Help Volume

System: PC Connectivity

16700 Series

Connectivity

Connectivity enables you to use the logic analysis system more efficiently in your current computing environment. In addition to the PC file sharing capabilities, the logic analysis system also incorporates a Netscape browser with a Home Page that delivers the capability for remote front-panel control, quick access to measurement status, and post-processing through an Agilent IntuiLink 16700 toolbar add-in for Microsoft Excel.

Another connectivity feature is the ability to execute user programs from remote PCs and UNIX workstations that remotely control the logic analysis system. This Remote Programmatic Interface (RPI) complements the capabilities delivered by the Web server and the Home Page.

The Netscape Browser

With an integrated Netscape browser, you have quick access to software downloads, tech support, and many other valuable links. For more information on starting Netscape, or setting up internet options, refer to Netscape setup (see page 8).

Accessing the Home Page

Use the Home Page to upload connectivity software from the logic analysis system, quickly check current measurement status, and start a remote front panel session. Part of the connectivity software is the Agilent IntuiLink 16700.

If you are accessing the Home Page from a PC, or any other type of computer system, simply type the logic analysis system URL in your web browser as you would for any other web site. Example: http://hostname

Remote Front Panel

The logic analysis system allows you to remotely connect to instruments and share a connection with other users. Use the *Home Page* as mentioned above, to start a remote session.

For information on viewing the status of shared connections, refer to Shared Connections (see page 9). The *Shared Connections* status window lists all currently running remote front panel sessions. It is accessed by selecting the *Shared Connections* field on the main system window.

Uploading the Agilent IntuiLink 16700

A part of the connectivity you upload from the logic analysis system is the IntuiLink 16700 software. Components include an Excel Toolbar Add-in and an ActiveX Automation Server. Use the Excel Add-in Toolbar for post processing measurement data on your remote computer.

For information on connecting to the logic analyzer and getting data into Excel, refer to the help system in the Excel Toolbar Add-in.

The Remote Programming Interface (RPI)

The RPI enables users to run measurements and analyze data programmatically from remotely networked computers. Any programming language and computing environment that supports Microsoft’s Component Object Model (COM) can be used.

For information on programming with Microsoft’s Component Object Model (COM), refer to the help system in the Excel Toolbar Add-in.

For information on programming with the ASCII Commands, refer to ASCII Commands (see page 10) located in this help system.

For information on PC File Sharing, refer to Mapping Windows/NT Network Drives (see the *Agilent Technologies 16700A/B-Series Logic Analysis System* help volume) and Share Analyzer Drive (see the

Agilent Technologies 16700A/B-Series Logic Analysis System help volume) under *The System Administrator Tools* in this help system.

Contents

1

16700 Series

Connectivity

Netscape Setup

The only required setup is the Internet Proxies. Access the Proxy setup dialog by selecting *Edit - Preferences - Advanced - Proxies* from the browser menu bar.

For more information, refer to the Netscape help or contact your system administrator.

View Shared Connections

To view the status of shared connections, from the main *System* window, select *File*, then *View Shared Connections*.

NOTE: When you first access the *Shared Connection* status window, all remote front panel sessions currently running are listed. You must periodically select *Refresh* to ensure the list is updated with any new connections.

Currently Active Connections

The top list in the status window shows all users who are currently sharing a session and the time they connected. The user name appears as either an IP address or an alias.

For more information on connecting to a shared remote front panel session, refer to the online help on the Home Page of the logic analysis system.

Previous Connections

The bottom list in the status window shows user names who have been connected in the past, with the latest disconnect on top and the oldest disconnect at the bottom of the list.

The user name appears as either *Remote_X_Session* if it was an X-connection, or, by the IP address or alias if it was a Java-connection.

For information on how to start a *Shared Connection*, refer to the analyzer's *Home Page*.

ASCII Commands

The Remote Programming Interface (RPI) is available in two forms. While only the ASCII RPI is explained here, information is available on the ActiveX/COM RPI when you upload the IntuiLink 16700 software components and access the Excel Add-in Toolbar help system.

The ASCII RPI is a simple mechanism that allows a user on a remote host to open a TCP socket connection to a logic analyzer instrument. Through this connection, simple ASCII string commands are sent, ASCII responses from the instrument are received, and binary or ASCII trace data is transferred to the host running the RPI program.

- wait (see page 42)

Hardware Module Commands

- analyzer (see page 27)
- scope (see page 31)
- pattgen (see page 35)
- emulator (see page 36)

Software Tool Commands

- listing (see page 37)
- compare (see page 39)
- fileout (see page 41)

NOTE: To print the complete list of ASCII RPI commands, go to *File* > *Print*, then select *Entire Volume*.

Use Model

In order to create an easy to use, yet powerful remote control mechanism, the design of the RPI adheres to the basic use model of "load-run-store".

Create a Configuration File

Set up an instrument configuration for the desired measurement while sitting in front of the logic analyzer. Save this measurement configuration to a configuration file. This allows you to use all the power of the instrument to setup your desired measurement.

Load-Run-Store

Once a configuration file is saved, write an RPI program that will remotely load this saved configuration file, modify a few critical measurement parameters, run the analyzer until the measurement is complete, and then offload the results or the raw trace data for post processing on the host.

System Setup

The RPI language is easily used on any host platform. However, before you can run your RPI program, the logic analyzer must be set up on the LAN. This is done by getting the appropriate network information from your system administrator, and entering this information into the logic analyzer.

1. Select the *Sys Admin* icon on the main toolbar, then select the *Network*! tab.
2. From the Network dialog, select the *Security* tab and make sure the *Remote Programming Interface* is enabled.

NOTE: To increase security when the RPI is not being used, disable the RPI interface.

Learning and Debugging Programs

Once setup on the LAN, you are ready to connect and start writing and debugging RPI programs. A simple way to learn and experiment with the RPI command language is to telnet from your remote computer to the logic analysis system specifying the special *port address* of *6500*.

For example, type: "telnet my_logic_analyzer 6500" where *my_logic_analyzer* is its IP address or machine alias name, then press the *Enter* key.

This will open a direct socket connection to the RPI in the logic analysis system. You know you have a connection when the -> prompt appears on a blank command line. This command line prompt indicates that the RPI is ready to accept a command.

Exercise:

At the prompt type: "modules"

The RPI polls the instrument cardcage and reports a list of all HW modules currently in the frame.

At the prompt type: "lock"

The RPI puts a full screen message box on the instrument console to warn people that the instrument is currently in use via the Remote Programming Interface (RPI).

This telnet mechanism is also useful in helping to debug RPI programs under development. You can have a debug telnet connection open at the same time an RPI program is running.

Data Transfers

To provide both a fast and easy to use process for data transfer, an uncompressed binary format is used. One of the benefits of this format is that it’s easy to decode and the software required to decode the binary is very simple.

It should be noted that since all data values are transferred in byte-aligned columns, there will be some generation of white space, especially when transferring large numbers of single-bit values.

Although data transfers from logic analyzers and scopes are in binary form, data transfers from the Listing tool will come in ASCII form. For the Listing tool, the ASCII form allows GUI control over the numeric formats used, as well as the use of powerful SW Analysis tools such as the Serial Analysis tool or the Filter tool.

Sample Programs

Source code for some sample RPI programs and an RPI utility library is shipped with your Agilent Logic Analyzer. They can be found in the directory "/logic/demo/rpi/".

You can transfer all files in this directory, including the makefile, onto your remote host using the various connectivity methods available from the logic analysis system. These include ftp, NFS, PC file sharing, or simply using the built in floppy drive.

After the files are transferred, you can compile and run the programs to get familiar with the basic capabilities of the RPI.

config Command

```
Use this command to load a previously saved instrument configuration file.
This operation will restore the instrument to the same setup that was
stored in the configuration file.   It also allows the currently configured
instrument to save it's current state to a new configuration file.

NOTE:
Configuration files can be located on the local hard drive of the instrument
OR, through the use of NFS mounting and PC sharing, can be located on any
mountable UNIX or sharable PC disk drive.

When saving a configuration file that already exists, an error message
will result.  However, using the -f argument with the -s option will force an
overwrite of an existing file.


Syntax:
        config [-l | -s [-f]] config_file

Options:

    -l config_file
        Loads a configuration file named "config_file".

    -s [-f] config_file
        Saves the current configuration and data to a file named "config_file".
        If the file name already exists, use the -f argument.

Examples:

    config -l pentium._E
        Loads a configuration file named "pentium._E".

    config -s myconfig
        Saves the current workspace configuration with data to a file
        named "myconfig".
```

ctl_port Command

```
This command provides access to the instrument target control port. It
will read and return the value present on the pins of the control port or set the
port to a specific value.

Values for the target control port can be set using the same syntax as
analyzer -trig commands:

    #hfx
        Hex where upper 4 bits are high and lower 4 bits stay the same
        (don't care).

    #b11110000
        Binary where upper 4 bits are high and lower 4 bits are low.

    #q377
        Octal where all 8 bits are high.

    #bxxxx1xxx
        Set bit 4 high, leave all others the same.

Syntax:
        ctl_port [option]

Options:

    ?
        Reads the target control port and returns an 8-bit value.

    value
        Sets the target control port to an 8-bit value.

Returns:
        <8 bit value>

Examples:

    ctl_port ?
        Reads the 8-bit value from the target control port.
        Returns:
        #he

    ctl_port #hfx
        Sets the target control port output pins: upper 4bits go to
        High, lower 4 bits stay what they were.
```

clear Command

```
The clear command clears the workspace of all modules and tools.
This command does NOT effect any system administration functions such
as LAN settings, printer settings, etc...

Syntax:

        clear[option]

Options:

        No options

Example:

    clear
        Clears the workspace of all modules and tools.
```

lock, unlock Command

```
This command controls the accessibility of the instrument for other users.
When locked, no other users can access the instrument from the local display
or remotely. If desired, a custom message can be shown on the local display
instead of a default message. As an example, a custom message might give
information as to who has the unit locked. The instrument can then be
unlocked when desired.

Syntax:

        lock ["message text"], unlock

Options:

    lock "message text"
        Locks all users out of the instrument. If a custom message is sent,
        it must be contained in quotes.

    unlock
        Unlocks the instrument to allow other users.

Examples:

    lock
        Locks the instrument and display a default message.

    unlock
        Unlocks a currently locked instrument.

    lock "Currently in use by Tom"
        Locks an instrument and display the message
        "Currently in use by Tom".
```

modules Command

```
Use this command to poll the system to identify the HW modules in the system,
and return information on Type, Slot, and State.  There are two states that
modules can be in, "active" or "available". Available means that the HW module
is plugged into a slot in the frame and is available to be included in a
measurement. The second state is "active".  In this state, the HW module is
"activated" by being included in a measurement setup.  When included in a
measurement setup, the HW module is both visible in the instrument workspace
and from the "Navigate" pulldown menu in the instrument GUI. Active modules
have either the default or user-defined ASCII names associated with them.


Syntax:
        modules [-a | -slot slot_id | -expanders]

Options:

    With no option
        Returns a list of both Active and Available modules. Type, Slot, and
        State information for each listed module is returned.

    -slot slot_id
        Returns information on a module in a specified "slot_id". The slot
        identifier is A-J for measurement modules and 1-4 for emulation modules.

    -a
        Returns a list of Active modules only. Type, Slot, and State
        information for each listed module is returned.

    -expanders
        Lists how many (and which) expander/slave cards each slot has.

Returns:
        For each module listed, the following information is returned:
        Type, Slot, State, "Name", "Model", and "Description"

        The above information is returned in the following form:
        Type is a 2-character string representing a logic analyzer (LA),
        oscilloscope (SC), pattern generator (PG), and emulation (EM).

        Slot is the letter or number identifier of the slot (A-J for
        measurement modules, 1-4 for emulation modules). State is shown as
        either a "1" if the module is active, or "0" if inactive and available.

        Also returned is the following HW module information:
        "Name", "Model", and "Description"

        Example: LA B 1 "Analyzer<B>" "16550A" "100MHz State/500MHz Timing"
        Where:   LA=logic analyzer, B=slot B, 1=active state, Name=Analyzer<B>,
                 Model=16550A, and Description=100MHz State/500MHz Timing

Examples:

    modules
        Returns:
        LA B 1 "Analyzer<B>" "16550A" "100MHz State/500MHz Timing"
        LA B 0 "Analyzer<B2>" "16550A" "100MHz State/500MHz Timing"
        LA D 0 "Analyzer<D>" "16556A" "1M Sample 100 MHz State/400 MHz Timing"
        LA D 0 "Analyzer<D2>" "16556A" "1M Sample 100 MHz State/400 MHz Timing"
        SC E 1 "Scope<E>" "16534A" "2GSa/s Oscilloscope"
        EM 1 0 "Emulator<1>" "Emulation Module" "Not Configured"

    modules -a
        Returns:
        LA B 1 "Analyzer<B>" "16550A" "100MHz State/500MHz Timing"
        SC E 1 "Scope<E>" "16534A" "2GSa/s Oscilloscope"
```

```
modules -expanders
    Slot D is a master card, with 1 expander card in slot C:
    Slot A: 0 expanders
    Slot B: 0 expanders
    Slot D: 1 expanders
    C
    Slot E: 0 expanders
    Slot 1: 0 expanders
```

start Command

```
This command starts HW modules running. The definition of running is dependent
on the HW module selected. For analyzer modules, "running" is when their trace
analyzers begin looking for a trigger, when oscilliscopes begin looking for a
trigger, when pattern generators begin generating vectors, and emulation probes
start the processor running.

All active modules may be started at once by using no option, individual modules
started
with -n name or -slot slot_id, and all modules in a "group run" list can be
started with the
-g option.

The -rep option applies to analyzers, oscilliscopes, and pattern generator
modules but does not apply to emulation probes. When used, it sets these
modules to repetitive run mode.

Syntax:

        start [-g | -n name | -slot slot_id] [-rep]

Options:

    no option
        Starts all active modules running.

    -n name
        Starts the active module named "module" running.

    -slot slot_id
        Starts a specific module named "slot_id". The slot identifier
        is A-J for measurement modules and 1-4 for emulation modules.

    -g
        Starts all modules configured in the group run list running.

    -rep
        Starts LA, SC, and PG modules running in repetitive mode.

Examples:

    start -a
        Starts all active modules running.

    start -n Emulator<2>
        Starts the processor in the emulation probe module named
        "Emulator<2>" running.

    start -g -rep
        Starts all modules in the group run list running repetitively.
```

status Command

This command queries active modules and returns their measurement status. Status information returned depends on the module being queried. Analyzers and oscilloscopes can be stopped or running. Pattern generators can be stopped or running. Emulators can be running, reset, or in a break state. All active modules may be queried at once by using no option, individual modules with -n name or -slot slot_id, and all modules grouped in the "group run" list are queried with the -g option.

Using "no option" on the command line, returns "running" if any measurement module is running, and "stopped" if no module in running. Remember, an emulator is not a measurement module, so the state of the target processor has no impact on the result of this command unless the emulator is specified by the "-n name" option.

Syntax:

```
        status [-g | -n name | -slot slot_id] [-v] [-text] [-clear]
```

Options:

```
    With no option
        Returns status of the frame. Will be either running, stopped, or
        waiting for trigger.

    -v
        Returns verbose status information instead of running/stopped.

    -slot slot_id
        Returns the status of a specific module named "slot_id". The slot
        identifier is A-J for measurement modules, 1-4 for emulation modules.

    -n name
        Returns the status of the active module named "name".

    -g
        Returns the status of all modules in the group run list.

    -text
        Retrieve the text messages from the Run Status display.

    -clear
        Clear the text message in the Run Status display.
```

Examples:

```
    status
        Query if the frame is running anything.
        Returns:
        stopped

    status -v
        Query status for all active modules in the system.
        Returns:
        Analyzer<A>: stopped
        Emulator<3>: MPC860 In Background

    status -n PatternGen<J>
        Query status for current module named "PatternGen<J>".
        Returns:
        running

    status -g
        Query status for all active modules in the group run list.
        Returns:
        Pentium: waiting for trigger
        Analyzer<F>: waiting in sequence level 3
        Emulator<3>: running

    status -text
        Show the text in the Run Status messages area.
        Returns:
        Analyzer<E>: Calibration Error

    status -clear
        Clear the messages area in the Run Status display.
```

stop Command

```
This command stops HW modules that are currently running. The definition of
running is dependent on the HW module selected. For analyzer modules, "running"
is when their trace analyzers begin looking for a trigger, when oscilloscopes
begin looking for a trigger, when pattern generators begin generating vectors,
and emulation probes start the processor running.

All running HW modules may be stopped at once by using no option.  Individual
modules may be stopped with -n name or -slot slot_id, and a selected list of
modules grouped in the "group run" list are stopped with the -g option.

Using "no option", will not stop the target processor connected to an
emulation module.  To do this, you must select the emulation module with
the -n name or -slot slot_id option.

Syntax:
        stop [-g | -n name | -slot slot_id]

Options:

    no option
        Stops all actively running modules.

    -slot slot_id
        Stops a specified module in the slot "slot_id". The slot identifier
        is A-J for measurement modules and 1-4 for emulation modules.

    -n name
        Stops the actively running HW module named "name".

    -g
        Stops all running modules in the group run list.

Examples:

    stop
        Stops all actively running modules.

    stop -n PatternGen<B>
        Stops the actively running pattern generator named "PatternGen<B>".

    stop -g
        Stops all actively running modules in the group run list.
```

tools Command

```
This command queries the system and identifies the active SW tools. Tools
that are "active" are currently included in a measurement setup and appear in
the instrument workspace and from the "Navigate" pulldown menu in the
instrument GUI. These active tools have logical names associated with them.

NOTE:  Only active tools will accept their associated tool commands like
"compare", "listing", etc...

Syntax:
        tools

Returns:

        name: type ( lister, compare, fileout)

Examples:

    tools
        Returns:
        Filter<1>: Filter
        Listing<1>: Listing
        Compare<1>: Compare
        Listing<2>: Listing
        Waveform<1>: Waveform
        Waveform<2>: Waveform
```

version Command

```
This command returns the version number for the product named by the option.
If no option is used, the version number for the system software is returned.

Syntax:

        version [product]

Options:

    with no option
        Returns the software version of the system.

    product
        Returns the software version of the named named.

Returns:

        Version number for the system or named product.

Examples:

    version
        Query version number of the system software.
        Returns:
        A.01.30.00

    version MCORE
        Query version number of the MCORE processor support package.
        Returns:
        A.01.31.00

    version PROC-SUPPORT
        Query version number of the PROC-SUPPORT bundle.
        Returns:
        A.01.30.00
```

analyzer Command

```
This command accesses the data captured by an active analyzer. The analyzer
is accessed by its logical name or slot id.

This command can also return information on the last data captured including
data size and boundary ranges. You can then select which labels of data you
are interested in and transfer all states or a partial range of data out the
communication channel.

Syntax:
        analyzer [-n name] [-i]
        analyzer [-n name] -d [-l labellist | all] [-r start..end | all]
                 [-t start..end | all]

Options for Data Query:

    -n name
        Sets the focus to the analyzer named "name".

    -slot slot_id
        Selects a specific analyzer located in "slot_id". The slot identifier
        is A-J for measurement modules and 1-4 for emulation modules.

    -i
        Queries for information on the last data captured.

    -d [-l label1,label2...| all] [-r start..end | all] [-t start...end | all]

        Begin upload of binary data out of the analyzer.  Use the
        -l option to list individual labels, -r to specify a range, and
        -t to specify a time period.

NOTE:
The -n name option is used to specify a specific analyzer module. If there
is only one active module, the -n name option is not required. However,
if there are multiple analyzer modules active, you must use the -n name
at least once to specify a module focus, then again each time you want to
change the focus to another analyzer module.

Returns:

    The -i information query structure returns the following:

NOTE:
Transferring Transitional Timing Data. When capturing data in transitional
timing mode, data is only stored when a transition occurs. Therefore, when
accessing data captured by an active analyzer configured with transitional
timing enabled, it is recommended that you transfer all states. Transferring
a partial range of captured data may result in ambiguous data values until
the first transition within that range is observed.

        Run ID: 1234567890
        States: -4095..4096
        Times: -1.0e-06...1.0e-06
        5 labels
        "ADDR" 32 bits unsigned integer
        "DATA" 16 bits unsigned integer
        "STAT" 5 bits unsigned integer
        "Time" 64 bits signed integer timescale picoseconds
        "State Number" N bits signed integer

NOTE:
To select which data is sent, the -d option must be accompanied by a range
or time selection, and by a label selection.
```

```
A range selection looks like this:
    -r start..end or -r all,
where start and end are integer state numbers. If the data has states
from -4095..4096, there are 8K states. The trigger position is at state
number 0.

The range can also be selected by time values, such as:
    -t start..end or -t all
where start and end are floating-point values in units of seconds. The
trigger location is always at time 0.0. So, to select from -1 microsecond
to +1 microsecond:
    -t -1.0e-06..1.0e-06

Finally, to select labels, the -l flag is used:
    -l ADDR,DATA,STAT,Time or -l all

If a label contains white space, the label is enclosed in quotation marks:
    -l "State Number","System Clock",ADDR

Once data is selected, a two-part binary data transfer occurs. First, a
simple 8-byte header is sent, indicating how many states will be transferred,
and how many bytes for each state will be sent. Then for each state, a row
of bytes is sent containing the data for each of the selected labels as
follows:
    4 bytes - Number of records
    4 bytes - Number of bytes per record
    nrecords *bytes per record - Data

Each record contains one state or time of the data requested. For each label
selected (-l option), there are an integer number of bytes containing the
value. Labels are sorted in order by which they were requested, and if
"all" is selected, they arrive in order by which they are listed in
the -i query.

The number of bytes for each label is the lowest possible integer number
of bytes given the bit width of the label. For example, a 17-bit label will
require 3 bytes (24 bits), a 16-bit value will require 2 bytes.

Examples:

    analyzer -n Analyzer<B> -i
        Returns:
        Run ID: 1234567890
        States: -4095..4096
        Times: -1.0e-06..1.0e-06
        5 labels
        "ADDR" 32 bits unsigned integer
        "DATA" 16 bits unsigned integer
        "STAT" 5 bits unsigned integer
        "Time" 64 bits signed integer timescale picoseconds
        "State Number" N bits signed integer

    analyzer -slot C -d -l all -r all
    <begin binary data transfer>
    ...
    <end transfer>
        Uploads data for all labels at all states.

    analyzer -n Analyzer<C> -d -l all -t -0.001..0.001
    <begin binary data transfer>
    ...
    <end transfer>
        Upload data for all labels, in the time range of -1 msec to +1 msec.

    analyzer -d -l addr,data -r -100..200
    <begin binary data transfer>
    ...
    <end transfer>
        Upload specific data for labels "addr" and "data" in the range
        of -100 to 200 states.
```

```
Options for the Trigger Subsystem

The following options control the analyzer trigger subsystem. It allows
simple pattern matching with ANDed/ORed pairs, simple storage qualification,
two level sequencing, simple durations and edge triggering.

The following options also allow you to recall up to 10 defined trigger
setups from a recall buffer. This allows easy, fast switching of triggers
between measurements.

    -trig anything
        Set to trigger on anything and store everything.

    -trig recall n
        Load a prestored trigger setup from the recall buffer "n".

    -trig recall "Macro Name"
        Recall stored trigger setup by its name

    -trig condition1 [store condition2]
        Trace for a condition 1 with optional store

    -trig condition1 [store condition2] [followedby condition3 [store
    condition4]]
        Trace for condition 1 followed by a condition 2 (with optional store
        at each level)

    -trig duration condition1 [< | >] time
        Trace when you find a value occurring for the desired time

Conditions:

A "condition" is a combination of Pattern, Range, and Edge definitions.
Patterns and ranges are defined as hex, octal, or binary numbers with
optional don't care digits. To specify the number base, a prefix is used:

    #h: Hexadecimal
    #q: Octal
    #b: Binary
    #e: Edge (see below)

The 'x' character denotes a don't care digit. So to define a simple
pattern condition, we might use something like this:
    ADDR=#hFFFFXXXX
The above is a pattern condition that will search for a state when the
value of the ADDR label lies between 0xFFFF0000 and 0xFFFFFFFF.

To specify a range, two pattern specifiers are joined by a comma (,). For
example, to specify the same condition above as a range:
    ADDR=#hFFFF0000,#hFFFFFFFF

To search for an edge or a glitch, we use an "edge specifier", defined
by "#e" followed by any combination of the following characters:
    x don't care
    r rising edge
    f falling edge
    t toggling edge
    e either edge (same as toggling)
    * glitch
    g glitch (same as *)

Two conditions may be combined with an AND or an OR.  For example:
    ADDR=#hFFFF0000,#hFFFFFFFF and DATA=#exxxRxxxx
would search for a rising edge in bit 5 of DATA while ADDR is within the
range 0xFFFF0000 - 0xFFFFFFFF.

Condition examples:
    Pattern and Range Examples:
    #hFFXX0022
        Hexadecimal number with 2 don't care digits (8 don't care bits

    #q7777xxxx
```

```
        Octal number with 4 don't care digits (12 don't care bits

    #b10110110xxxx0000
        Binary number with 4 don't care bits

    #hFF00,#hFFFF
        Range from 0xff00 to 0xffff

    NOTE:
    Don't care digits are not allowed in ranges.

Edge Examples:
    #eXXXXRFEG
        Edge specifier with 4 don't care bits, then Rising, Falling,
        Either, and Glitch bits.


Examples:

    analyzer -n Analyzer<B> -trig addr=#h12e4c and ctl=#h00
        Trigger when addr=0x12e4c and ctl=0.

    analyzer -trig addr=#h12xx or addr=#h13xx store addr=#h1200,#h13ff
        Setup trigger for default analyzer to start on addresses with
        don't cares and store everything in the range 12xx to 13xx of
        label named "addr".

    analyzer -trig addr=#h210 followedby addr=#h344
        Trigger on access to address 210 followed by access to address 344.

    analyzer -trig recall=1
        Loads trigger setup from the recall buffer 1.

    analyzer -trig recall="Enter Main"
        Recalls a trigger setup named "Enter Main".

    analyzer -n MyTarget -trig duration status=#h22 > 30 ns
        Trigger analyzer named "MyTarget" when label status has value 22
        for more than 30 ns.

    analyzer -trig duration rdwr!=#h0 < 30 ns
        Trigger when no rdwr pattern is found for less than 30 ns.

    analyzer -trig io=#exxxxxFxF and cycle=#h1
        Trigger if bit 0 & 2 of label named "io" transition low while
        label cycle is at pattern binary 1.
```

scope Command

```
This command accesses the data captured by an active oscilloscope module.
The scope is selected by name or slot id, and can be queried for information
about data captured in the last run using the -i option, or data can be
uploaded using the -d option. In addition to the entire data, data can also
be uploaded from only channels of interest for a specific range of data.

A "channel" can be either a single digit channel number, as in 1,2,3, or 4,
or the channel label name, such as "Ground" or "rd/wr".  Default label names
given to the channels are "Channel D1" where the "D" is actually the slot
number of the card and the "1" is the scope channel number between
1 and 10 (if you have enough expansion cards).

Syntax:

        scope [-n name] [-slot slot_id] -i -d -l [channellist | all]
              -c [channellist | all] -r [range | all] -t [timerange | all]

Options:

Options to Access Data Capture

        -n name
        Selects the active scope module by name

        -slot slot_id
        Selects a specific scope module by a slot_id. The slot identifier
        is A-J for measurement modules.

        -i
        Query information on last data captured.

        -c
        Query names of available channels.

        -d [-l ch1,ch2,...|all][-c 1,2,...|all][-r start..end|all][-t
start..end|all]
        Begins upload of binary LBP data out of scope.

NOTE:
The -n name option is used to specify a specific scope module. If there is
only one active module, the -n name option is not required. However, if
there are multiple scope modules active, you must use the -n name at least
once to specify a module focus, then again each time you want to change the
focus to another scope module.

Returns:

        -i information query structure returns the following:

        Run ID: 374199271
        States: -16383..16384
        Times: -8.191740e-06..8.192260e-06
        4 labels
        "State Number" 32 bits signed integer
        "Time" 64 bits signed integer timescale picoseconds
        "Channel A1" 15 bits yincrement 2.5247e-04 (volts/bit) yorigin
        -1.6203e+00
        "Channel A2" 15 bits yincrement 2.5247e-04 (volts/bit) yorigin
        -1.6203e+00

        Analog data such as scope data is given in its unsigned integer format,
        and the -i information provides the scale factors needed to convert
        back to floating-point voltages.

        For "Channel A1" above, we have 15-bit integer values. To convert them
        to voltage, apply the following (where value is the 15-bit integer):

            voltage = yorigin + yincrement*value


        -c channel information query structure returns the following:

        1: "Channel A1" 15 bits yincrement 2.5247e-04 (volts/bit) yoffset
        -1.6203e+00
        2: "Channel A2" 15 bits yincrement 2.5247e-04 (volts/bit) yoffset
        -1.6203e+00
```

Examples:

```
    scope -n Scope<E> -i
        Query last data captured for scope named "Scope<E>".
        Returns:
        Run ID: 1250539440
        States: -16383..16384
        Times: -8.191659e-06..8.192341e-06
        4 labels
        "State Number" 32 bits signed integer
        "Time" 64 bits double timescale picoseconds
        "Channel A1" 15 bits yincrement 2.5247e-04 (volts/bit) yorigin
        -1.6203e+00
        "Channel A2" 15 bits yincrement 2.5247e-04 (volts/bit) yorigin
        -1.6203e+00

    scope -c
        Query available scope channels.
        Returns:
        1: "Channel A1" 15 bits yincrement 2.5247e-04 (volts/bit) yoffset
        -1.6203e+00
        2: "Channel A2" 15 bits yincrement 2.5247e-04 (volts/bit) yoffset
        -1.6203e+00

    scope -n Scope<E> -d -c all -t all
        Uploads all scope data.
        Returns:
        <begin binary data transfer>
        ...
        <end transfer>

    scope -d 1 "Ground" -r -100..200
        Upload specific data for channels "1" and "ground" in the range of
        -100 to 200 states.
        Returns:
        <begin binary data transfer>
        ...
        <end transfer>
```

Options to Access Trigger and Measurement Subsystems

Syntax:

```
       scope [-n name] [-c 1,2,...] [-l channel1,channel2,...]
       [-meas | all] [-range,...-tgmode]
```

Options:

These options to the scope command allow setting and querying of various measurement parameters and access to the automatic measurement results.

A "channel" can be either a single digit channel number, as in 1,2,3, or 4, or the channel label name, such as "Ground" or "rd/wr". Default label names given to the channels are "Channel D1" where the "D" is actually the slot number of the card and the "1" is the scope channel number between 1 and 10 (if you have enough expansion cards).

```
    -n name
        Selects the active scope named "name".
    -autoscale
        Autoscale the scope.

    -meas [type | all]
        Query automatic measurement results. See "Automatic Measurement Types
        and Returned Value" below.

    -range channel [range | ?]
        Set or query channel range (vertical).

    -offset channel [offset | ?]
        Set or query channel offset.

    -trange [range | ?]
        Set or query display range (horizontal).

    -delay [delay | ?]
        Set or query display delay.

    -sweep [trig | auto | ?]
        Set or query triggered or auto sweep.

    -tglevel [N | ?]
        Set or query channel trigger level.

    -tgsource [channel | ext | ?]
        Set or query current trigger source.

    -tgslope [rising | falling | ?]
        Set or query trigger slope.

    -tgmode [edge | pattern | immediate | ?]
        set or query trigger mode


Automatic Measurement Types and Returned Values

    all
        return structure with all measurement results.

    falltime
        .90% to 10% time of leftmost falling edge.
        Falltime: 0.000000268200

    risetime
        10% to 90% time of leftmost rising edge.
        Risetime: 0.000000420800

    frequency
        Frequency: 9.9E37

    preshoot
        Preshoot: 0.000000000000

    overshoot
        Overshoot: 0.000000000000

    period
        Period: 9.9E37

    pwidth
        +Width: 9.9E37

    nwidth
        -Width: 0.000003408333

    vamp
        Vamp: 0.113105058670

    vavg
        Vavg: -0.058784030290
    vbase
        Vbase: -0.117573976517

    vmax
        Vmax: -0.004468917847

    vmin
        Vmin: -0.117573976517

    vpp
        Vpp: 0.113105058670

    vtop
        Vtop: -0.004468917847

    vacrms
        Vacrms: 0.012887802882

    vdcrms
        Vdcrms: 0.060179378230

To select which scope channel the measurement results come from, use the
-c channel option as follows:

        scope -c 1 -meas all
        or
scope -l Channel E2 -meas period

To query the current setting of any of the trigger options, use a "?"
instead of a value. For example:

        scope -trange 1 0.001

sets the display range to 0.001 seconds (1 msec). To query the display time
range:

        scope -trange ?

To set the display range to 0.001 seconds (1 msec):

        scope -trange 0.001

Examples:

    scope -n Oscilloscope<B> -meas risetime
        Query rise time for scope named "Oscilloscope<B>".
        Returns:
        Risetime:0.004

    scope -tgsource 3
        Set trigger source to channel 3.

    scope -delay ?
        Query current timebase delay.
        Returns:
        0.00346
```

pattgen Command

```
This command provides access to the pattern generator module. It allows the
user to load an ASCII stimulus file into the pattern generator module. The
user can also query a vector number for its value, or modify single vectors
within a currently loaded stimulus file.

Syntax:
        pattgen [option]

    -n name
        Selects a pattern generator module. See the note below.

    -slot slot_id
        Selects a specific pattern generator module by a slot_id. The slot
        identifier is A-J for measurement modules.

    -f vectorfile
        Loads an ASCII stimulus file named "vectorfile" into the target module.

    -v vector_num [label1=value1,label2=value2,...]
        Queries single vectors, or modifies single vectors with new values
        for each specified label.

NOTE:
The -n name option is used to specify a specific pattern generator module.
If there is only one installed module, the -n name option is not required.
However, if there are multiple pattern generator modules installed, you must
use the-n name at least once to specify a module focus, then again each time
you want to change the focus to another pattern generator module.

Returns:

-v vector_num query information structure returns the following:

        label1=value
        label2=value
        etc...

Examples:

    pattgen -f mem_ctl
        Loads vectors from the file named "mem_ctl".

    pattgen -n Pattgen<B> -v 3
        First sets the focus to the pattern generator module named
        "Pattgen<B>", then queries for the value of vector number 3.
        Returns:
        data=3
        ctl=3
        chip_sel=0

    pattgen -v 3 chip_sel=1
        Modifys the value in vector 3 under label "chip_sel" to a
        value of 1.
```

emulator Command

```
This command provides access to emulation probe HW modules. Processor control
includes resetting the processor, breaking into the monitor, step, or starting
the processor running (using the system "start" command or the -run flag).
It can also download binary processor code into the target memory.

Syntax:

        emulator [-n name] [-slot slot_id] [-reset | -break | -run | -step]

Options:

    -n name
        Selects the emulator named "name". See the note below.

    -slot slot_id
        Selects the emulator in "slot_id".The slot identifier is 1-4 for
        emulation modules.

    -reset
        Resets the processor on the target system.

    -break
        Breaks the target system's processor into the monitor.

    -run
        Runs the processor.

    -step
        Steps the processor.

NOTE:
The -n name option is used to specify a specific emulation module. If there
is only one active module, the -n name option is not required. However,
if there are multiple active emulation modules, you must use -n name at least
once
to specify an emulation module focus, then again each time you want to change
the focus to another emulation module.

Examples:

    emulator -n Emulator<1> -r
        First sets the focus to the emulation module "Emulator<1>, then resets
        the procesor on the target system.

    emulator -break
        Breaks the processor on the target system into the monitor.

    emulator -run
        Runs the processor on the target system.

    emulator -step
        Steps the processor on the target system.
```

listing Command

```
This command accesses the data displayed by an active lister. The lister
is accessed by it’s logical name.  This command can return information on
the last data captured including data size, labels, and boundary ranges.
You can then select which labels of data you are interested in and transfer
all states or a partial range of data out the communication channel.

Syntax:
        listing [-n name] [-i] -d -l [labellist | all] -r [range | all]

Options:

    -n name
        Specifies a specific lister display by name.

    -i
        Query for information on the last data captured.

    -d -l [label1,label2,... | all]
        Begins upload of ASCII LBP data out of the lister for a specific
        list of labels, or all labels.

    -r [start..end | all]
        Specifies a range between start-state and end-state, or all states.

NOTE:
The -n name option is used to specify a specific lister display. If there is
only one lister display, the -n name option is not required. However, if there
are multiple lister displays, you must use -n name at least once to specify a
lister display focus, then again each time you want to change the focus to
another lister display.

Returns:

The -i query returns the following:
        Run ID: 1799474489
        States: -2032..2063
        Times: -8.128000e-06..8.256000e-06
        "State Number" 12 characters format Decimal
        "Lab1" 4 characters format Hex
        "Time" 11 characters format Absolute

NOTE:
A maximum of 30,000 states can be transferred by this command.

Examples:

    listing -n Lister<2> -i
        Sets focus to Lister<2>, then queries for information on its last
        data captured.
        Returned:
        Run ID: 1799474489
        States: -2032..2063
        Times: -8.128000e-06..8.256000e-06
        "State Number" 12 characters format Decimal
        "Lab1" 8 characters format Hex
        "Time" 11 characters format Absolute

    listing -n MEMBD -d -l all -r all
        Sets focus to Lister named MEMBD, then uploads data on all labels
        in all states.
        Returns:
        <begin ASCII transfer>
        ...
        <end transfer>
```

```
listing -d -l addr,data -r -100,200
    Uploads specific data for labels "addr" and "data" in the range of
    -100 to 200 states.
    Returns:
    <begin ASCII transfer>
    ...
    <end transfer>
```

compare Command

This command accesses the SW compare tool. A compare tool that is active on the workspace automatically executes a compare against the reference buffer whenever an analyzer captures a new trace.

The -i option returns the number of differences found. If the number -1 is returned, it means the compare has not been executed. The -l option returns a list of label pairs and their masks.

There are two ways to do a compare. One is to compare a dataset with a reference buffer, and another is to compare one dataset to another from another tool (perhaps fileIn from a simulation).

The more typical compare is against a reference. In this case, label pairs usually look like the following:

```
        addr,addr_ref
```

Because it is possible to compare any two labels (ie, "ADDR,DATA"), it is possible to set a compare mask by selecting both pairs. For example, we have the following two label pairs:

```
        ADDR,ADDR
        and
        ADDR,DATA
```

In order to set the mask on ADDR,DATA, we enter the following command and option:

```
        compare -m ADDR,DATA=#hffff0000
```

If all label pairs are unique, masks can be set by their first label in the pair:

```
        compare -m ADDR=#hffff0000
```

The comparison masks are values that are "ANDed" to the captured trace label before it is diffed with the reference buffer. Therefor, a "1" in a bit position means this bit is significant to compare and a "0" means this bit is a don't care.

Syntax:

```
        compare [-n name] -i -l -m [label1=mask1,label2=mask2,...]
```

Options:

-n name
    Sets focus to the active compare tool named "name".

-i
    Query information on last comparison.

-x
    Executes the compare.

-l
    List current label pairs

-m [lab1=mask |lab1,lab2=mask]

    Query or set up label comparison masks.

NOTE:
The -n tool option is used to specify a specific compare tool. If there is only one compare tool, the -n tool option is not required. However, if there are multiple compare tools, you must use -n tool at least once to specify a compare tool focus, then again each time you want to change the focus to another

```
compare tool.

Returns:

        The -i query returns the following:
        67

        The -l query returns the following:
        label1,label1_ref (mask=0xff00)
        label2,label2_ref


Examples:

    compare -n DMA_Comp<1> -i
        Sets focus to compare tool named DMA_Comp<1>, then queries for status
        differences found.
        Returns:
        1

    compare -m ctl=#hff00,-m data=#h00ff
        Set up mask #hff00 on label ctl, and mask #h00ff on label data.

    compare -m Lab1,Lab2=#hff00
        Sets mask for a label pair using both primary and secondary labels.

    compare -l
        Lists the current label pairs and their masks. (If there are no masks,
        nothing is listed).
        Return:
        Current label pairs:
        Lab1, Lab1_ref (mask=0Xff00)
        Lab1, Lab2_ref

    compare -x
        Re-execute compare

    compare -i
        See if anything failed.
        Returns:
        0

    compare -m data=#hff00 -x -i
        Changes the mask for label "data", executes a compare, and returns the
        number of differences.
        Returns:
        23
```

fileout Command

```
This command controls the saving of data from a fileout tool into a
specified file.

Syntax:
        fileout [-n name] [-f file] [-s]

Options:

    -n name
        Sets focus to a specific fileout tool named "name".

    -f file
        Defines a filename named "file" to save to.

    -s
        Save data to file previously specified.

NOTE:
The -n name option is used to specify a specific fileout tool. If there
is only one fileout tool, the -n name option is not required. However, if
there are multiple fileout tools, you must use -n name at least once to
specify a fileout tool focus, then again each time you want to change the
focus to another fileout tool.

Examples:

    fileout -n Fileout<1> -f pentium.out
        Sets focus to the fileout tool named "Fileout<1>" then defines
        the save filename as "pentium.out".

    fileout -s
        Save data to whatever file was specified with -f option. In this
        example, it was "pentium.out".
```

wait Command

```
This command causes the remote programming interface to pause for a number of
seconds, or until the current measurement completes. You can wait n seconds
or until the measurement completes by using both a delay and the -complete
option.

Without specifying a specific module, slot, or group to wait for,
"wait -complete" will wait until the entire instrument is stopped. By
specifying a specific slot, module, or tool name, or -g, you can wait until
a single measurement completes.

WARNING:
With out a timeout value, if a measurement never completes, remote
programs will hang.

Syntax:

        wait [n] [-complete] [-n name | -slot slot_id | -g]

Options:

    n
        Waits "n" seconds.

    -complete
        Waits until measurement is complete.

    -n name
        Waits until the named module stops.

    -g
        Waits until the group run group completes.

    -slot slot_id
        Waits until module in the indicated "slot_id" completes.


Examples:

    wait 10
        Waits 10 seconds.

    wait -complete
        Waits until measurement is complete.

    wait 30 -complete
        Wait until the measurement is complete, but not longer than
        30 seconds.

    wait 120 -slot D -complete
        Wait until slot D completes, but not longer than 2 minutes.

    wait -n Analyzer<B> -complete
        Wait until Analyzer<B> completes.

    wait -g -complete
        Wait until group run completes.
```

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