terms" on page 9

## Turning Filter Terms On and Off

You can remove a filter term from the filter expression by turning it off. This will preserve the term definition, but will remove the term from the set of *OR'ed* terms that are used to filter the data.

1. Select the term you want to turn on or off.
2. Select *Term [ON]/[OFF]* to toggle the control.
   Terms that are turned off do not appear in the filter expression at the top of the Filter Terms window.

**NOTE:** Terms that are off are grayed-out. None of the labels or values can be changed until the term is turned on again.

**See Also**

"Adding and Deleting Filter Terms" on page 9

"Adding Labels to a Filter Term" on page 11

"Deleting Labels from a Filter Term" on page 12

## Creating Powerful Filter Expressions

The default filter expression ORs all filter terms together. You can override the default to create more complex combinations of filter terms when you need powerful filter expressions.

**NOTE:** Expression filtering is not available and will be greyed out when Timing Zoom is turned on. In order for expression filtering to work properly, you MUST turn Timing Zoom off and configure the Pattern Filter tool to a single data set input.

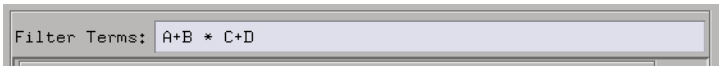
1. In the menu bar, select *Options*, then set *Expression* to *On*.
   The Filter Terms field becomes a text entry field. See the Note above.
2. Create any desired expression composed of the defined filter terms.
3. Available operators are:

   *=AND

+=OR

()=grouping filter terms

**Example:**



Filter Terms: A+B * C+D

## Loading & Saving Filter Configurations

Pattern Filter settings can be saved to a configuration file on hard disk or floppy disk, and settings can be loaded from previously saved configuration files.

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

**NOTE:** The *Load Configuration* window can be accessed via File->Load Configuration.
The *Save Configuration* window can be accessed via File->Save Configuration.

## Printing the Pattern Filter Window

The print windows operation enables you to print just the Pattern Filter tool window. Use this operation if you want a hardcopy or electronic record of configurations and data currently displayed in the viewing area of the Pattern Filter window.

**NOTE:** Only the currently displayed viewing area of the Pattern Filter window is printed. If any data or configuration fields appear offscreen, scroll the desired data or configuration fields into the window's viewing area before printing.

1. Optional - configure the Print Options (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume) if desired. Print Options include print destination, file format type, filename autoincrement, and color/b&w; pixel mapping.
2. In the Pattern Filter tool menu bar, select *File*, then select *Print This Window*. The print output will be as configured in the Print Options in step 1.

**See Also**

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

## 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, see:

- "To Load Object File Symbols" on page 21
- "To Load User-Defined Symbols" on page 39

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 38
- "Setting Up Object File Symbols" on page 21
- "Using Symbols In The Logic Analyzer" on page 33
- "Displaying Data in Symbolic Form" on page 20

## 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. Select the label name where you want to display symbolic values.
2. Choose *Properties...*.
3. In the Properties dialog:
   - Set ShowValue to *On*.
   - Set Base to *Symbols* or *Line #s*.
   - Select the *OK* button.

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

### To View Symbolic Values in a Listing Display

1. Select the numeric base of the label where you want to display symbolic values.
2. Set the numeric base to *Symbols* or *Line #s*.
   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 25).
3. Load the object file (see page 21) or ASCII symbol file into the logic analyzer.
4. If necessary, relocate sections of your code (see page 23).

**See Also**

"Using Symbols In The Logic Analyzer" on page 33

"Symbol File Formats" on page 24

## 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*. Select the *Browse...* button 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. Select the *Browse...* button to find the object file and select the *Load* button 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. Select 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 33

"Symbol File Formats" on page 24

## 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. Select the *Symbol* tab and then the *Object File* subtab.
2. Select the *Relocate Sections...* button.
3. In the Address column, select the address you wish to relocate.
4. In the *Edit selected section* field, enter the new address.
5. Select *Apply Edit*.
6. Repeat steps 3 through 5 above for any other sections to be relocated.
7. Select the *Close* button.

### To Relocate All Sections of Code

1. Select the *Symbol* tab and then the *Object File* subtab.
2. Select the *Relocate Sections...* button.
3. Enter the value you wish to use in the *Offset all selections by* field.
4. Select the *Apply Offset* button.
5. Select the *Close* button.

## To Delete Object File Symbol Files

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

## Symbol File Formats

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

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

For 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 30) file.

**See Also**

"Creating ASCII Symbol Files" on page 25Creating ASCII Symbol Files

"Creating a readers.ini File" on page 30Creating 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 25

"Record Header Format" on page 25

**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 27

"Sections" on page 27

"Functions" on page 27

"Variables" on page 29

"Source Line Numbers" on page 28

"Comments" on page 29

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

[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/Stabs, Ticoff or Coff/Stabs symbol file is processed by creating a reader.ini file.

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

**Reader options**

### C++Demangle

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

### C++DemOptions

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

### MaxSymbolWidth

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

### OutSectionSymbolValid

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

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

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

### ReadElfSection

```
2= Process all globals from ELF section (Default)
   Get size information of local variables
```

```
1= Get size information of global and local variables
   Symbols for functions will not be read, and
   only supplemental information for those symbols in
   the Dwarf or stabs section will be read.
0= Do not read the Elf Section
```

If a file only has an ELF section this will have no effect and the ELF section will be read completely. This can occur if the file was created without a "generate debugger information" flag (usually -g). Using the -g will create a Dwarf or Stabs debug section in addition to the ELF section.

### StabsType

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

### ReadOnlyTicoffPage

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

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

### AppendTicoffPage

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

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

## Examples

### Example for Elf/Stabs

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

### Example for Coff/Stabs (using Ticoff reader)

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

### Example for Ticoff

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

## 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 33
- "Using Symbols as Search Patterns in Listing Displays" on page 34
- "Using Symbols as Trigger Terms in the Source Viewer" on page 34
- "Using Symbols as Pattern Filter Terms" on page 34
- "Using Symbols as Ranges in the Software Performance Analyzer" on page 35
- "Displaying Data in Symbolic Form" on page 20

## 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, select the label button next to one of the resource terms, and choose *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. Select the button to the right of the numeric base field.
5. In the *Symbol Selector* (see page 35) 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 21).

## Using Symbols as Search Patterns in Listing Displays

1. Under the *Search* tab in the Listing display, select the *Advanced searching* button.
2. In the Goto Pattern dialog, select the *Define* button.
3. In the Search Pattern dialog, select the *Symbols* numeric base.
4. Select *Pattern*, *Range*, *Not Pattern*, or *Not Range*.
5. Select the button to the right of the numeric base field.
6. In the *Symbol Selector* (see page 35) 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, select *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. Select the button to the right of the numeric base field.
5. In the *Symbol Selector* (see page 35) 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. Select 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. Select the *Absolute XXXX* button.
5. In the *Symbol Selector* (see page 35) dialog, select the symbol you want to use.

## Using Symbols as Ranges in the Software Performance Analyzer

1. In the state interval SPA tool, select the *Symbols* button in the Define Ranges dialog.
2. In the Symbol Selector (see page 36) 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 36) 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. Select 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 37) to compensate for microprocessor prefetches.
   - Set Align to x Byte (see page 37) to trigger on odd-byte boundaries.

4. Select the Beginning, End, or Range of the symbol.
5. Select the *OK* button.
   The name of your symbol now appears as the value of the resource term.
6. Select the *Cancel* button 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 36) 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. Drag to select the symbols you want to use in the list of *Matching Symbols*.
   - Select the *Select All* button to select all symbols in the list.
   - Select the *Unselect All* button to unselect all symbols in the list.
3. Select the *Add Selected Symbols To Range List* button to place the selected symbols into the *Current ranges* list in the Define Ranges dialog.
4. Select the *Close* button to exit the *Symbol Selector* dialog.

## Search Pattern

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

### Asterisk wildcard (*)

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

### 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*.

## User-Defined Symbols

"To Create User-Defined Symbols" on page 38

"To Replace User-Defined Symbols" on page 38

"To Delete User-Defined Symbols" on page 39

"To Load User-Defined Symbols" on page 39

## To Create User-Defined Symbols

1. Under the *Symbol* tab, select the *User Defined* tab.
2. Select the label name you want to define symbols for.
3. At the bottom of the *User Defined* tab, enter a symbol name in the entry field.
4. Select a numeric base.
5. Select *Pattern* or *Range* type for the symbol.
6. Enter values for the pattern or range the symbol will represent.
7. Select the *Add* button.
8. Repeat steps 3 through 7 for additional symbols.
9. You can edit your list of symbols by using Replace (see page 38) and Delete (see page 39), if desired.

**See Also** "Using Symbols In The Logic Analyzer" on page 33

## To Replace User-Defined Symbols

1. Under the *Symbol* tab, select the *User Defined* tab.
2. Select the label you want to replace symbols for.

3. Select the symbol to replace.
4. At the bottom of the *User Defined* tab, modify the symbol name, numberic base, Pattern/Range type, and value, as desired.
5. Select the *Replace* button.
6. Repeat steps 3 through 5 to replace other symbols, if desired.

## To Delete User-Defined Symbols

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

## To Load User-Defined Symbols

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

1. In the menu bar of your analyzer window, select *File* and then *Load Configuration....*
2. In the Load Configuration dialog, select the directory and filename to be loaded.
3. Select the target of the load operation.
4. Select the *Load* button.
   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 33

# 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 a microprocessor or standard bus in the device under test. An analysis probe provides an interface between the signals of the microprocessor or standard bus and the inputs of the logic analyzer. Also called a *preprocessor*.

**analyzer 1** In a logic analyzer with two *machines*, refers to the machine that is on by default. The default name is *Analyzer<N>*, where N is the slot letter.

**analyzer 2** In a logic analyzer with two *machines*, refers to the machine that is off by default. The default name is *Analyzer<N2>*, where N is the slot letter.

**arming** An instrument tool must be armed before it can search for its trigger condition. Typically, instruments are armed immediately when *Run* or *Group Run* is selected. You can set up one instrument to arm another using the *Intermodule Window*. In these setups, the second instrument cannot search for its trigger condition until it receives the arming signal from the first instrument. In some analyzer instruments, you can set up one analyzer *machine* to arm the other analyzer machine in the *Trigger Window*.

**asterisk (*)** See *edge terms*, *glitch*, and *labels*.

**bits** Bits represent the physical logic analyzer channels. A bit is a *channel* that has or can be assigned to a *label*. A bit is also a position in a label.

**card** This refers to a single instrument intended for use in the Agilent Technologies 16600A-series or 16700A/B-series mainframes. One card fills one slot in the mainframe. A module may comprise a single card or multiple cards cabled together.

**channel** The entire signal path from the probe tip, through the cable and module, up to the label grouping.

**click** When using a mouse as the

pointing device, to click an item, position the cursor over the item. Then quickly press and release the *left mouse button*.

**clock channel** A logic analyzer *channel* that can be used to carry the clock signal. When it is not needed for clock signals, it can be used as a *data channel*, except in the Agilent Technologies 16517A.

**context record** A context record is a small segment of analyzer memory that stores an event of interest along with the states that immediately preceded it and the states that immediately followed it.

**context store** If your analyzer can perform context store measurements, you will see a button labeled *Context Store* under the Trigger tab. Typical context store measurements are used to capture writes to a variable or calls to a subroutine, along with the activity preceding and following the events. A context store measurement divides analyzer memory into a series of context records. If you have a 64K analyzer memory and select a 16-state context, the analyzer memory is divided into 4K 16-state context records. If you have a 64K analyzer memory and select a 64-state context, the analyzer memory will be divided into 1K 64-state records.

**count** The count function records periods of time or numbers of state transactions between states stored in memory. You can set up the analyzer count function to count occurrences of a selected event during the trace, such as counting how many times a variable is read between each of the writes to the variable. The analyzer can also be set up to count elapsed time, such as counting the time spent executing within a particular function during a run of your target program.

**cross triggering** Using intermodule capabilities to have measurement modules trigger each other. For example, you can have an external instrument arm a logic analyzer, which subsequently triggers an oscilloscope when it finds the trigger state.

**data channel** A *channel* that carries data. Data channels cannot be used to clock logic analyzers.

**data field** A data field in the pattern generator is the data value associated with a single label within a particular data vector.

**data set** A data set is made up of all labels and data stored in memory of any single analyzer machine or instrument tool. Multiple data sets can be displayed together when sourced into a single display tool. The Filter tool is used to pass on partial data sets to analysis or display tools.

**debug mode** See *monitor*.

**delay** The delay function sets the horizontal position of the waveform on the screen for the oscilloscope and timing analyzer. Delay time is measured from the trigger point in seconds or states.

**demo mode** An emulation control session which is not connected to a real target system. All windows can be viewed, but the data displayed is simulated. To start demo mode, select *Start User Session* from the Emulation Control Interface and enter the demo name in the *Processor Probe LAN Name* field. Select the *Help* button in the *Start User Session* window for details.

**deskewing** To cancel or nullify the effects of differences between two different internal delay paths for a signal. Deskewing is normally done by routing a single test signal to the inputs of two different modules, then adjusting the Intermodule Skew so that both modules recognize the signal at the same time.

**device under test** The system under test, which contains the circuitry you are probing. Also known as a *target system*.

**don't care** For *terms*, a "don't care" means that the state of the signal (high or low) is not relevant to the measurement. The analyzer ignores the state of this signal when determining whether a match occurs on an input label. "Don't care" signals are still sampled and their values can be displayed with the rest of the data. Don't cares are represented by the *X* character in numeric values and the dot (.) in timing edge specifications.

**dot (.)** See *edge terms*, *glitch*, *labels*, and *don't care*.

**double-click** When using a mouse as the pointing device, to double-click an item, position the cursor over the item, and then quickly press and release the *left mouse button* twice.

**drag and drop** Using a Mouse: Position the cursor over the item, and then press and hold the *left mouse button*. While holding the left mouse button down, move the mouse to drag the item to a new location. When the item is positioned where you want it, release the mouse button.

Using the Touchscreen:
Position your finger over the item, then press and hold finger to the screen. While holding the finger down, slide the finger along the screen dragging the item to a new location. When the item is positioned where you want it, release your finger.

**edge mode** In an oscilloscope, this is the trigger mode that causes a trigger based on a single channel edge, either rising or falling.

**edge terms** Logic analyzer trigger resources that allow detection of transitions on a signal. An edge term can be set to detect a rising edge, falling edge, or either edge. Some logic analyzers can also detect no edge or a *glitch* on an input signal. Edges are specified by selecting arrows. The dot (.) ignores the bit. The asterisk (*) specifies a glitch on the bit.

**emulation module** A module within the logic analysis system mainframe that provides an emulation connection to the debug port of a microprocessor. An E5901A emulation module is used with a target interface module (TIM) or an analysis probe. An E5901B emulation module is used with an E5900A emulation probe.

**emulation probe** The stand-alone equivalent of an *emulation module*. Most of the tasks which can be performed using an emulation module can also be performed using an emulation probe connected to your logic analysis system via a LAN.

**emulator** An *emulation module* or an *emulation probe*.

**Ethernet address** See *link-level address*.

**events** Events are the things you are looking for in your target system. In the logic analyzer interface, they take a single line. Examples of events are *Label1 = XX* and *Timer 1 > 400 ns*.

**filter expression** The filter expression is the logical *OR* combination of all of the filter terms. States in your data that match the filter expression can be filtered out or passed through the Pattern Filter.

**filter term** A variable that you define in order to specify which states to filter out or pass through. Filter terms are logically OR'ed together to create the filter expression.

**Format** The selections under the logic analyzer *Format* tab tell the logic analyzer what data you want to collect, such as which channels represent buses (labels) and what logic threshold your signals use.

**frame** The Agilent Technologies 16600A-series or 16700A/B-series logic analysis system mainframe. See also *logic analysis system*.

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

**glitch** A glitch occurs when two or more transitions cross the logic threshold between consecutive timing analyzer samples. You can specify glitch detection by choosing the asterisk (*) for *edge terms* under the timing analyzer Trigger tab.

**grouped event** A grouped event is a list of *events* that you have grouped, and optionally named. It can be reused in other trigger sequence levels. Only available in Agilent Technologies 16715A, 16716A, and 16717A logic analyzers.

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

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

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

**intermodule bus** The intermodule bus (IMB) is a bus in the frame that allows the measurement modules to communicate with each other. Using the IMB, you can set up one instrument to *arm* another. Data acquired by instruments using the IMB is time-correlated.

**intermodule** Intermodule is a term used when multiple instrument tools are connected together for the purpose of one instrument arming another. In such a configuration, an arming tree is developed and the group run function is designated to start all instrument tools. Multiple instrument configurations are done in the Intermodule window.

**internet address** Also called Internet Protocol address or IP address. A 32-bit network address. It is usually represented as decimal numbers separated by periods; for example, 192.35.12.6. Ask your LAN administrator if you need an internet address.

**labels** Labels are used to group and identify logic analyzer channels. A label consists of a name and an associated bit or group of bits. Labels are created in the Format tab.

**line numbers** A line number (Line #s) is a special use of *symbols*. Line numbers represent lines in your source file, typically lines that have no unique symbols defined to represent them.

**link-level address** Also referred to as the Ethernet address, this is the unique address of the LAN interface. This value is set at the factory and cannot be changed. The link-level address of a particular piece of equipment is often printed on a label above the LAN connector. An example of a link-level address in hexadecimal: 0800090012AB.

**local session** A local session is when you run the logic analysis system using the local display connected to the product hardware.

**logic analysis system** The Agilent Technologies 16600A-series or 16700A/B-series mainframes, and all tools designed to work with it. Usually used to mean the specific system and tools you are working with right now.

**machine** Some logic analyzers allow you to set up two measurements at the same time. Each measurement is handled by a different machine. This is represented in the Workspace window by two icons, differentiated by a *1* and a *2* in the upper right-hand corner of the icon. Logic analyzer resources such as pods and trigger terms cannot be shared by the machines.

**markers** Markers are the green and yellow lines in the display that are labeled *x*, *o*, *G1*, and *G2*. Use them to measure time intervals or sample intervals. Markers are assigned to patterns in order to find patterns or track sequences of states in the data. The x and o markers are local to the immediate display, while G1 and G2 are global between time correlated displays.

**master card** In a module, the master card controls the data acquisition or output. The logic analysis system references the module by the slot in which the master card is plugged. For example, a 5-card Agilent Technologies 16555D would be referred to as *Slot C: machine* because the master card is in slot C of the mainframe. The other cards of the module are called *expansion cards*.

**menu bar** The menu bar is located at the top of all windows. Use it to select *File* operations, tool or system *Options*, and tool or system level *Help*.

**message bar** The message bar displays mouse button functions for the window area or field directly beneath the mouse cursor. Use the mouse and message bar together to prompt yourself to functions and shortcuts.

**module/probe interconnect cable**

The module/probe interconnect cable connects an E5901B emulation module to an E5900B emulation probe. It provides power and a serial connection. A LAN connection is also required to use the emulation probe.

**module** An instrument that uses a single timebase in its operation. Modules can have from one to five cards functioning as a single instrument. When a module has more than one card, system window will show the instrument icon in the slot of the *master card*.

**monitor** When using the Emulation Control Interface, running the monitor means the processor is in debug mode (that is, executing the debug exception) instead of executing the user program.

**panning** The action of moving the waveform along the timebase by varying the delay value in the Delay field. This action allows you to control the portion of acquisition memory that will be displayed on the screen.

**pattern mode** In an oscilloscope, the trigger mode that allows you to set the oscilloscope to trigger on a specified combination of input signal levels.

**pattern terms** Logic analyzer resources that represent single states to be found on labeled sets of bits; for example, an address on the address bus or a status on the status lines.

**period (.)** See *edge terms*, *glitch*, *labels*, and *don't care*.

**pod pair** A group of two pods containing 16 channels each, used to physically connect data and clock signals from the unit under test to the analyzer. Pods are assigned by pairs in the analyzer interface. The number of pod pairs avalaible is determined by the channel width of the instrument.

**pod** See *pod pair*

**point** To point to an item, move the mouse cursor over the item, or position your finger over the item.

**preprocessor** See *analysis probe*.

**primary branch** The primary branch is indicated in the *Trigger sequence step* dialog box as either the *Then find* or *Trigger on* selection. The destination of the primary branch is always the next state in the sequence, except for the Agilent Technologies 16517A. The primary branch has an optional occurrence count field that can be used to count a number of occurrences of the branch condition. See also *secondary branch*.

**probe** A device to connect the various instruments of the logic analysis system to the target system. There are many types of probes and the one you should use depends on the instrument and your data requirements. As a verb, "to probe" means to attach a probe to the target system.

**processor probe** See *emulation probe*.

**range terms** Logic analyzer resources that represent ranges of values to be found on labeled sets of bits. For example, range terms could identify a range of addresses to be found on the address bus or a range of data values to be found on the data bus. In the trigger sequence, range terms are considered to be true when any value within the range occurs.

**relative** Denotes time period or count of states between the current state and the previous state.

**remote display** A remote display is a display other than the one connected to the product hardware. Remote displays must be identified to the network through an address location.

**remote session** A remote session is when you run the logic analyzer using a display that is located away from the product hardware.

**right-click** When using a mouse for a pointing device, to right-click an item, position the cursor over the item, and then quickly press and release the *right mouse button*.

**sample** A data sample is a portion of a *data set*, sometimes just one point. When an instrument samples the target system, it is taking a single measurement as part of its data acquisition cycle.

**Sampling** Use the selections under the logic analyzer Sampling tab to tell the logic analyzer how you want to make measurements, such as State vs. Timing.

**secondary branch** The secondary branch is indicated in the *Trigger sequence step* dialog box as the *Else on* selection. The destination of the secondary branch can be specified as any other active sequence state. See also *primary branch*.

**session** A session begins when you start a *local session* or *remote session* from the session manager, and ends when you select *Exit* from the main window. Exiting a session returns all tools to their initial configurations.

**skew** Skew is the difference in channel delays between measurement channels. Typically, skew between modules is caused by differences in designs of measurement channels, and differences in characteristics of the electronic components within those channels. You should adjust measurement modules to eliminate as much skew as possible so that it does not affect the accuracy of your measurements.

**state measurement** In a state measurement, the logic analyzer is clocked by a signal from the system under test. Each time the clock signal becomes valid, the analyzer samples data from the system under test. Since the analyzer is clocked by the system, state measurements are *synchronous* with the test system.

**store qualification** Store qualification is only available in a *state measurement*, not *timing measurements*. Store qualification allows you to specify the type of information (all samples, no samples, or selected states) to be stored in memory. Use store qualification to prevent memory from being filled with unwanted activity such as no-ops or wait-loops. To set up store qualification, use the *While storing* field in a logic analyzer trigger sequence dialog.

**subnet mask** A subnet mask blocks out part of an IP address so that the networking software can determine whether the destination host is on a local or remote network. It is usually represented as decimal numbers separated by periods; for example, 255.255.255.0. Ask your LAN administrator if you need a the subnet mask for your network.

**symbols** Symbols represent patterns and ranges of values found on labeled sets of bits. Two kinds of symbols are available:

- Object file symbols - Symbols from your source code, and symbols generated by your compiler. Object file symbols may represent global variables, functions, labels, and source line numbers.
- User-defined symbols - Symbols you create.

Symbols can be used as *pattern* and *range* terms for:

- Searches in the listing display.
- Triggering in logic analyzers and in the source correlation trigger setup.
- Qualifying data in the filter tool and system performance analysis tool set.

**system administrator** The system administrator is a person who manages your system, taking care of such tasks as adding peripheral devices, adding new users, and doing system backup. In general, the system administrator is the person you go to with questions about implementing your software.

**target system** The system under test, which contains the microprocessor you are probing.

**terms** Terms are variables that can be used in trigger sequences. A term can be a single value on a label or set of labels, any value within a range of values on a label or set of labels, or a glitch or edge transition on bits within a label or set of labels.

**TIM** A TIM (Target Interface Module) makes connections between the cable from the emulation module or emulation probe and the cable to the debug port on the system under test.

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

**timer terms** Logic analyzer resources that are used to measure the time the trigger sequence remains within one sequence step, or a set of sequence steps. Timers can be used to detect when a condition lasts too long or not long enough. They can be used to measure pulse duration, or duration of a wait loop. A single timer term can be used to delay trigger until a period of time after detection of a significant event.

**timing measurement** In a timing measurement, the logic analyzer samples data at regular intervals according to a clock signal internal to the timing analyzer. Since the analyzer is clocked by a signal that is not related to the system under test, timing measurements capture traces of electrical activity over time. These measurements are *asynchronous* with the test system.

**tool icon** Tool icons that appear in the workspace are representations of the hardware and software tools selected from the toolbox. If they are placed directly over a current measurement, the tools automatically connect to that measurement. If they are placed on an open area of the main window, you must connect them to a measurement using the mouse.

**toolbox** The Toolbox is located on the left side of the main window. It is used to display the available hardware and software tools. As you add new tools to your system, their icons will appear in the Toolbox.

**tools** A tool is a stand-alone piece of functionality. A tool can be an instrument that acquires data, a display for viewing data, or a post-processing analysis helper. Tools are represented as icons in the main window of the interface.

**trace** See *acquisition*.

**trigger sequence** A trigger sequence is a sequence of events that you specify. The logic analyzer compares this sequence with the samples it is collecting to determine when to *trigger*.

**trigger specification** A trigger specification is a set of conditions that must be true before the instrument triggers.

**trigger** Trigger is an event that occurs immediately after the instrument recognizes a match between the incoming data and the trigger specification. Once trigger occurs, the instrument completes its *acquisition*, including any store qualification that may be specified.

**workspace** The workspace is the large area under the message bar and to the right of the toolbox. The workspace is where you place the different instrument, display, and analysis tools. Once in the workspace, the tool icons graphically represent a complete picture of the measurements.

**zooming** In the oscilloscope or timing analyzer, to expand and contract the waveform along the time base by varying the value in the s/Div field. This action allows you to select specific portions of a particular waveform in acquisition memory that will be displayed on the screen. You can view any portion of the waveform record in acquisition memory.

# Index

**A**

adding terms, 9
Align to x Byte option, 37
Align to x Byte option for symbols, 37
All, 11
ASCII format, 25
ASCII format symbols, 25, 27, 28, 29
ASCII symbol file, 28

**B**

beyond prefetch depth, 37
browser dialog, 33
browser, symbol search, 33
browsing, 36
browsing the symbol database, 36

**C**

clear, 12
code, assigning address offsets, 23
COFF symbol reader options, 31
comments, 29
creating a file, 25
creating ASCII symbol files, 25
creating powerful filter expressions, 15

**D**

data displayed in symbolic form, 20
delete all, 9
delete term, 9
deleting labels, 12
deleting terms, 9
displaying symbols to represent data, 20

**E**

ELF symbol reader options, 31
ELF/stabs file format, 24
example, 25, 36, 37
example measurements, 6, 7

**F**

file formats for ASCII symbols, 25
file versions, 21
files, 21
files, loading user-defined symbol files, 39
filter expression, 9
filter terms, 9, 10, 11, 12, 13
filtering data out, 6
finding the symbol you want, 36
functions, 27

**H**

hardcopy, 18
help, symbols, 19

**I**

IEEE-695 file format, 24
in ASCII format, 27, 28, 29
in symbol browser, 36

**L**

labels, adding and deleting, 11, 12
line numbers, 28
loading, 21
loading and saving filter configurations, 17
loading files including symbols, 21
loading object file symbols, 21
loading user-defined symbol files, 39

**M**

masking off addresses of symbols, 37
matching states, filtering out, 6
matching states, passing through, 7
measurement examples, 6, 7, 8
Modify Term, 11
modify term, 12
modifying terms, 10, 11, 12, 13

**N**

Not Pattern, 11
Not Range, 11

**O**

object file symbol browser, 33
object file symbol files, 21
object file symbols, 33, 36
odd-numbered addresses, 37
odd-numbered addresses represented by symbols, 37
offset, 37
offset addresses, assigning, 23
Offset By option of the symbol browser, 37
OMF96 file format, 24
OMFx86 file format, 24

**P**

Pass All Data, 8
pass all states, 8
Pass Matching Data, 7
pass through states, 7
passing data through, 7, 8
Pattern, 11
Pattern field, 36
prefetch, 37
printer, 18
printing windows, configurations, 18

**R**

Range, 11
readers.ini file, 30
record headers, 25
relocating sections of code, 23
Remove Matching Data, 6
removing data, 6

# Index

rename term, 14
replacing labels, 13
results, 36

**S**

Scroll Files field, 33
Search Pattern field, 36
searching the symbol database, 36
sections, 27
simple example, 25
simple form, 25
source line numbers, 28
Stabs symbol reader options, 31
start address, 27
symbol demangling, 30
symbol file formats, 24
symbol file versions, 21
symbol selector dialog, 35, 36
symbol type, 33
symbol types, 33
symbols, 36
symbols, displaying to represent data, 20
symbols, how they are used in the logic analyzer, 33
symbols, loading object file symbols, 21
symbols, loading user-defined symbol files, 39
symbols, outside defined sections, 30
symbols, setting up, 21
symbols, types and use, 19
symbols, user-defined details, 38

**T**

tab, symbols, 19
terms, changing name, 14
terms, filter, 12
TI COFF file format, 24
trigger term, 33, 37
triggering beyond, 37
triggering on a symbol, 33, 36, 37
turn filter terms off and on, 15

**U**

User Symbols, 33
user-defined symbol files, loading, 39
user-defined symbols, details, 38
using pattern filter, 2

**V**

value, changing, 10
variables, 29
versions, 21
versions of symbol files, 21

**W**

wildcard characters, 36
window, printing, 18
workspace, 18