

ARTISAN®
TECHNOLOGY GROUP

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

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

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

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

Custom engineering
so your equipment works exactly as you specify.

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

Expert team | Trust guarantee | 100% satisfaction

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

Find the *Keysight / Agilent 16517A* at our website: ***Click HERE***

Programmer's Guide

Publication number 16517-97007
February, 2000

For Safety information, Warranties, and Regulatory information, see the pages behind the index

© Copyright Agilent Technologies 1987-2000
All Rights Reserved

Agilent Technologies 16517A/18A 4-GSa/s Timing and 1-GSa/s Synchronous State Logic Analyzer

In This Book

This guide, combined with the *Agilent Technologies 16500B/16501A Programmer's Guide*, provides you with the information needed to program the Agilent Technologies 16517A/18A logic analyzer module. Each module has its own reference to supplement the mainframe manual since not all mainframes will be configured with the same modules.

This guide is organized in three parts. Part 1 consists of chapter 1 whichs contains general information and instructions to help you get started.

Chapter 1 also contains:

- Mainframe system commands that are frequently used with the logic analyzer module
- 16517A/18A Logic Analyzer command tree
- Alphabetic command-to-subsystem directory

Part 2 consists of chapters 2 through 7 which contain the subsystem commands for the logic analyzer and chapter 8 which contains information on the SYSTem:DATA and SYSTem:SETup commands for this module.

Part 3, chapter 9, contains program examples of actual tasks that show you how to get started in programming the 16517A/18A logic analyzer. These examples are written in HP BASIC 6.2; however, the program concepts can be used in any other popular programming language that allows communications with either the GPIB or RS-232C buses.

1 Programming the Agilent Technologies

2 Format Menu Commands

3 Trigger Menu Commands

4 Waveform Menu Commands

5 Listing Menu Commands

6 Compare Menu Commands

7 Symbols Commands

8 Data and Setup Commands

9 Programming Examples

Index

Error messages for the 16517A/18A are noted in the text of many of the commands and also included in generic system error messages in the *Agilent Technologies 16500B/16501A Programmer's Guide*.

Contents

Part 1 General Information

1 Programming the HP 16517A/18A

Selecting the Module 1–3
Programming the Logic Analyzer 1–3
Mainframe Commands 1–5
Command Set Organization 1–8
Module Status Reporting 1–11
MESE<N> 1–12
MESR<N> 1–14

Part 2 Commands

2 Format Menu Commands

FORMat 2–4
CLOCk (State mode only) 2–5
LABel 2–6
REMove 2–7
SOFFset (State Mode Only) 2–8
SYNC 2–9
THReshold 2–9
TYPE 2–10

3 Trigger Menu Commands

Qualifier 3–7
TRIGger (TRACe) 3–10
ACQuisition (WIDetiming Type Only) 3–11
ARMedby 3–11
BRANch 3–12
CLEar 3–14
DURation (Timing mode only) 3–15
EDGE (Timing Mode Only) 3–17
FIND 3–18
PATTern 3–20
REName 3–21

SAMPclk (State Mode Only) 3–22
SEQuence 3–23
SETUPHOLDA (Timing Mode Only) 3–24
SETUPHOLDB (Timing Mode Only) 3–25
SETUPHOLDC (Timing Mode Only) 3–26
SPERiod 3–27
TIMER 3–28
TPOSition 3–29

4 Waveform Menu Commands

WAVeform 4–8
ACCumulate 4–9
ACQuisition (WIDetiming Type Only) 4–10
CENTer 4–10
CLRPattern 4–11
CLRStat 4–11
DELay 4–12
INSert 4–13
LABel 4–14
MINus 4–15
MMODe 4–16
OCONdition 4–17
OPATtern 4–18
OSEarch 4–19
OTIMe 4–20
OVERlay 4–21
PLUS 4–22
RANGe 4–23
REMove 4–23
RUNTil 4–24
SAMPclk (State Mode Only) 4–25
SIZE 4–26
SOFFset (State Mode Only) 4–27
SPERiod 4–28
TAVerage 4–29
TMAXimum 4–29

TMINimum 4–30
TPOSition 4–30
VRUNs 4–31
XCONdition 4–32
XOTime 4–33
XPATtern 4–33
XSEarch 4–34
XTIMe 4–35

5 List Menu Commands

LIST 5–8
ACQuisition (Timing Mode Only) 5–9
CLRPattern 5–10
CLRStat 5–10
COLumn 5–11
DATA 5–12
LINE 5–12
MMODe 5–13
OCONdition 5–14
OPATtern 5–15
OSEarch 5–16
OSTate 5–17
OTAG 5–17
OTIMe 5–18
OVERlay 5–19
REMove 5–19
RUNTil 5–20
SAMPclk (State Mode Only) 5–21
SHOW (State Mode Only) 5–22
SOFFset (State Mode Only) 5–23
SPERiod 5–24
TAVerage 5–25
TMAXimum 5–25
TMINimum 5–26
TPOSition 5–26
VRUNs 5–28

XCONdition 5–28
XOTag 5–29
XOTime 5–30
XPATtern 5–30
XSEarch 5–31
XSTate 5–32
XTAG 5–33
XTIMe 5–34

6 Compare Menu Commands

COMPare 6–4
CLEar 6–5
CMASk 6–5
COPY 6–6
DATA 6–7
FIND 6–8
LINE 6–9
MENU 6–10
RANGe 6–10
RUNTil 6–11
SET 6–12

7 Symbol Subsystem Commands

SYMBol 7–4
BASE 7–5
PATTern 7–6
RANGe 7–6
REMove 7–7
WIDTh 7–8

8 DATA and SETup Commands

Data Format 8–3
:SYSTem:DATA 8–4
Section Header Description 8–6
Section Data 8–6

Data Preamble Description 8–6
Acquisition Data Description 8–10
SYSTem:SETup 8–13

Part 3 Programming Examples

9 Programming Examples

Making a timing analyzer measurement 9–3
Making a state analyzer measurement 9–5
Transferring the logic analyzer configuration 9–9
Transferring the logic analyzer acquired data 9–11
Checking for measurement completion 9–15
Making a Compare Measurement 9–17
Using the COMPare:DATA? Query 9–22

Index

Part 1

1 Programming the Agilent Technologies 16517A/18A 1-1

General Information

Programming the Agilent Technologies 16517A/18A

Introduction

This chapter introduces you to the basic command structure used to program the Agilent Technologies 16517A/18A logic analyzer. Also included is an example program that sets up the timing analyzer for a basic timing measurement. Additional program examples are in chapter 9.

Selecting the Module

Before you can program the logic analyzer, you must first "select" it. This directs your commands to the logic analyzer.

To select the module, use the system command :SELect followed by the numeric reference for the slot location of the logic analyzer (1 through 10 refers to slot A through J respectively). For example, if the logic analyzer is in slot E, then the command:

```
:SELect 5
```

would select this module. For more information on the select command, refer to the *Agilent Technologies 16500B/16501A Programmer's Guide*.

Programming the Logic Analyzer

A typical logic analyzer program will do the following:

- select the appropriate module
- specify the analyzer type
- assign pods
- assign labels
- set pod thresholds
- specify a trigger condition
- set up the display
- specify acquisition type
- start acquiring data

The following example program sets up the logic analyzer to make a simple timing analyzer measurement.

Example

```
10    OUTPUT XXX;":SELECT 3"
20    OUTPUT XXX;":FORMAT:TYPE WIDETIMING"
30    OUTPUT XXX;":FORMAT:LABEL 'COUNT',POS,0,255"
40    OUTPUT XXX;":TRIGGER:PATTERN 'patt1', 'COUNT', '#HFF'"
50    OUTPUT XXX;":WAVEFORM:RANGE 1E-6"
60    OUTPUT XXX;":MENU 3,2"
70    OUTPUT XXX;":WAVEFORM:INSERT 'COUNT'"
80    OUTPUT XXX;":RMODE SINGLE"
90    OUTPUT XXX;":START"
100   END
```

The three Xs (XXX) after the "OUTPUT" statements in the previous example refer to the device address required for programming over either GPIB or RS-232C. Refer to your controller manual and programming language reference manual for information on initializing the interface.

Program Comments

Line 10 selects the logic analyzer in slot C.

Line 20 specifies the wide timing mode.

Line 30 sets up the Format menu by assigning the label COUNT, and assigning a polarity and channels to the label.

Line 40 selects the trigger pattern for the timing analyzer.

Line 50 sets the range to 1 μs (10 times 100 ns/div).

Line 60 changes the onscreen display to the Waveform menu.

Line 70 inserts the label "COUNT" in the Waveform menu.

Line 80 specifies the Single run mode.

Line 90 starts data acquisition.

Mainframe Commands

These commands are part of the Agilent Technologies 16500B/16501A mainframe system and are mentioned here only for reference. For more information on these commands, refer to the *Agilent Technologies 16500B/16501A Programmer's Guide*.

CARDcage? Query

The CARDcage query returns a string of integers which identifies the modules that are installed in the mainframe. The returned string is in two parts. The first five two-digit numbers identify the card type. The identification number for the 16517A logic analyzer is 04 and the identification number for the 16518A is 05. A "–1" in the first part of the string indicates no card is installed in the slot.

The five, single-digit numbers in the second part of the string indicate which slots have cards installed, which card has the controlling software for the module, and where the master card is located.

Example

```
4,5,-1,-1,32,1,1,0,0,5
```

A returned string of `4,5,-1,-1,32,1,1,0,0,5` means that an 16518A expansion card (ID number 5) is loaded in slot B and an 16517A master card (ID number 4) is loaded in slot A. The next two slots (C and D) are empty (–1). Slot E contains a logic analyzer module (ID number 32).

The next group of numbers (`2,2,0,0,5`) indicate that a two-card module is installed in slots A and B with the master card in slot B. The "0" indicates an empty slot, or the module software is not recognized or, is not loaded. The last digit (5) in this group indicates a single module card is loaded in slot E. Complete information for the CARDcage query is in the *Agilent Technologies 16500B/16501A Programmer's Guide*.

MENU Command/query

The MENU command selects a new displayed menu. The first parameter (X) specifies the desired module. The optional, second parameter specifies the desired menu in the module. It defaults to 0 if it is not specified). The query returns the currently selected and displayed menu.

The menus for the 16517A/18A Logic Analyzer are:

X,0 — Format

X,1 — Trigger

X,2 — Waveform

X,3 — Listing

X,4 — Not used

X,5 — Compare

X,6 — Not used

X,7 — Skew Adjust

SELect Command/query

The SELect command selects which module or intermodule will have parser control. SELect 0 selects the intermodule, SELect 1 through 10 selects modules A through J respectively. Values –1 and –2 select software options 1 and 2. The SELect query returns the currently selected module.

STARt Command

The STARt command starts the specified module or intermodule. If the specified module is configured in an intermodule arming tree, STARt will start all modules configured for intermodule.

STOP Command

The STOP command stops the specified module or intermodule. If the specified module is configured for intermodule, STOP will stop all modules configured in an intermodule arming tree.

STARt and STOP are Overlapped Commands. Overlapped Commands allow execution of subsequent commands while the logic analyzer operations initiated by the Overlapped Command are still in progress. For more information, see *OPC and *WAI commands in Chapter 5 of the *Agilent Technologies 16500B/16501A Programmer's Guide*.

RMODe Command/query

The RMODe command specifies the run mode (single or repetitive) for a module or intermodule. If the selected module is configured in an intermodule arming tree, the intermodule run mode will be set by this command. The RMODe query returns the current setting.

SYSTem:ERRor? Query

The SYSTem:ERRor query returns the oldest error in the error queue. In order to return all the errors in the error queue, a simple FOR/NEXT loop can be written to query the queue until all errors are returned. Once all errors are returned, the query will return zeros.

SYSTem:PRINt Command/query

The SYSTem:PRINt command initiates a print of the screen or listing buffer over the current printer communication interface. The SYSTem:PRINt query sends the screen or listing buffer data over the current controller communication interface.

MMEMory Subsystem

The MMEMory Subsystem provides access to both internal disc drives for loading and storing configurations.

INTermodule Subsystem

The INTermodule Subsystem commands are used to specify intermodule arming between multiple modules.

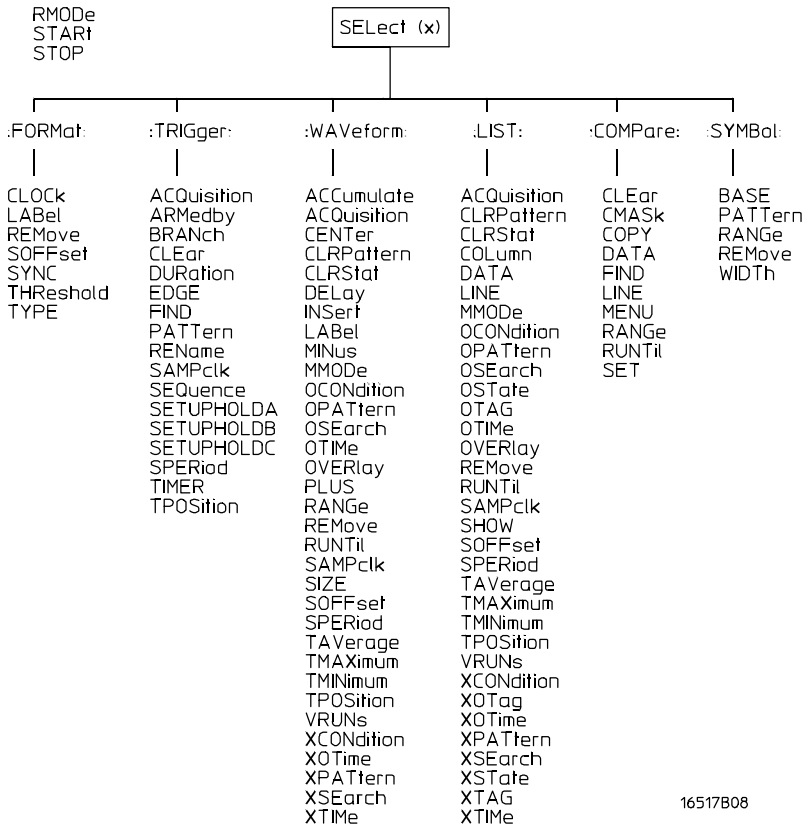
Command Set Organization

The command set for the 16517A/18A is divided into subsystems. The subsystem commands are covered in their individual chapters starting with Chapter 2, "Format Menu Commands."

Each of these chapters contains a description of the subsystem, syntax diagrams, and the commands in alphabetical order. The commands are shown in long form and short form using upper and lowercase letters. For example, LABel indicates that the long form of the command is LABEL and the short form is LAB. Each of the commands contain a description of the command and its arguments, the command syntax, and a programming example.

Figure 1-1 on the following page shows the command tree for the 16517A/18A logic analyzer module. The (x) following the SELect command at the top of the tree represents the slot number in which the logic analyzer module is installed. The number may range from 1 through 10, representing slots A through J, respectively.

Figure 1-1



16517A/18A Command Tree

Table 1-1

Alphabetical Command-to-Subsystem Directory

Command	Subsystem
ACCumulate	WAVeform
ACQuisition	LIST, TRIGger, WAVeform
ARMedby	TRIGger
BASE	SYMBol
BRANch	TRIGger
CENter	WAVeform
CLEar	COMPare, TRIGger
CLOCk	FORMat
CLRPattern	LIST, WAVeform
CLRStat	LIST, WAVeform
CMASk	COMPare
COLumn	LIST
COPY	COMPare
DATA	COMPare, LIST
DELay	WAVeform
DURation	TRIGger
EDGE	TRIGger
FIND	COMPare, TRIGger
INSert	WAVeform
LABel	FORMat, WAVeform
LINE	COMPare, LIST
MENU	COMPare
MINus	WAVeform
MMODe	LIST, WAVeform
OCONdition	LIST, WAVeform
OPATtern	LIST, WAVeform
OSEarch	LIST, WAVeform
OSTate	LIST
OTAG	LIST
OTIMe	LIST, WAVeform
OVERlay	LIST, WAVeform
PATTern	SYMBol, TRIGger
PLUS	WAVeform
RANGe	COMPare, SYMBol, WAVeform
REMove	FORMat, LIST, SYMBol, WAVeform
REName	TRIGger
RUNTil	COMPare, LIST, WAVeform
SAMPclk	LIST, TRIGger, WAVeform
SEQuence	TRIGger
SET	COMPare
SETUPHOLDA	TRIGger
SETUPHOLDB	TRIGger
SETUPHOLDC	TRIGger
SHOW	LIST
SIZE	WAVeform
SOFFset	FORMat, LIST, WAVeform
SPERiod	LIST, TRIGger, WAVeform
SYNC	FORMat
TAVerage	LIST, WAVeform
THReshold	FORMat
TIMER	TRIGger
TMAXimum	LIST, WAVeform
TMINimum	LIST, WAVeform
TPOSition	LIST, TRIGger, WAVeform
TYPE	FORMat
VRUNs	LIST, WAVeform
WIDTh	SYMBol
XCONdition	LIST, WAVeform
XOTag	LIST
XOTime	LIST, WAVeform
XPATtern	LIST, WAVeform
XSEarch	LIST, WAVeform
XSTate	LIST
XTAG	LIST
XTIMe	LIST, WAVeform

Module Status Reporting

Each module reports its status to the Module Event Status Register (MESR<N>), which in turn reports to the Combined Event Status Register (CESR) in the Agilent Technologies 16500B/16501A mainframe (see *Agilent Technologies 16500B/16501A Programmer's Guide* chapter 6). The Module Event Status Register is enabled by the Module Event Status Enable Register (MESE<N>).

The MESE<N> and MESR<N> instructions are not used in conjunction with the SELect command, so they are not listed in the 16517A/18A's command tree.

The following descriptions of the MESE<N> and MESR<N> instructions provide the module specific information needed to enable and interpret the contents of the registers.

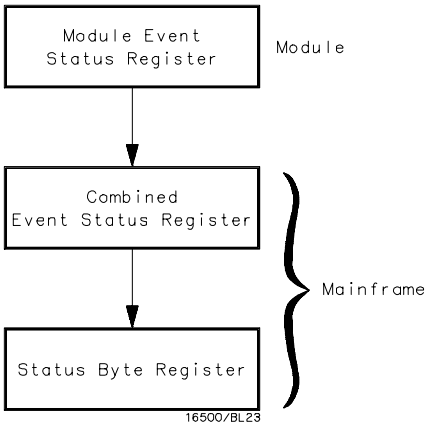


Figure 1-2

Module Status Reporting

MESE<N>

Command `:MESE<N><enable_mask>`

The MESE<N> command sets the Module Event Status Enable register bits. The MESE register contains a mask value for the bits to enable in the MESR register. A one in the MESE will enable the corresponding bit in the MESR, a zero will disable the bit.

The first parameter <N> specifies the module (1 through 10 refers to the module in slot A through J). The second parameter specifies the enable value in decimal.

Refer to table 1-2 for information about the Module Event Status register bits, bit weights, and what each bit masks for the module. Complete information for status reporting is in chapter 6 of the *Agilent Technologies 16500B/16501A Programmer's Guide*.

`<N>` {1|2|3|4|5|6|7|8|9|10} number of slot in which the module resides

`<enable_mask>` integer from 0 to 255

Example

```
OUTPUT XXX;":MESE5 1"
```

Query `:MESE<N>?`

The MESE query returns the current setting in decimal.

Returned Format `[:MESE<N>]<enable_mask><NL>`

Example

```
OUTPUT XXX;":MESE5?"
```

Table 1-2 **Module Event Status Enable Register**

Bit	Weight	Enables
7	128	Not Used
6	84	Not Used
5	32	External clock period specification
4	16	Default skew values or memory error
3	8	Pattern searches failed
2	4	Trigger found
1	2	RNT-Run until satisified
0	1	MC-Measurement complete

The Module Event Status Enable Register contains a mask value for the bits to be enabled in the Module Event Status Register (MESR). A one in the MESE enables the corresponding bit in the MESR, and a zero disables the bit.

MESR<N>

Query `:MESR<N>?`

The MESR<N> query returns the contents of the Module Event Status register in decimal. When you read the MESR, the value returned is the total bit weights of all bits that are set at the time the register is read. Reading the register clears the Module Event Status Register.

Table 1-3 shows each bit in the Module Event Status Register and their bit weights for this module.

The parameter 1 through 10 refers to the module in slot A through J respectively.

Returned Format `[MESR<N>]<status><NL>`

`<N>` {1|2|3|4|5|6|7|8|9|10} number of slot in which the module resides

`<status>` integer from 0 to 255

Example

```
OUTPUT XXX;":MESR5?"
```

Table 1-3 Module Event Status Register

Bit	Weight	Condition
7	128	Not used
6	64	Not used
5	32	1 = External clock period out of specification 0 = External clock period met specification
4	16	1 = Default skew values are being used or 1 = Memory error could not be corrected during redundancy test 0 = Skew values adjusted and no memory errors
3	8	1 = Search could not place X/O markers 0 = Search placed X/O markers
2	4	1 = Trigger found 0 = Trigger not found
1	2	1 = Run until condition satisified 0 = Run until condition not satisified
0	1	1 = Measurement complete 0 = Measurement did not complete

Part 2

2 Format Menu Commands 2-1
3 Trigger Menu Commands 3-1
4 Waveform Menu Commands 4-1
5 Listing Menu Commands 5-1
6 Compare Menu Commands 6-1
7 Symbols Commands 7-1
8 Data and Setup Commands 8-1

Commands

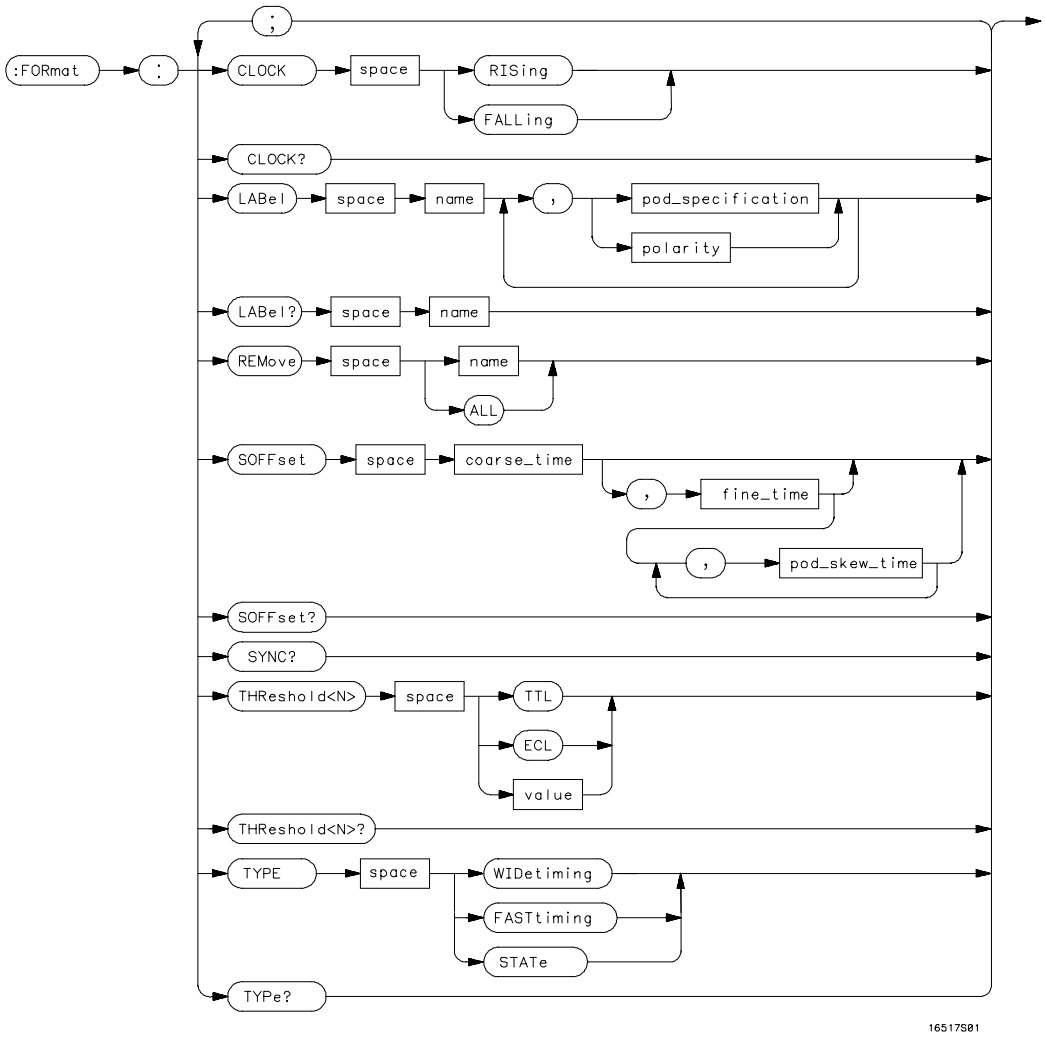
Format Menu Commands

Introduction

The Format menu commands allow you access the FORmat subsystem which contains the following commands:

- CLOCk
- LABel
- REMove
- SOFFset
- SYNC
- THReshold
- TYPE

Figure 2-1



FORMat Subsystem Syntax Diagram

Table 2-1 **FORMat Parameter Values**

Parameter	Values
<N>	{1|2} for one card or {1|2|3|4|5|6|7|8|10} for up to five cards in pairs
name	string of up to 6 alphanumeric characters
polarity	{POSitive|NEGative}
pod_specification	an integer from 0 to 255 or 0 to 15 (depending on type) for a pod (pods are assigned from left to right)
coarse_time	a real number from –5 ns to +5 ns in increments of 200 ps
fine_time	an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns
pod_skew_time	an absolute real number that is within ±500 ps of <fine_time> with a maximum of ±5 ns
value	voltage (real number) –5.00 to +5.00 in 0.01 volt increments

FORMat

Selector

```
:FORMat
```

The FORMat selector is used as part of a compound header to access those settings normally found in the Format menu. It always follows the SELECT(n) command when you first access the module. It must precede any command you wish to send to the FORMat subsystem unless you send combined command messages (see "Combined Commands in the Same Subsystem" in Chapter 1 of the *Agilent Technologies 16500B/16501A Programmer's Guide*).

Example

```
OUTPUT XXX;":FORMAT:CLOCK?"
```

CLOCk (State mode only)

Command `:FORMat:CLOCk <clock_edge>`

The CLOCk command allows you to specify the clock edge on which the state analyzer will be clocked. The external clock for the state analyzer is always on pod 1 of the master card. The clock options are RISing (edge) and FALLing (edge).

`<clock_edge>` {RISing|FALLing}

Example

```
OUTPUT XXX;":FORMAT:CLOCK FALLING"
```

Query `:FORMAT:CLOCK?`

The CLOCk query returns the current clock specification.

Returned Format `[:FORMat:CLOCk] <clock_edge><NL>`

Example

```
OUTPUT XXX;":FORMAT:CLOCK?"
```

LABel

Command

```
:FORMat:LABel <name>[[, <polarity>]
[, <assignment>]...]
```

The LABel command allows you to specify polarity and to assign channels to new or existing labels. If the specified label name does not match an existing label name, a new label will be created.

The order of the pod specification parameters is significant. The first <assignment> listed will match the left most pod in the Format menu. The <assignment> after that assigns channels to the next pod to the right as displayed in the Format menu. For example, sending pod specifications 255 and 15 assigns all eight channels of pod 2 and the lower four channels of pod 1. If multiple boards are installed, the pod specifications in this example will assign channels to pod 2 and pod 1 of top most board. Not including enough pod specifications results in the lower numbered pod being assigned a value of zero (all channels excluded). If you include more pod specifications than there are pods, the extra ones will be ignored. However, an error is reported anytime more pod specifications are listed than the number of available pods.

The polarity can be specified at any point after the label name.

Because pods contain either 4 or 8 channels depending on the mode, the <assignment> value for a pod must be between 0 and 15 (24–1) or 0 and 255 (28–1) respectively. When specifying the pod assignment in binary (base 2), each bit will correspond to a single channel. A "1" in a bit position means the associated channel in that pod is assigned to that pod and bit. A "0" in a bit position means the associated channel in that pod is excluded from the label. For example, assigning #B11110011 is equivalent to entering ****..** from the front panel.

A label can not have a total of more than 32 channels assigned to it.

`<name>` string of up to 6 alphanumeric characters

`<polarity>` {POSitive|NEGative}

`<assignment>` For each pod, an integer from 0 to 255 in STATE or WIDETIMING mode or 0 to 15 in FASTTIMING mode (pods are assigned from left to right)

Examples

```
OUTPUT XXX;":FORMAT:LABEL 'STAT', POSITIVE, 255,15"
OUTPUT XXX;":FORMAT:LABEL 'SIG 1', '#B00001111',
'#B00000000'"
```

Query

```
:FORMat:LABel? <name>
```

The LABel query returns the current specification for the selected (by name) label. If the label does not exist, nothing is returned. Numbers are always returned in decimal format.

Returned Format

```
[:FORMat:LABel] <name>,<polarity>[, <assignment>]...<NL>
```

`<name>` string of up to 6 alphanumeric characters

`<polarity>` {POSitive|NEGative}

Example

```
OUTPUT XXX;":FORMAT:LABEL? 'DATA'"
```

REMove

Command

```
:FORMat:REMove {<name>|ALL}
```

The REMove command allows you to delete all labels or any one label specified by name.

`<name>` string of up to 6 alphanumeric characters

Examples

```
OUTPUT XXX;":FORMAT:REMOVE 'A'"
OUTPUT XXX;":FORMAT:REMOVE ALL"
```

SOFFset (State Mode Only)

Command

```
:FORMat:SOFFset <coarse_time>[,<fine_time>
[, <pod_skew_time1>, ..., <pod_skew_time10>]]
```

The SOFFset command allows you to specify the offset between the external clock and the internal sample clock in the State mode. An error (–211, Legal command but settings conflict) results if the State mode is not selected.

`<coarse_time>` a real number from –5 ns to +5 ns in increments of 200 ps

`<fine_time>` an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns

`<pod_skew_time1...10>` an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns

Examples

```
OUTPUT XXX;":FORMat:SOFFSET 1.2E–9"
OUTPUT XXX;":FORMat:SOFFSET 1.1E–9, 1.3E–9, 1.25E–9, 1.25E-9"
```

Query

```
:FORMat:SOFFset?
```

The SOFFset query returns the current setting for the clock offset. If the SOFFset setting is only set with <coarse_time>, the query returns one time value. If the setting is set with <coarse_time>, <fine_time>, and <pod_skew_time>, the query returns <coarse_time>, <fine_time> followed by a time value for each pod.

Returned Format

```
[:FORMat:SOFFset]  <coarse_time>[,<fine_time>
[, <pod_skew_time1>, ..., <pod_skew_time10>]]<NL>
```

Example

```
OUTPUT XXX;":FORMat:SOFFSET?"
```

SYNC

Query `:FORMat:SYNC?`

The SYNC query instructs the 16517A/18A to synchronize the internal clock to the external clock. It then returns a 1 if the synchronization was successful (the external clock was valid) or a 0 if the synchronization was unsuccessful (the external clock was invalid).

Returned Format `[:FORMat:SYNC] {1|0}<NL>`

Example

```
OUTPUT XXX;":FORMAT:SYNC?"
```

THReshold

Command `:FORMat:THReshold<N> {TTL|ECL|<value>}`

The THReshold command allows you to set the voltage threshold for a given pod to ECL, TTL, or a specific voltage from –5.00 V to +5.00 V in 0.01 volt increments. The order of the pod number <N> is significant in multiple-card systems. Pod number 1 will set the threshold of pod 1 of the card in the lowest numbered slot in the mainframe while pod number 3 will set the threshold of pod 1 of the card in the next higher numbered slot.

`<N>` an integer from 1 to 10 depending on number of cards. Pod number options are {1|2} for one card (two pods) or {1|2|...|10} for up to five cards in pairs.

`<value>` voltage (real number) –5.00 to +5.00 in increments of 0.01 volts

`TTL` default value of +1.5 V

`ECL` default value of –1.3 V

Example

```
OUTPUT XXX;":FORMAT:THRESHOLD1 4.0"
```

Query `:FORMat:THReshold<N>?`

The THReshold query returns the current threshold for a given pod.

Returned Format
```
[:FORmat:THReshold<N>] <value><NL>
```

Example
```
OUTPUT XXX;":FORMAT:THRESHOLD2?"
```

TYPE

Command
```
:FORMat:TYPE <analyzer_type>
```

The TYPE command allows you to specify the acquisition mode of the analyzer. The acquisition modes are STATe, WIDetiming or FASTtiming.

There is one important note about using the TYPE command instead of the front-panel interface. When you change the mode from FASTtiming to WIDetiming from the front panel, you have the option of specifying whether or not to restore bits 4 through 7 to their previous value. When you change the mode from FASTtiming to WIDetiming over the bus, bits 4 through 7 are always restored to their previous value.

`<analyzer_type>` {WIDetiming|FASTtiming|STATe}

Example
```
OUTPUT XXX;":FORMAT:TYPE STATE"
```

Query
```
:FORMat:TYPE?
```

The TYPE query returns the current type or mode (timing).

Returned Format
```
[:FORMat:TYPE] {WIDEtiming|FASTtiming|STATe}<NL>
```

Example
```
OUTPUT XXX;":FORMat:TYPE?"
```

3

Trigger Menu Commands

Introduction

The Trigger menu commands allow access to the TRIGer subsystem and contains the following commands:

- ACQuisition
- ARMedby
- BRANch
- CLEar
- DURation
- EDGE
- FIND
- PATTern
- REName
- SAMPclk
- SEQuence
- SETUPHOLDA
- SETUPHOLDB
- SETUPHOLDC
- SPERiod
- TIMER
- TPOSition

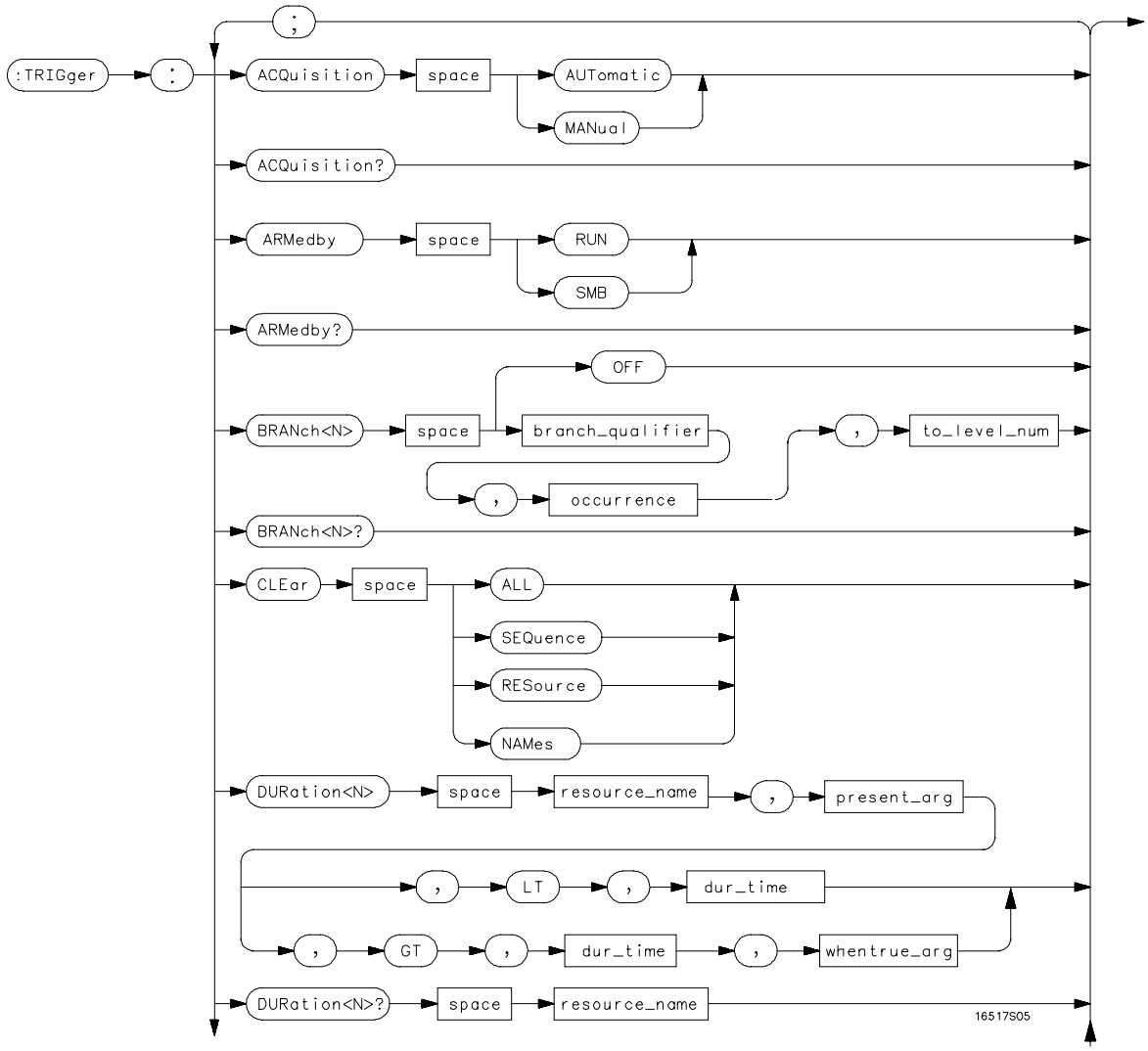
Hint

The trigger macros, except for the setup and hold macros, are not accessible over the bus but can be programmed using the individual user levels contained in each macro. Once you set up a macro from the front panel, you can touch the Modify Trigger field, then the Break Down Macro field to see what user levels to use to program the macro over the bus.

See Also

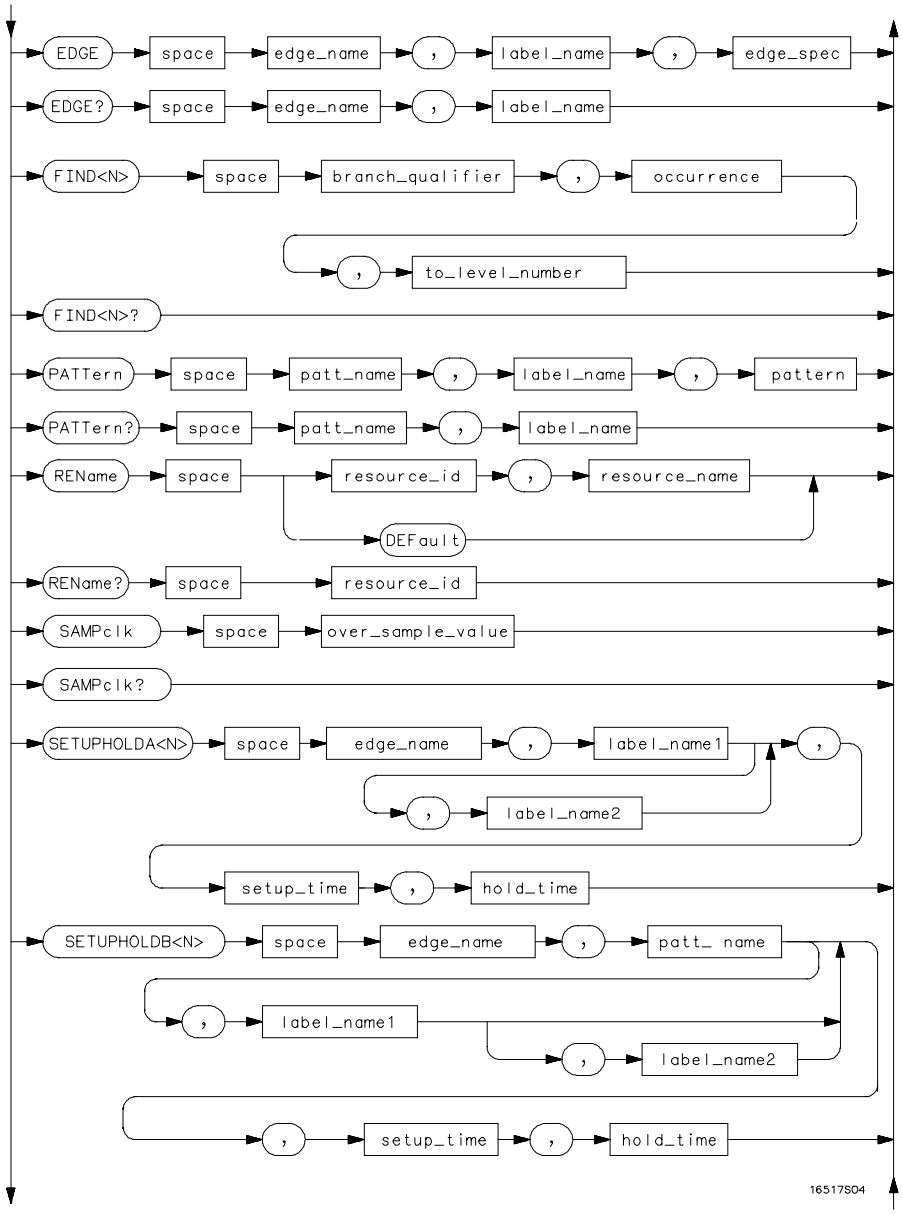
"Modify Trigger Field," in Chapter 4, "Trigger Menu" in the *Agilent Technologies 16517A/18A User's Reference*.

Figure 3-1



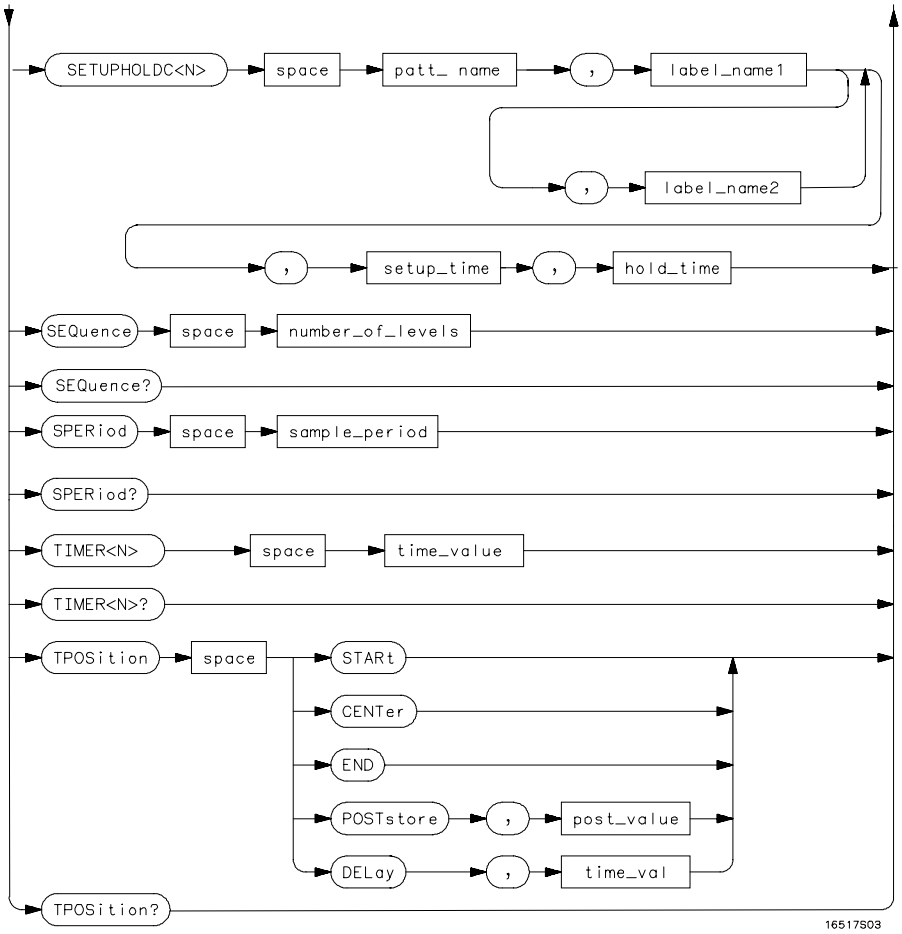
TRIGger Subsystem Syntax Diagram

Figure 3-1 (continued)



TRIGger Subsystem Syntax Diagram (continued)

Figure 3-1 (Continued)



TRIGger Subsystem Syntax Diagram (continued)

Table 3-1 TRIGger Parameter Values

Parameter	Values
branch_qualifier	<qualifier>
occurrence	number from 1 to 16777216
to_level_number	integer from 1 to last level or TRIGger
resource_name	a string of 8 alphanumeric characters
present_arg	{PRESent|ABSent}
dur_time	a real number from 0 ns to 510 ns in 2 ns increments
whentrue_arg	{UNTilexit|UPONEXit|UPONENtry}
edge_name	a string of 8 alphanumeric characters
edge_spec	a string consisting of {R|F|E|.} R, F, and E represents rising, falling, either edge respectively. A period (.) represents a don't care.
proceed_qualifier	<qualifier>
patt_name	a string of 8 alphanumeric characters
label_name	string of up to 6 alphanumeric characters
resource_id	{PATTern1|PATTern2|PATTern3|PATTern4|EDGE1|EDGE2}
pattern	"{#B{0|1|X}...| #Q{0|1|2|3|4|5|6|7|X}...| #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X}...| {0|1|2|3|4|5|6|7|8|9}...}"
over_sample_value	an integer from 1 to 32 in powers of 2
setup_time	a real number from 2 ns to 510 ns in 2 ns increments
hold_time	a real number from 2 ns to 33.554 ms in 2 ns increments
number_of_levels	an integer from 1 to 4
sample_period	a real number from 500 ps to 524.29 µs in WIDetiming mode
time_value	a real number from 2 ns to 33.554 ms
post_value	integer from 1 to 99 representing percentage
time_val	real number from either (2 × sample period) or 16 ns whichever is greater to (1048574 × sample period)
qualifier	see "Qualifier" on page 3-6

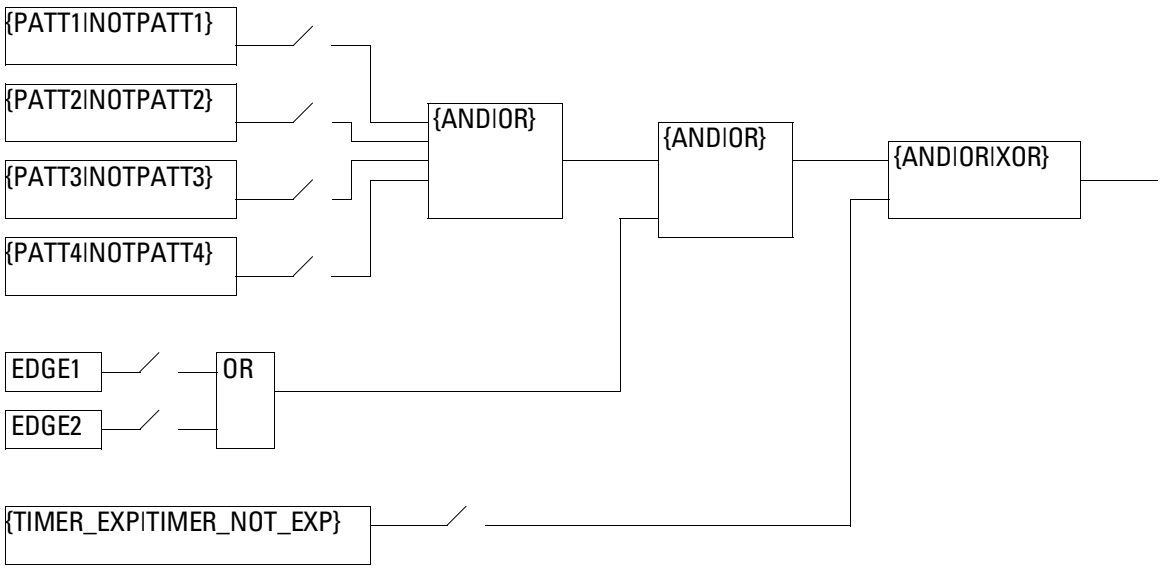
Qualifier

The qualifier for the trigger subsystem can be terms patt1 through patt4, Edge 1 and Edge 2, and Timer. In addition, qualifiers can be the NOT boolean function of the patt1 through patt4 terms and the Timer. The qualifier can also be an expression or combination of expressions as shown below and in figures 3-2 and 3-3. In addition a combination of expressions is shown in "Complex Qualifier," in figure 3-4 on page 3-9.

The following figures show how qualifiers are specified in all commands of the TRIGger subsystem that use <qualifier>.

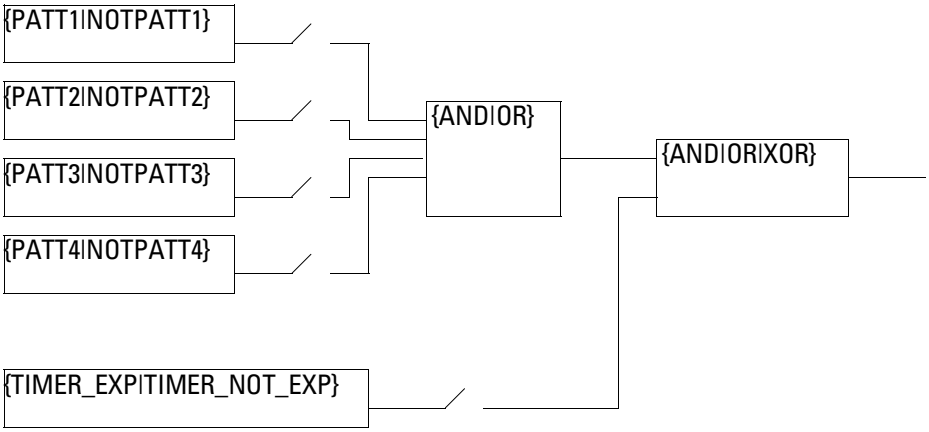
`<qualifier>` {"ANYSTATE"|"NOSTATE"|"<expression>"}

Figure 3-2



Timing <expression>

Figure 3-3



State <expression>

Qualifier Rules

The following rules apply to qualifiers:

- Qualifiers are quoted strings and, therefore, need quotes.
- Expressions are evaluated from left to right.
- Parentheses are used to change the order evaluation and, therefore, are optional.
- An expression must map into the combination logic presented in the combination pop-up menu within the TRIGger menu as shown in figure 3-4.

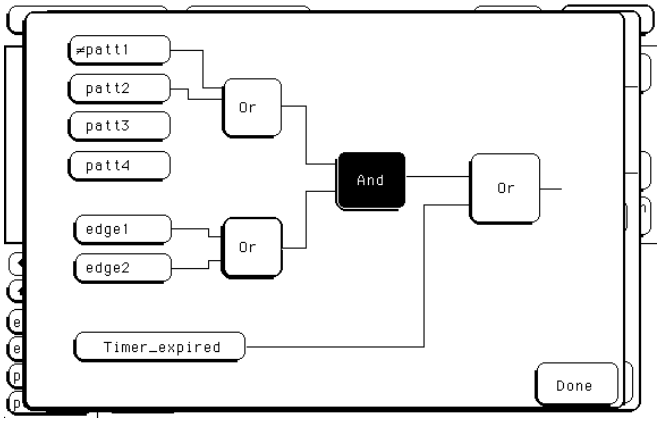
Examples

```
'PATT1'
'( PATT1 OR PATT2 )'
'(( PATT1 OR PATT2 ) AND EDGE2 )'
'(( PATT1 OR PATT2 ) AND EDGE2 AND TIMER_NOT_EXP )'
'(( PATT1 OR PATT2 ) AND ( EDGE1 OR EDGE2)) OR TIMER_EXP'
'TIMER_NOT_EXP AND (( PATT1 OR PATT2 ) OR (EDGE1 OR EDGE2))'
```

The following statements are all correct and have the same meaning. Notice that the conventional rules for precedence are not followed. The expressions are evaluated from left to right.

```
":TRIGGER:BRANCH1 'PATT3 AND PATT4 OR EDGE1 OR EDGE2', 1, 3"
":TRIGGER:BRANCH1 '((PATT3 AND PATT4) OR (EDGE1 OR EDGE2))',
1, 3"
":TRIGGER:BRANCH1 'EDGE1 OR (PATT3 AND PATT4) OR EDGE2',1, 3"
```

Figure 3-4



Complex Qualifier

Figure 3-4 is a front-panel representation of the complex qualifier (NOTPATT1 OR PATT2) AND (EDGE1 OR EDGE2) OR TIMER_EXP).

Example

This example would be used to specify this complex qualifier.

```
OUTPUT XXX;":TRIGGER:BRANCH1 '((PATT1 OR PATT2) AND (EDGE1 OR
EDGE2))', 2, 3"
```

In the first level, the operators you can use are AND, or OR for pattern resources, but only OR is available for the edge resources. Either AND or OR may be used at the second level to join the two groups together. The timer resource can be combined with the pattern and edge resources with AND, OR, or XOR. It is acceptable for a group to consist of a single term. Thus, an expression like (PATT2 AND EDGE1) is legal since the two operands are both simple terms from separate groups.

TRIGger (TRACe)

Selector

```
:TRIGger
```

The TRIGger (TRACe) selector is used as a part of a compound header to access the settings found in the Trigger menu. It always follows the SELECT(n) command when you first access the module. It must precede any command you wish to send to the TRIGger subsystem unless you send combined command messages (see "Combined Commands in the Same Subsystem" in Chapter 1 of the *Agilent Technologies 16500B/16501A Programmer's Guide*).

Although the menu is called Trigger and the primary keyword is TRIGger, the command parser in the 16517A/18A is designed to accept TRACe. This eliminates the need to change programs written for earlier modules that required the keyword TRACe.

Example

```
OUTPUT XXX;":TRIGGER:CLEAR ALL"
```

ACQuisition (WIDetiming Type Only)

Command

```
:TRIGger:ACQuisition {AUTOmatic|MANual}
```

The ACQuisition command allows you to specify the acquisition mode for the timing analyzer. This command results in an error (–211, Legal Command but settings conflict) if the STATe or FASTtiming is selected.

Example

```
OUTPUT XXX;":TRIGGER:ACQUISITION AUTOMATIC"
```

Query

```
:TRIGger:ACQuisition?
```

The ACQuisition query returns the current acquisition mode specified. This query results in an error (–211, Legal Command but settings conflict) if the STATe or FASTtiming is selected.

Returned Format

```
[:TRIGger:ACQuisition] {AUTOmatic|MANual}<NL>
```

Example

```
OUTPUT XXX;":TRIGGER:ACQUISITION?"
```

ARMedby

Command

```
:TRIGger:ARMedby {RUN|SMB}
```

The ARMedby command allows you to specify which source arms the 16517A/18A. The timing analyzer can be armed internally with the RUN option or externally with the SMB option. When the RUN option is specified and when the timing analyzer is not in the intermodule tree, sending RUN immediately arms the analyzer. If the timing analyzer is to be armed by another module in the intermodule tree, it is eventually armed by RUN through the other module. Specifying SMB allows the timing analyzer to be armed by an external source connected to the SMB connector on the rear panel.

Example

```
OUTPUT XXX;":TRIGGER:ARMEDBY RUN"
```

Query

```
:TRIGger:ARMedby?
```

The ARMedby query returns the current arming source.

Returned Format

```
[:TRIGger:ARMedby] {RUN|SMB}<NL>
```

Example

```
OUTPUT XXX;":TRIGGER ARMEDBY?"
```

BRANch

Command

```
:TRIGger:BRANch<N> {OFF|<branch_qualifier>,
<occurrence>, <to_level_number>}
```

The BRANch command defines the secondary branch qualifier, occurrence count, and the sequence level to jump to for a given sequence level. When this branch qualifier is matched, the sequencer will jump to the specified sequence level. When the OFF option is specified, the secondary branch for the given sequence level is turned off.

When an occurrence count of greater than 1 is specified, the occurrence counter for the specified sequence level is assigned to the secondary branch. Therefore, any occurrence count previously specified for the primary branch using the FIND command will be overridden. Also, when you specify an occurrence count greater than 1, any previous use of the timer resource for either branch in this sequence level will be overridden.

If you specify the timer resource to be part of the secondary branch qualifier for this level, any previous setting of an occurrence count in either the primary or the secondary branch will be overridden.

If you specify the timer resource to be part of the secondary branch qualifier, and you specify an occurrence count greater than 1 for the secondary branch, an error (–211, Legal command but settings conflict) results and the command is ignored. For example, ":TRIGGER:BRANCH1 'TIMER_EXP OR PATT1',2,2" is illegal because the timer is specified as part of the qualifier and the specified occurrence count is 2.

> The BRANch command in conjuction with the FIND command allows you to program sequence levels in terms of user-programmable levels only. There is no programming support for macros except SETUP and HOLD. See Hint on page 3-2.

`<N>` integer from 1 to <number_of_levels>

`<to_level_number>` integer from {1 to <number_of_levels>|TRIGGER}

`<occurrence>` integer from 1 to 16777216

`<branch_qualifier>` <qualifier>. See "Qualifier" on page 3-6

Examples

```
OUTPUT XXX;":TRIGGER:BRANCH1 'ANYSTATE', 3, 4"
OUTPUT XXX;":TRIGGER:BRANCH2 'PATT1', 7, 3"
OUTPUT XXX;":TRIGGER:BRANCH3 '((PATT1 OR PATT2) OR
NOTPATT3)', 1, 4"
```

Query Syntax `:TRIGger:BRANch<N>?`

The BRANch query returns the current branch qualifier specification for a given sequence level.

Returned Format `[:TRIGger:BRANch<N>] <branch_qualifier>, <occurrence_count>, <to_level_number><NL>`

Example

```
OUTPUT XXX;":TRIGGER:BRANCH3?"
```

CLEar

Command `:TRIGger:CLEar {All|SEQuence|RESource|NAMes}`

The CLEar command allows you to clear all settings in the Timing Trigger menu and replace them with the default, clear only the sequence levels, clear only the resource term patterns and edges, or clear only the resource names.

Example

```
OUTPUT XXX;":TRIGGER:CLEAR RESOURCE"
```

DURation (Timing mode only)

Commands

```
:TRIGger:DURation<N> <resource_name>,
<present_arg>, LT, <dur_time>

:TRIGger:DURation<N> <resource_name>,
<present_arg>, GT, <dur_time>, <whentrue_arg>
```

The DURation command allows you to specify the pattern duration in terms of the presence or absence of the pattern, duration of the pattern being greater than or less than a specified time, and when the pattern is considered true for a given pattern resource term as it is used in a given sequence level. Refer to Chapter 4, "Trigger Menu" in the *Agilent Technologies 16517A/18A User's Reference* for detailed information on the <whentrue_arg> options.

When you specify the pattern <resource_name>, the 16517A/18A looks for the name given to the resource instead of the default names PATT1 through PATT4. This differs from the Agilent Technologies 16550A and 1660-series logic analyzers. The name given to the pattern <resource_name> is not case sensitive. For example, "A" is equivalent to "a" in a pattern resource name.

This command results in an error (–211, Legal Command but settings conflict) if the State Mode is selected.

<N>	an integer from 1 to <number_of_levels>
<resource_name>	a string of 8 alphanumeric characters naming the pattern resource term
<present_arg>	{PRESent|ABSent}
GT	greater than
LT	less than
<dur_time>	a real number between 0 ns and 510 ns in 2 ns increments
<whentrue_arg>	{UNTilexit|UPONEXit|UPONENtry}

Example

```
OUTPUT XXX;":TRIGGER:DURATION1 'PATT1', PRESENT, GT, 300ns,
UNTILEXIT"
```

Query

:TRIGger:DURation<N>? <resource_name>

The DURation query returns the current pattern duration for the given pattern resource term at the specified sequence level.

Returned Format

[:TRIGger:DURation<N>]<resource_name>, <present_arg>, {GT|LT}, <dur_time>[, <whentrue_arg>]<NL>

Example

```
OUTPUT XXX;":TRIGGER:DURATON<N>? 'PATT1'"
```

EDGE (Timing Mode Only)

Command

```
:TRIGger:EDGE <edge_name>, <label_name>,
<edge_spec>
```

The EDGE command allows you to specify the edge specification for a given edge resource. Edge specifications are strings with a length equal to the number of assigned bits and can contain R (rising), F (falling), or E (either edge). Each command deals with only one label in the given term; therefore, a complete specification could require several commands.

When you specify the edge resource, the 16517A/18A looks for the name given to the resource instead of the default names EDGE1 or EDGE2. This differs from the Agilent Technologies 16550A and 1660-series logic analyzers. The name given to the edge resource is not case sensitive. For example, "EDGE1" is equivalent to "edge1" in pattern resource names.

This command results in an error (–211, Legal Command but settings conflict) if the State Mode is selected.

`<edge_name>` a string of 8 alphanumeric characters naming the edge resource term

`<label_name>` a string of 6 alphanumeric characters

`<edge_spec>` a string containing {R|F|E} with a length equal to the number of assigned bits

Example

```
OUTPUT xxx;":TRIGGER:EDGE 'EDGE1', 'DATA', 'R'"
```

Query

```
:TRIGger:EDGE? <edge_name>, <label_name>
```

The EDGE query returns the current edge specification for the given edge resource and label.

Returned Format

```
[TRIGger:EDGE] <edge_name>, <label_name>, <edge_spec><NL>
```

Example

```
OUTPUT xxx;":TRIGGER:EDGE? 'EDGE1', 'DATA'"
```

FIND

Command

```
:TRIGger:FIND<N> <branch_qualifier>,
<occurrence>, <to_level_number>
```

The FIND command defines the primary branch qualifier, occurrence count, and sequence level to jump to for a given sequence level.

When an occurrence count of greater than 1 is specified, the occurrence counter for the specified sequence level is assigned to the primary branch. Therefore, any occurrence count previously specified for the secondary branch, using the BRANch command, will be overridden. Also, when you specify an occurrence count greater than 1, any previous use of the timer resource for either branch in this sequence level will be overridden.

If you specify the timer resource to be part of the primary branch qualifier for this level, any previous setting of an occurrence count in either the primary or secondary branch will be overridden.

If you specify the timer resource to be part of the primary branch qualifier, and you specify an occurrence count greater than 1 for the primary branch, an error (–211, Legal command but settings conflict) results and the command is ignored. For example, ":TRIGGER:FIND1 'TIMER_EXP OR PATT1',2,2" is illegal because the timer is specified as part of the qualifier and the specified occurrence count is 2.

> The FIND command in conjuction with the BRANch command allows you to program sequence levels in terms of user-programmable levels only. There is no programming support for macros except SETUP and HOLD. See Hint on page 3-2.

`<N>` integer from 1 to the number of existing sequence levels (maximum 4)

`<branch_qualifier>` <qualifier>. See "Qualifier" on page 3-6

> The "NOSTATE" qualifier is not available for the FIND command.

<occurrence_count> an integer from 1 to 16777216

<to_level_number> integer from {1 to <number_of_levels>|TRIGger}

<number_of_levels> integer from 1 to the number of existing sequence levels (maximum 4)

Examples

```
OUTPUT XXX;":TRIGGER:FIND1 'ANYSTATE', 2, 4"
OUTPUT XXX;":TRIGGER:FIND3 '((NOTPATT1 AND NOTPATT2) OR
NOTPATT3)', 1, 3"
```

Query

```
:TRIGger:FIND4?
```

The FIND query returns the current branch qualifier specification for a given sequence level.

Returned Format

```
[:TRIGger:FIND<N>] <branch_qualifier>, <occurrence_count>,
<to_level_number><NL>
```

Example

```
OUTPUT XXX;":TRIGGER:FIND<N>?"
```

PATTern

Command

```
:TRIGger:PATTern <patt_name>, <label_name>,
<pattern>
```

The PATTern command allows you to a specify a pattern recognizer term for a given pattern resource. Each command deals with only one label in the given term; therefore, a complete specification could require several commands. Since a label can contain 32 bits or less, the range of the pattern value will be between $2^{32} - 1$ and 0. When the value of a pattern is expressed in binary, it represents the bit values for the label inside the pattern recognizer term. Since the pattern parameter may contain don't cares and be represented in several bases, it is handled as a string of characters rather than a number.

When you specify the pattern <patt_name>, the 16517A/18A looks for the name given to the pattern instead of the default name PATT1 through PATT4. This differs from the Agilent Technologies 16550A and 1660-series logic analyzers. The name given to the pattern resource is not case sensitive. For example, "PATTERN1" is equivalent to "pattern1" in a pattern resource name.

`<patt_name>` a string of 8 alphanumeric characters naming the pattern resource term

`<label_name>` string of up to 6 alphanumeric characters

`<pattern>` "{#B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Example

```
OUTPUT XXX;":TRIGGER:PATTERN 'ADDR','LABEL1', '#B11100011'"
```

Query `:TRIGger:PATTern? <patt_name>, <label_name>`

The PATTern query returns the current pattern recognizer specification for a given pattern resource.

Returned Format `[:TRIGger:PATTern] <patt_name>, <label_name>, <pattern><NL>`

Example

```
OUTPUT XXX;":TRIGGER:PATTERN? 'ADDR', 'LABEL1'"
```

REName

Command `:TRIGger:REName {<resource_id>, <resource_name>|DEFault}`

The REName command allows you to assign a logical name to a given pattern or edge resource. This logical name is then used when referring to the resource in other commands and queries. The DEFault option returns all pattern and edge logical names to the default values.

`<resource_id>` {PATTern1|PATTern2|PATTern3|PATTern4|EDGE1|EDGE2}

`<resource_name>` a string of up to 8 alphanumeric characters

Example

```
OUTPUT XXX;":TRIGGER:RENAME PATTERN1, 'JMP0'"
```

Query `:TRIGger:REName? <resource_id>`

The REName query returns the current logical name assigned to a given resource.

Returned Format `[:TRIgger:REName] <resource_id>, <resource_name><NL>`

Example

```
OUTPUT XXX;":TRIGGER:RENAME? PATTERN2"
```

SAMPclk (State Mode Only)

Command `:TRIGger:SAMPclk <over_sample_value>`

The SAMPclk command allows you to specify the current oversampling value from 1 to 32 samples per clock depending on the current value of the external clock period. The <over_sample_value> times the sample clock cannot exceed 2 GHz. For example, at a clock speed of 100 MHz the maximum <over_sample_value> is 16 ($100 \times 16 = 1.6$ GHz). Additionally, at the slowest clock rate (20 MHz), the maximum <over_sample_value> can be 32 (32×20 MHz = 640 MHz). This command results in an error (–211, Legal command but settings conflict) if the State mode is not selected.

`<over_sample_value>` an integer from 1 to 32 in powers of 2

Example

```
OUTPUT XXX;":TRIGGER:SAMPLCK 16"
```

Query `:TRIGger:SAMPclk?`

The SAMPclk query returns the current value of oversampling.

Returned Format `[:TRIGger:SAMPclk] <over_sample_value><NL>`

Example

```
OUTPUT XXX;":TRIGGER:SAMPLCLK?"
```

SEQuence

Command `:TRIGger:SEQuence <number_of_levels>`

The SEQuence command defines the timing analyzer trace sequence. First, it deletes the current trace sequence. Then, it inserts the number of levels specified, with default settings. The number of user levels can be between 1 and 4.

`<number_of_levels>` integer from 1 to 4

Example

```
OUTPUT XXX;":TRIGGER:SEQUENCE 4"
```

Query `:TRIGger:SEQuence?`

The SEQuence query returns the number of user levels in the current sequence specification.

Returned Format `[:TRIGger:SEQuence] <number_of_levels><NL>`

Example

```
OUTPUT XXX;":TRIGGER:SEQUENCE?"
```

The SEQuence command allows you to program sequence levels in terms of user-programmable levels only. There is no programming support for macros except SETUP and HOLD. See Hint on page 3-2.

SETUPHOLDA (Timing Mode Only)

Command

```
:TRIGger:SETUPHOLDA<N> <edge_name>, <label_name1>
[, <label_name2>], <setup_time>, <hold_time>
```

The SETUPHOLDA command allows you to specify the setup and hold macro number 1. This macro finds setup or hold time violations on the channels specified by <label_name1> or <label_name2> relative to the clock edge specified by <edge_name>. The sequence level specified in this command must exist prior to sending this command. Make sure you send the SEQuence command with the appropriate number of levels just prior to sending the SETUPHOLDA command. When the edge is specified in setup and hold, two edge resources are used; therefore, the other edge resource is not available for any sequence level. This command results in an error (–211, Legal Command but settings conflict) if the State Mode is selected.

`<N>` an integer from 1 to <number_of_levels> for the sequence level in which the setup and hold edge is to be specified.

`<label_name>` a string of up to 6 alphanumeric characters

`<edge_name>` a string of up to 8 alphanumeric characters containing the name (given by the REName command) for edge1 or edge2

`<set_up_time>` a real number from 2 ns to 510 ns in 2 ns increments

`<hold_time>` a real number from 2 ns to 33.554 ms in 2 ns increments

Examples

```
OUTPUT XXX;":TRIGGER:SETUPHOLDA1 'EDGE1', 'ADDR', 6E-9, 4E-9"
OUTPUT XXX;":TRIGGER:SETUPHOLDA1 'EDGE1', 'ADDR1', 'ADDR2',
6E-9, 4E-9"
```

SETUPHOLDB (Timing Mode Only)

Command

```
:TRIGger:SETUPHOLDB<N> <edge_name>, <patt_name>,
<label_name1> [, <label_name2>], <setup_time>,
<hold_time>
```

The SETUPHOLDB command allows you to specify the setup and hold macro number 2. This macro finds setup or hold time violations on the channels specified by <label_name1> or <label_name2> clocked by an edge within a valid pattern. The sequence level specified in this command must exist prior to sending this command. Make sure you send the SEQuence command with the appropriate number of levels just prior to sending the SETUPHOLDB command. When the edge is specified in setup and hold, two edge resources are used; therefore, the other edge resource is not available for any sequence level. This command results in an error (–211, Legal Command but settings conflict) if the State Mode is selected.

`<N>` an integer from 1 to <number_of_levels> for the sequence level in which the setup and hold edge is to be specified.

`<label_name>` a string of up to 6 alphanumeric characters

`<edge_name>` a string of up to 8 alphanumeric characters containing the name (given by the REName command) or edge1 or edge2

`<patt_name>` a string of up to 8 alphanumeric characters containing the name (given by the REName command) or patt1 through patt4

`<set_up_time>` a real number from 2 ns to 510 ns in 2 ns increments

`<hold_time>` a real number from 2 ns to 33.554 ms in 2 ns increments

Examples

```
OUTPUT XXX;":TRIGGER:SETUPHOLDB1 'EDGE1', 'patt1', 'ADDR',
6E-9, 4E-9"
OUTPUT XXX;":TRIGGER:SETUPHOLDB1 'EDGE1', 'patt1', 'ADDR1',
'ADDR2', 6E-9, 4E-9"
```

SETUPHOLDC (Timing Mode Only)

Command

```
:TRIGger:SETUPHOLDC<N> <patt_name>, <label_name1>
[, <label_name2>], <setup_time>, <hold_time>
```

The SETUPHOLDC command allows you to specify the setup and hold macro number 3. This macro finds setup or hold time violations on the channels specified by <label_name1> or <label_name2> clocked by a pattern/pulse. The sequence level specified in this command must exist prior to sending this command. Make sure you send the SEQuence command with the appropriate number of levels just prior to sending the SETUPHOLDC command.

This command results in an error (–211, Legal Command but settings conflict) if the State Mode is selected.

`<N>` an integer from 1 to <number_of_levels> for the sequence level in which the setup and hold edge is to be specified.

`<label_name>` a string of up to 6 alphanumeric characters

`<patt_name>` a string of up to 8 alphanumeric characters containing the name (given by the REName command) or patt1 through patt4

`<set_up_time>` a real number from 2 ns to 510 ns in 2 ns increments

`<hold_time>` a real number from 2 ns to 33.554 ms in 2 ns increments

Examples

```
OUTPUT XXX;":TRIGGER:SETUPHOLDC1 'patt1', 'ADDR', 6E-9, 4E-9"
OUTPUT XXX;":TRIGGER:SETUPHOLDC1 'patt1', 'ADDR1', 'ADDR2',
6E-9, 4E-9"
```

SPERiod

Command `:TRIGger:SPERiod <sample_period>`

The SPERiod command allows you to set the sample period for the next acquisition of the timing analyzer when in the wide timing mode. If the fast timing mode is selected, the sample period is always 250 ps. When this command is sent, the acquisition mode is automatically set to manual (see "TRIGger:ACQuisition" on page 3-11). The value you specify for this command is rounded to the nearest allowable setting.

This command results in an error (–211, Legal command but settings conflict) if the analyzer is in the State mode or the Fast Timing mode.

`<sample_period>` real number from 500 ps to 65.536 µs

Example

```
OUTPUT XXX;":TRIGGER:SPERIOD 64E–9"
```

Query `:TRIGger:SPERiod?`

The SPERiod query returns the sample period of the current acquisition. If there is no valid data, the query returns 9.9E37.

Returned Format `[:TRIGger:SPERiod] <sample_period><NL>`

Example

```
OUTPUT XXX;":TRIGGER:SPERIOD?"
```

TIMER

Command `:TRIGger:TIMER<N> <time_value>`

The TIMER command allows you to set the time value the timer resource will be set to upon entering a given sequence level. The limits of the timer are 2 ns to 33.554 ms in 2 ns increments in the timing modes. The increment value varies with the external clock in the State mode. The <time_value> will be rounded to the nearest clock period. Sending this command will automatically override any settings for the occurrence counter in either the primary or the secondary branch.

`<N>` an integer from 1 to the number of existing sequence levels (maximum of 4)

`<time_value>` real number from 2 ns to 33.554 ms

Example

```
OUTPUT XXX;":TRIGGER:TIMER1 100E-6"
```

Query `:TRIGger:TIMER<N>?`

The TIMER query returns the current time value at the sequence level specified by <N>.

Returned Format `[:TRIGger:TIMER<N>] <time_value><NL>`

Example

```
OUTPUT XXX;":TRIGGER:TIMER1?"
```

TPOSition

Command

```
:TRIGger:TPOSition {STARt|CENTer|END|DELay,
<time_val>|POSTstore, <post_value>}
```

The TPOSition (trigger position) command allows you to set the trigger at the start, center, end or at any position in the trace using delay or poststore. Delay is specified as a real number representing the time between the trigger and the first acquired sample. In the state mode, if the external clock period is greater than 16 ns, the poststore clock runs at the external clock frequency and the delay limits are 0 ns to (1048574 × external clock period). If the external clock period is less than 16 ns, the external clock is divided down by a power of 2 such that the period of the poststore clock is a minimum of 16 ns.

Otherwise, in the timing mode, with the sequencer running at 500 MHz, the poststore clock runs at the same rate as the sample clock or

sample period×2^{x} where x is great enough so that the minimum poststore clock period is 16 ns.

Poststore is defined as 1 to 99 percent with a poststore of 99 percent being the same as start position and a poststore 1 percent being the same as an end trace.

If the analyzer is a timing analyzer, this command will automatically set the acquisition mode to MANUAL (see "TRIGger:ACQuisition" on page 3-11).

`<time_val>` real number from 0 to (1048574 × external clock period), divided down sample period, or divided down external clock period.

`<post_value>` integer from 1 to 99 representing percentage of poststore.

Examples

```
OUTPUT XXX;":TRIGGER:TPOSITION END"
OUTPUT XXX;":TRIGGER:TPOSITION POSTstore, 75"
```

Query

```
:TRIGger:TPOSition?
```

The TPOSition query returns the current trigger position setting.

Returned Format

```
[:TRIGger:TPOSition] {STARt|CENTer|END|DELay,
<time_val>|POSTstore, <post_value>}<NL>
```

Example

```
OUTPUT XXX;":TRIGGER:TPOSITION?"
```

4

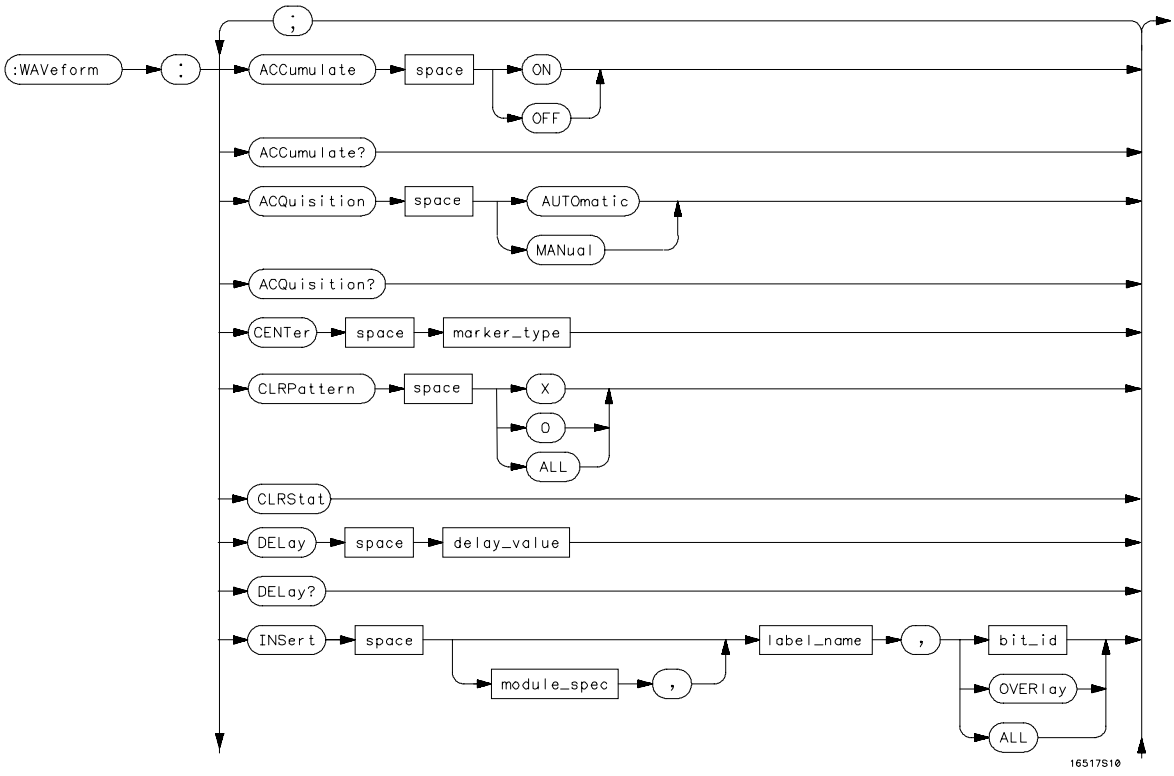
Waveform Menu Commands

Introduction

The WAVeform subsystem contains the commands available for the Timing Waveforms menu in the 16517A/18A. These commands are:

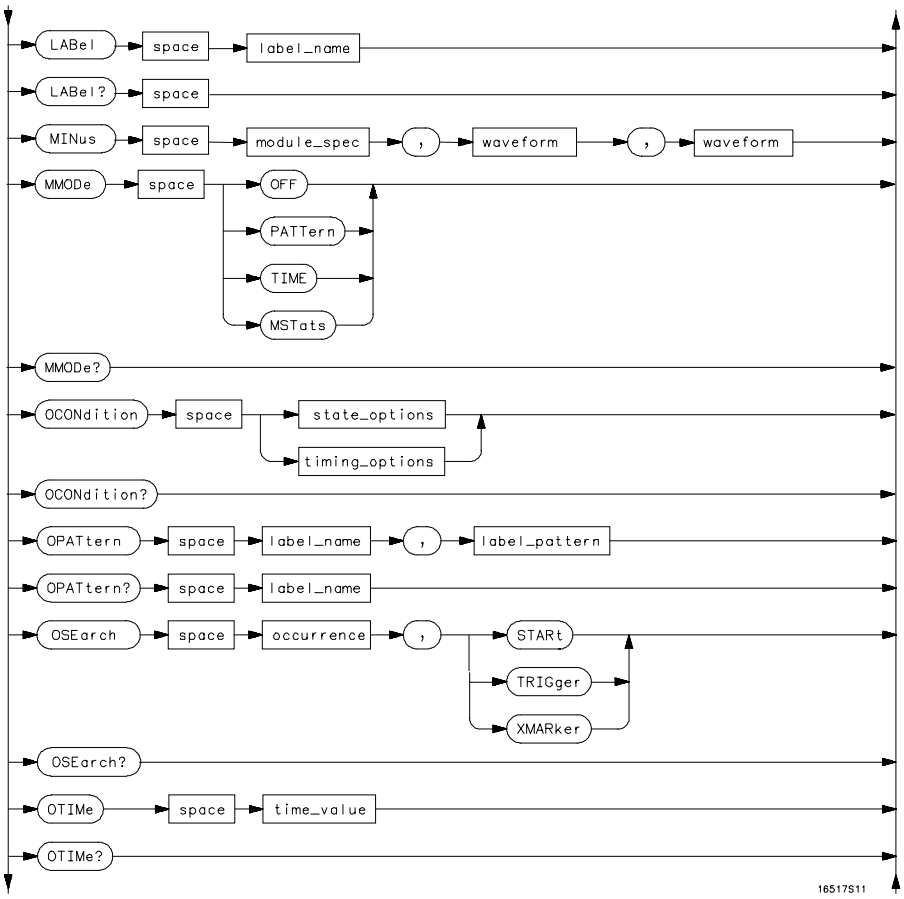
- ACCumulate
- ACQuisition
- CENTer
- CLRPattern
- CLRStat
- DELay
- INSert
- LABel
- MINus
- MMODe
- OCONdition
- OPATtern
- OSEarch
- OTIMe
- OVERlay
- PLUS
- RANGe
- REMove
- RUNTil
- SAMPclk
- SIZE
- SOFFset
- SPERiod
- TAVerage
- TMAXimum
- TMINimum
- TPOSition
- VRUNs
- XCONdition
- XOTime
- XPATtern
- XSEarch
- XTIMe

Figure 4-1



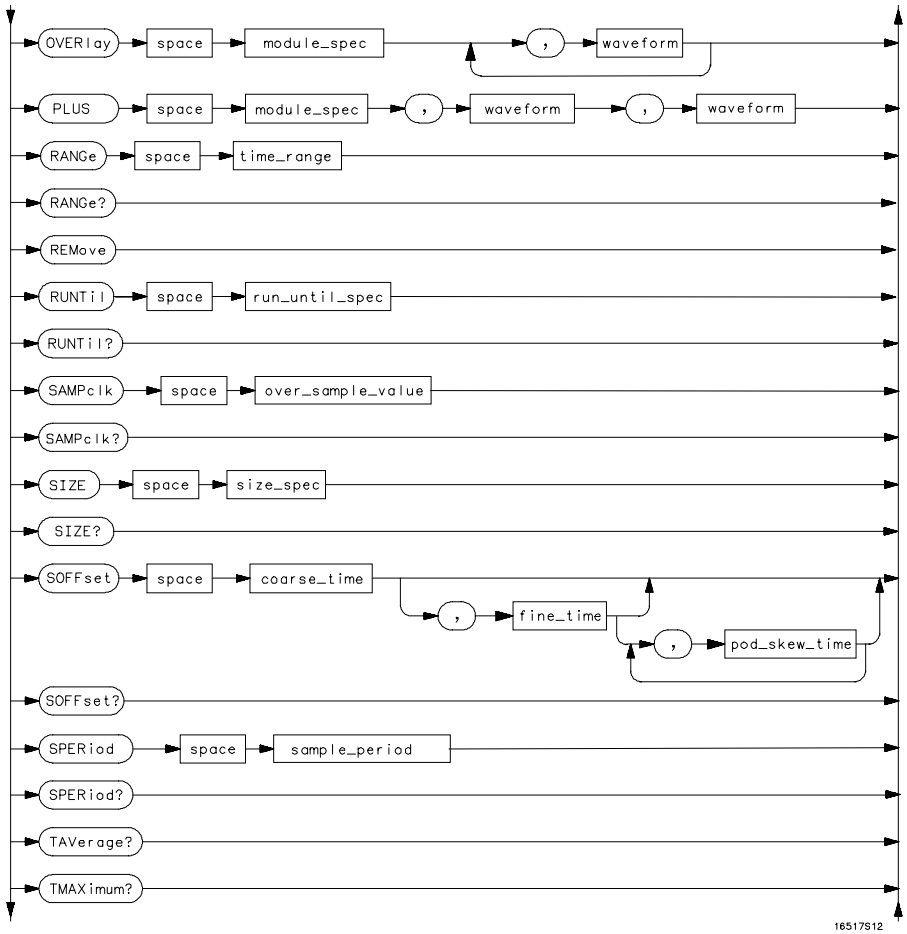
WAVeform Subsystem Syntax Diagram

Figure 4-1 (continued)



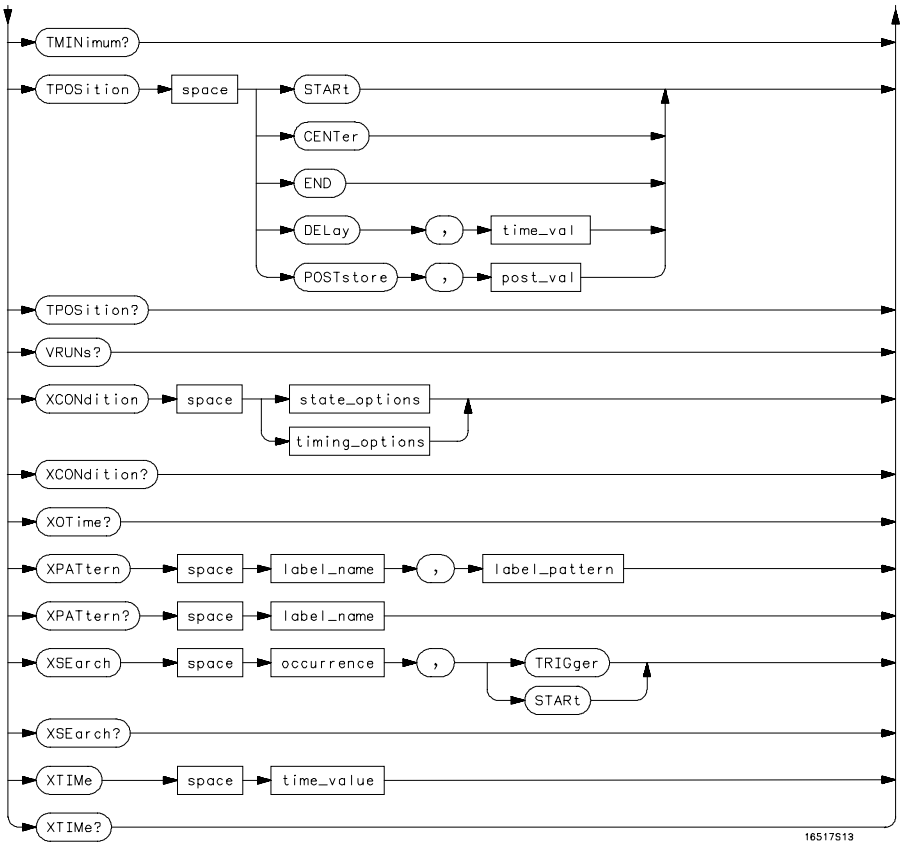
WAVeform Subsystem Syntax Diagram (continued)

Figure 4-1 (continued)



WAVeform Subsystem Syntax Diagram (continued)

Figure 4-1 (continued)



WAVeform Subsystem Syntax Diagram (continued)

Table 4-1 WAVeform Parameter Values

Parameter	Value
marker_type	{X|O|XO|TRIGger}
delay_value	real number between –2500 s and +2500 s
module_spec	{1|2|3|4|5|6|7|8|9|10}
label_name	string of up to 6 alphanumeric characters
bit_id	integer from 0 to 31
waveform	string containing <acquisition_spec>{1|2}
acquisition_spec	{A|B|C|D|E|F|G|H|I|J} (slot where acquisition card is located)
state_options	{ENTering|LEAVing|CLOCk}
timing_options	{ENTering|LEAVing}
label_pattern	"{#B{0|1|X} . . . | #Q{0|1|2|3|4|5|6|7|X}...| #H{0|1|2|3|4|5|6|7|8|9|A|C|D|E|F|X}...| {0|1|2|3|4|5|6|7|8|9|X}...}"
occurrence	integer from –131071 to +131071
time_value	real number
module_number	slot number in which the time base card is installed
time_range	real number between 2.5 ns and 500 s
run_until_spec	{OFF|LT, <value>|GT, <value>|INRange ,<value>, <value>|OUTRange, <value>, <value>|EQUal|NEQual}
GT	greater than
LT	less than
value	real number

Table 4-1 (Continued) **WAVeform Parameter Values**

Parameter	Value
over_sample_value	an integer from 1 to 32 in powers of 2
size_spec	{BESTfit|SMALl|MEDium|LARGe}
coarse_time	a real number from –5 ns to +5 ns in increments of 200 ps
Fine_time	an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns
pod_skew_time	an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns
sample_period	a real number from 250 ps to 524.29 µs depending on mode
time_val	real number from either (2 × sample period) or 16 ns whichever is greater to (1048574 × sample period)
post_val	an integer from 1 to 99 representing percentage of data after the trigger

WAVeform

Selector

```
:WAVeform
```

The WAVeform selector is used as part of a compound header to access the commands found in the Timing Waveform menu. It always follows the SELECT(n) command when you first access the module. It must precede any command you wish to send to the WAVeform subsystem unless you send combined command messages (see "Combined Commands in the Same Subsystem" in Chapter 1 of the *Agilent Technologies 16500B/16501A Programmer's Guide*).

Example

```
OUTPUT XXX;":WAVEFORM:DELAY 100E-9"
```

ACCumulate

Command `:WAVeform:ACCumulate <setting>`

The ACCumulate command allows you to control whether the Waveform display gets erased between each individual run or whether subsequent waveforms are allowed to be displayed over the previous waveforms.

`<setting>` {0|OFF} or {1|ON}

Example

```
OUTPUT XXX;":WAVEFORM:ACCUMULATE ON"
```

Query `:WAVeform:ACCumulate?`

The ACCumulate query returns the current setting. The query always returns the setting as the characters o (off) or 1 (on).

Returned Format `[:WAVeform:ACCumulate] {0|1}<NL>`

Example

```
OUTPUT XXX;":WAVEFORM:ACCUMULATE?"
```

ACQuisition (WIDetiming Type Only)

Command `:WAVeform:ACQuisition {AUTOmatic|MANual}`

The ACQuisition command allows you to specify the acquisition mode for the timing analyzer. This command results in an error (–211, Legal Command but settings conflict) if the STATe or FASTtiming is selected.

Example

```
OUTPUT XXX;":WAVEFORM:ACQUISITION AUTOMATIC"
```

Query `:TRIGger:ACQuisition?`

The ACQuisition query returns the current acquisition mode specified. This query results in an error (–211, Legal Command but settings conflict) if the STATe or FASTtiming is selected.

Returned Format `[:WAVeform:ACQuisition] {AUTOmatic|MANual}<NL>`

Example

```
OUTPUT XXX;":WAVEFORM:ACQUISITION?"ACCumulate
```

CENTer

Command `:WAVeform:CENTer <marker_type>`

The CENTer command allows you to center the waveform display about the specified markers. Centering about both the X and O markers will change the s/div and delay settings.

`<marker_type>` {X|O|XO|TRIGger}

Example

```
OUTPUT XXX;":WAVEFORM:CENTER X"
```

CLRPattern

Command

```
:WAVeform:CLRPattern {X|O|ALL}
```

The CLRPattern command allows you to clear the patterns in the Specify Patterns menu, which are the patterns used for pattern searches.

Example

```
OUTPUT XXX;":WAVEFORM:CLRPATTERN ALL"
```

CLRStat

Command

```
:WAVeform:CLRStat
```

The CLRStat command allows you to clear the waveform statistics without having to stop and restart the acquisition.

Example

```
OUTPUT XXX;":WAVEFORM:CLRSTAT"
```

DELay

Command

```
:WAVeform:DELay <delay_value>
```

The DELay command specifies the amount of time between the timing trigger and the horizontal center of the the timing waveform display. The allowable range for delay values is –2500 s to +2500 s. If the acquisition mode is Automatic, as delay becomes large in an absolute sense, the sample rate is adjusted so that data will be acquired in the time window of interest.

`<delay_value>` real number between –2500 s and +2500 s

Example

```
OUTPUT XXX;":WAVEFORM:DELAY 100E-6"
```

Query

```
:WAVeform:DELay?
```

The DELay query returns the current time offset (delay) value from the trigger to the center of the waveform display.

Returned Format

```
[:WAVeform:DELay] <delay_value><NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:DELAY?"
```

INSert

Command

```
:WAVeform:INSert [<module_spec>,]
<label_name>[, {<bit_id>|OVERlay|ALL}]
```

The INSert command allows you to add waveforms to the waveform display. Waveforms are added from top to bottom on the screen. When 96 waveforms are present, inserting an additional waveform replaces the last waveform. Bit numbers are zero based, so a label with 8 bits is referenced as bits 0 through 7. Specifying OVERlay causes a composite waveform display of all bits or channels for the specified label. If you do not specify the third parameter, ALL is assumed.

`<module_spec>` {1|2|3|4|5|6|7|8|9|10}

`<label_name>` string of up to 6 alphanumeric characters

`<bit_id>` integer from 0 to 31

Example

```
OUTPUT XXX;":WAVEFORM:INSERT 1, 'WAVE',10"
```

LABel

Command `:WAVeform:LABel <label_name>`

The LABel command in the Waveform menu allows you to change which label's pattern is displayed in the Pattern-at-Marker field when in the time marker mode. If the <label_name> you send does not match an existing label, an error (200, Label not found) results and the command is ignored.

`<label_name>` a string of up to 6 alphanumeric characters representing an existing label name (see FORmat:LABel in chapter 2)

Example

```
OUTPUT XXX;":WAVEFORM:LABEL DATA"
```

Query `:WAVeform:LABel?`

The LABel query returns the name of the label currently displayed in the Pattern-at-Marker field.

Returned Format `[:WAVeform:LABel] <label_name><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:LABEL?"
```

MINus

Command

```
:WAVeform:MINus <module_spec>, <waveform>,
<waveform>
```

The MINus command inserts time-correlated A–B (A minus B) oscilloscope waveforms on the screen. The first parameter is the module specifier where the oscilloscope module resides, where 1 through 10 refers to slots A through J. The next two parameters specify which waveforms will be subtracted from each other. The second waveform is subtracted from the first waveform.

> MINus is only available for oscilloscope waveforms.

`<module_spec>` {1|2|3|4|5|6|7|8|9|10}

`<waveform>` string containing <acquisition_spec>{1|2}

`<acquisition_spec>` {A|B|C|D|E|F|G|H|I|J} (slot where acquisition card is located)

Example

```
OUTPUT XXX; ":WAVEFORM:MINUS 2, 'A1', 'A2'"
```

MMODe

Command

```
:WAVeform:MMODe {OFF|PATTern|TIME|MSTats}
```

The MMODe (Marker Mode) command selects the mode that controls marker placement and the display of the marker readouts. When PATTern is selected, the markers will be placed on patterns. When TIME is selected, the markers move to specific time values. In MSTats, the markers are placed on patterns, but the readouts will be time statistics.

Example

```
OUTPUT XXX; ":WAVEFORM:MMODE TIME"
```

Query

```
:WAVeform:MMODe?
```

The MMODe query returns the current marker mode.

Returned Format

```
[:WAVeform:MMODe] {OFF|PATTern|TIME|MSTats}<NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:MMODE?"
```

OCONdition

Command

```
:WAVeform:OCONdition {<state_options>|
<timing_options>}
```

The OCONdition command specifies where the O marker is placed. The O marker can be placed on the entry or exit point of the OPATtern when in the PATTern or STATistics marker modes. An additional option in state mode is CLOCk which places the O marker on externally clocked states only, ignoring the oversampled states.

`<state_options>` {ENTering|LEAVing|CLOCk}

`<timing_options>` {ENTering|LEAVing}

Examples

```
OUTPUT XXX; ":WAVEFORM:OCONDITION ENTERING"
OUTPUT XXX; ":WAVEFORM:OCONDITION CLOCK"
```

Query

```
:WAVeform:OCONdition?
```

The OCONdition query returns the current setting.

Returned Format

```
[:WAVeform:OCONdition] {<state_options>|<timing_options>}<NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:OCONDITION?"
```

OPATtern

Command `:WAVeform:OPATtern <label_name>, <label_pattern>`

The OPATtern command allows you to construct a pattern recognizer term for the O marker which is then used with the OSEarch criteria and OCONdition when placing the marker on patterns. Since this command deals with only one label at a time, a complete specification could require several invocations.

When the value of a pattern is expressed in binary, it represents the bit values for the label inside the pattern recognizer term. Whatever base is used, the value must be between 0 and $(2^{32} - 1)$, since a label may not have more than 32 bits. Because the <label_pattern> parameter may contain don't cares, it is handled as a string of characters rather than a number.

`<label_name>` string of up to 6 alphanumeric characters

`<label_pattern>` "{#B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Example

```
OUTPUT XXX; ":WAVEFORM:OPATTERN 'A', '511'"
```

Query `:WAVeform:OPATtern? <label_name>`

The OPATtern query, in pattern marker mode, returns the pattern specification for a given label name. In the time marker mode, the query returns the pattern under the O marker for a given label. If the O marker is not placed on valid data, don't cares (X) are returned.

Returned Format `[:WAVeform:OPATtern] <label_name>, <label_pattern><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:OPATTERN? 'A'"
```

OSEarch

Command `:WAVeform:OSEarch <occurrence>, <origin>`

The OSEarch command defines the search criteria for the O marker which are then used with the associated OPATtern recognizer specification and the OCONdition when placing the marker on patterns. The origin parameter tells the marker to begin a search at the start of acquired data, at the trigger, or at the X marker. The occurrence parameter determines which occurrence of the OPATtern recognizer specification, relative to the origin, the marker actually searches for. An occurrence of 0 places a marker on the selected origin. With a negative occurrence, the marker searches before the origin. With a positive occurrence, the marker searches after the origin.

`<origin>` {STARt|TRIGger|XMARker}

`<occurrence>` integer from –131071 to +131071

Example

```
OUTPUT XXX; ":WAVEFORM:OSEARCH +10,TRIGGER"
```

Query `:WAVeform:OSEarch?`

The OSEarch query returns the search criteria for the O marker.

Returned Format `[:WAVeform:OSEarch] <occurrence>,<origin><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:OSEARCH?"
```

OTIMe

Command `:WAVeform:OTIMe <time_value>`

The OTIMe command positions the O marker in time when the marker mode is TIME. If data is not valid, the command performs no action.

`<time_value>` real number

Example

```
OUTPUT XXX; ":WAVEFORM:OTIME 30.0E-6"
```

Query `:WAVeform:OTIMe?`

The OTIMe query returns the O marker position in time. If acquired data is not valid, the query returns 9.9E37.

Returned Format `[:WAVeform:OTIMe] <time_value><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:OTIME?"
```

OVERlay

Command

```
:WAVeform:OVERlay <module_spec>, <waveform>
[, <waveform>]...
```

The OVERlay command overlays two or more oscilloscope waveforms and adds the resultant waveform to the current waveforms display. The first parameter of the command syntax specifies which slot contains the oscilloscope time base card. The next parameters are the labels of the waveforms that are to be overlaid.

`<module_spec>` {1|2|3|4|5|6|7|8|9|10}

`<waveform>` string containing <acquisition_spec>{1|2}

`<acquisition_spec>` {A|B|C|D|E|F|G|H|I|J} (slot where acquisition card is located)

Example

```
OUTPUT XXX;":WAVEFORM:OVERLAY 4, 'C1','C2'"
```

PLUS

Command

```
:WAVeform:PLUS <module_spec>, <waveform>,
<waveform>
```

The PLUS command inserts time-correlated A+B oscilloscope waveforms on the screen. The first parameter is the module specifier where the oscilloscope module resides, where 1 through 10 refers to slots A through J. The next two parameters specify which waveforms will be added to each other.

PLUS is only available for oscilloscope waveforms.

`<module_spec>` {1|2|3|4|5|6|7|8|9|10}

`<waveform>` string containing <acquisition_spec>{1|2}

`<acquisition_spec>` {A|B|C|D|E|F|G|H|I|J} (slot where acquisition card is located)

Example

```
OUTPUT XXX; ":WAVEFORM:PLUS 2, 'A1', 'A2'"
```

RANGe

Command `:WAVeform:RANGe <time_range>`

The RANGe command specifies the full-screen time in the timing waveform menu. It is equivalent to ten times the seconds-per-division setting on the display. The allowable values for RANGe are from 2.5 ns to 500 s.

`<time_range>` real number between 2.5 ns and 500 s

Example

```
OUTPUT XXX;":WAVEFORM:RANGE 100E-9"
```

Query `:WAVeform:RANGe?`

The RANGe query returns the current full-screen time.

Returned Format `[:WAVeform:RANGe] <time_value><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:RANGE?"
```

REMove

Command `:WAVeform:REMove`

The REMove command deletes all waveforms from the display.

Example

```
OUTPUT XXX;":WAVEFORM:REMOVE"
```

RUNTil

Command `:WAVeform:RUNTil <run_until_spec>`

The RUNTil (run until) command allows you to define a stop condition when the run mode is repetitive. Specifying OFF causes the analyzer to make runs until either the display's Stop field is touched or the STOP command is issued.

There are four conditions based on the time between the X and O markers. These four conditions are as follows:

- Less Than (LT) a specified time value.
- Greater Than (GT) a specified time value.
- In the range (INRange) between two time values.
- Out of the range (OUTRange) between two time values.

End points for the INRange and OUTRange should be at least 250 ps apart, since this is the minimum time at which data is sampled.

When Compare is allocated to this module, there are two conditions that are based on a comparison of the acquired state data and the compare reference image. The analyzer will run until one of the following conditions is true:

- Every channel of every label has the same value (EQUal), unless unmasked.
- Any channel of any label that is not masked has a different value (NEQual).

`<run_until_spec>` {OFF|LT, <value>|GT, <value>|INRange, <value>,<value>|OUTRange, <value>,<value>|EQUal|NEQual}

`<value>` real number

Examples

```
OUTPUT XXX;":WAVEFORM:RUNTIL GT, 800.0E-6"
OUTPUT XXX;":WAVEFORM:RUNTIL INRANGE, 4.5E-9, 5.5E-9"
```

Query `:WAVeform:RUNTil?`

The RUNTil query returns the current stop criteria.

Returned Format `[:WAVeform:RUNTil] <run_until_spec><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:RUNTIL?"
```

SAMPclk (State Mode Only)

Command `:WAVEFORM:SAMPclk <over_sample_value>`

The SAMPclk command allows you to specify the current oversampling value from 1 to 32 samples per clock depending on the current value of the external clock period. The <over_sample_value> times the sample clock cannot exceed 2 GHz. For example, at a clock speed of 100 MHz the maximum <over_sample_value> is 16 (100 × 16 = 1.6 GHz). Additionally, at the slowest clock rate (20 MHz), the maximum <over_sample_value> can be 32 (32 × 20 MHz = 640 MHz). This command results in an error (–211, Legal command but settings conflict) if the State mode is not selected.

`<over_sample_value>` an integer from 1 to 32 in powers of 2

Example

```
OUTPUT XXX;":WAVEFORM:SAMPLCK 16"
```

Query `:WAVeform:SAMPclk?`

The SAMPclk query returns the current value of oversampling.

Returned Format `[:WAVEFORM:SAMPclk]<over_sample_value><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:SAMPLCLK?"
```

SIZE

Command `:WAVeform:SIZE <size_spec>`

The SIZE command allows you to specify the waveform size for this module.

`<size_spec>` {BESTfit|SMALl|MEDium|LARGe}

Example

```
OUTPUT XXX;":WAVEFORM:SIZE BESTFIT"
```

Query `:WAVeform:SIZE?`

The SIZE query returns the current setting for this module.

Returned Format `[:WAVeform:SIZE] <size_spec><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:SIZE?"
```

SOFFset (State Mode Only)

Command

```
:WAVeform:SOFFset <coarse_time>[,<fine_time>
[, <pod_skew_time1>, ..., <pod_skew_time10>]]
```

The SOFFset command allows you to specify the offset between the external clock and the internal sample clock in the State mode. An error (–211, Legal command but settings conflict) results if the State mode is not selected.

`<coarse_time>` a real number from –5 ns to +5 ns in increments of 200 ps

`<fine_time>` an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns

`<pod_skew_time1...10>` an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns

Examples

```
OUTPUT XXX;":WAVEFORM:SOFFSET –1.2E–9"
OUTPUT XXX;":WAVEFORM:SOFFSET 1.1E–9, 1.3E–9, 1.25E–9,
1.25E-9"
```

Query

```
:WAVeform:SOFFset?
```

The SOFFset query returns the current setting for the clock offset. If the SOFFset setting is only set with <coarse_time>, the query returns one time value. If the setting is set with <coarse_time>, <fine_time>, and <pod_skew_time>, the query returns <coarse_time>, <fine_time> followed by a time value for each pod.

Returned Format

```
[:WAVeform:SOFFset]  <coarse_time>[,<fine_time>,
<pod_skew_time1>, ...,<pod_skew_time10>]<NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:SOFFSET?"
```

SPERiod

Command `:WAVeform:SPERiod <sample_period>`

The SPERiod command allows you to set the sample period for the next acquisition of the timing analyzer when in the wide timing mode. If the fast timing mode is selected, the sample period is always 250 ps. When this command is sent, the acquisition mode is automatically set to manual (see TRIGger:ACQuisition in chapter 3). The value you specify for this command is rounded to the nearest allowable setting.

This command results in an error (–211, Legal command but settings conflict) if the analyzer is in the state mode or the fast timing mode.

`<sample_period>` real number from 500 ps to 65.536 µs

Example

```
OUTPUT XXX;":WAVEFORM:SPERIOD 50E–9"
```

Query `:WAVeform:SPERiod?`

The SPERiod query returns the sample period of the current acquisition. If there is no valid data, the query returns 9.9E37.

Returned Format `[:WAVeform:SPERiod] <sample_period><NL>`

Example

```
OUTPUT XXX;":WAVEFORM:SPERIOD?"
```

TAVerage

Query `:WAVeform:TAVerage?`

The TAVerage query returns the value of the average time between the X and O markers when the marker mode is MSTats. If there is no valid acquired data, the query returns 9.9E37.

Returned Format `[:WAVeform:TAVerage] <value><NL>`

`<value>` real number

Example

```
OUTPUT XXX;":WAVEFORM:TAVERAGE?"
```

TMAXimum

Query `:WAVeform:TMAXimum?`

The TMAXimum query returns the value of the maximum time between the X and O markers when the marker mode is MSTats. If there is no valid acquired data, the query returns 9.9E37.

Returned Format `[:WAVeform:TMAXimum] <value><NL>`

`<value>` real number

Example

```
OUTPUT XXX;":WAVEFORM:TMAXIMUM?"
```

TMINimum

Query `:WAVeform:TMINimum?`

The TMINimum query returns the value of the minimum time between the X and O markers when the marker mode is MSTats. If there is no valid acquired data, the query returns 9.9E37.

Returned Format `[:WAVeform:TMINimum] <value><NL>`

`<value>` real number

Example

```
OUTPUT XXX;":WAVEFORM:TMINIMUM?"
```

TPOSition

Command `:WAVeform:TPOSition {STARt|CENTer|END|DELay, <time_val>|POSTstore, <post_value}`

The TPOSition (trigger position) command allows you to set the trigger at the start, center, end or at any position in the trace using delay or poststore. Delay is specified as a real number representing the time between the trigger and the first acquired sample. In the state mode, if the external clock period is greater than 16 ns, the poststore clock runs at the external clock frequency and the delay limits are 0 ns to (1048574 × external clock period). If the external clock period is less than 16 ns, the external clock is divided down by a power of 2 such that the period of the poststore clock is a minimum of 16 ns.

Otherwise, in the timing mode with the sequencer running at 500 MHz, the poststore clock runs at the same rate as the sample clock or *sample period*×2^x where x is great enough so that the minimum poststore clock period is 16 ns.

Poststore is defined as 1 to 99 percent with a poststore of 99 percent being the same as start position and a poststore 1 percent being the same as an end trace.

If the analyzer is a timing analyzer, this command will automatically set the acquisition mode to MANUAL (see "TRIGger:ACQuisition" on page 3-11).

`<time_val>` real number from 0 to (1048574 × external clock period) or divided down sample period, or divided down external clock period.

`<post_value>` integer from 1 to 99 representing percentage of poststore.

Example

```
OUTPUT XXX;":WAVEFORM:TPOSITION CENTER"
```

Query

```
:WAVeform:TPOSition?
```

The TPOSition query returns the current trigger position setting.

Returned Format

```
[:WAVeform:TPOSition] {STARt|CENTer|END|DELay,
<time_val>|POSTstore, <post_value>}<NL>
```

Example

```
OUTPUT XXX;":WAVeform:TPOSITION?"
```

VRUNs

Query

```
:WAVeform:VRUNs?
```

The VRUNs query returns the number of valid runs and total number of runs made when the marker mode is MSTats. Valid runs are those where the pattern search for both the X and O markers was successful, resulting in valid delta time measurements.

Returned Format

```
[:WAVeform:VRUNs] <valid_runs>, <total_runs><NL>
```

`<valid_runs>` zero or positive integer

`<total_runs>` zero or positive integer

Example

```
OUTPUT XXX;":WAVEFORM:VRUNS?"
```

XCONdition

Command

```
:WAVeform:XCONdition {<state_options>|
<timing_options>}
```

The XCONdition command specifies where the X marker is placed. The X marker can be placed on the entry or exit point of the XPATtern when in the PATTern or STATistics marker modes. An additional option in state mode is CLOCk, which places the X marker on externally clocked states only, ignoring the oversampled states.

`<state_options>` {ENTering|LEAVing|CLOCk}

`<timing_options>` {ENTering|LEAVing}

Examples

```
OUTPUT XXX; ":WAVEFORM:XCONDITION ENTERING"
OUTPUT XXX; ":WAVEFORM:XCONDITION CLOCK"
```

Query

```
:WAVeform:XCONdition?
```

The XCONdition query returns the current setting.

Returned Format

```
[:WAVeform:XCONdition] {<state_options>|<timing_options>}<NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:XCONDITION?"
```

XOTime

Query `:WAVeform:XOTime?`

The XOTime query returns the time from the X marker to the O marker. If the acquired data is not valid, the query returns 9.9E37.

Returned Format `[:WAVeform:XOTime] <value><NL>`

`<value>` real number

Example

```
OUTPUT XXX;":WAVEFORM:XOTIME?"
```

XPATtern

Command `:WAVeform:XPATtern <label_name>, <label_pattern>`

The XPATtern command allows you to construct a pattern recognizer term for the X marker which is then used with the XSEarch criteria and XCONdition when placing the marker on patterns. Since this command deals with only one label at a time, a complete specification could require several invocations.

When the value of a pattern is expressed in binary, it represents the bit values for the label inside the pattern recognizer term. Whatever base is used, the value must be between 0 and $2^{32} - 1$, since a label may not have more than 32 bits. Because the <label_pattern> parameter may contain don't cares, it is handled as a string of characters rather than a number.

`<label_name>` string of up to 6 alphanumeric characters

`<label_pattern>` "{#B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Example

```
OUTPUT XXX; ":WAVEFORM:XPATTERN 'A','511'"
```

Query

```
:WAVeform:XPATtern? <label_name>
```

The XPATtern query, in pattern marker mode, returns the pattern specification for a given label name. In the time marker mode, the query returns the pattern under the X marker for a given label. If the X marker is not placed on valid data, don't cares (X) are returned.

Returned Format

```
[:WAVeform:XPATtern] <label_name>, <label_pattern><NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:XPATTERN? 'A'"
```

XSEarch

Command

```
:WAVeform:XSEarch <occurrence>, <origin>
```

The XSEarch command defines the search criteria for the X marker which are then used with the associated XPATtern recognizer specification and the XCONdition when placing the marker on patterns. The origin parameter tells the marker to begin a search either at the start of acquired data or at the trigger. The occurrence parameter determines which occurrence of the XPATtern recognizer specification, relative to the origin, the marker actually searches for. An occurrence of 0 (zero) places a marker on the origin. With a negative occurrence, the marker searches before the origin. With a positive occurrence, the marker searches after the origin.

`<origin>` {TRIGger|STARt}

`<occurrence>` integer from –131071 to +131071

Example

```
OUTPUT XXX; ":WAVEFORM:XSEARCH +10,TRIGGER"
```

Query

```
:WAVeform:XSEarch?
```

The XSEarch query returns the search criteria for the X marker.

Returned Format

```
[:WAVeform:XSEarch] <occurrence>,<origin><NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:XSEARCH?"
```

XTIMe

Command

```
:WAVeform:XTIMe <time_value>
```

The XTIMe command positions the X marker in time when the marker mode is TIME. If the acquired data is not valid, the command performs no action.

`<time_value>` real number

Example

```
OUTPUT XXX; ":WAVEFORM:XTIME 40.0E-6"
```

Query

```
:WAVeform:XTIMe?
```

The XTIMe query returns the X marker position in time. If the acquired data is not valid, the query returns 9.9E37.

Returned Format

```
[:WAVeform:XTIMe] <time_value><NL>
```

Example

```
OUTPUT XXX;":WAVEFORM:XTIME?"
```

5

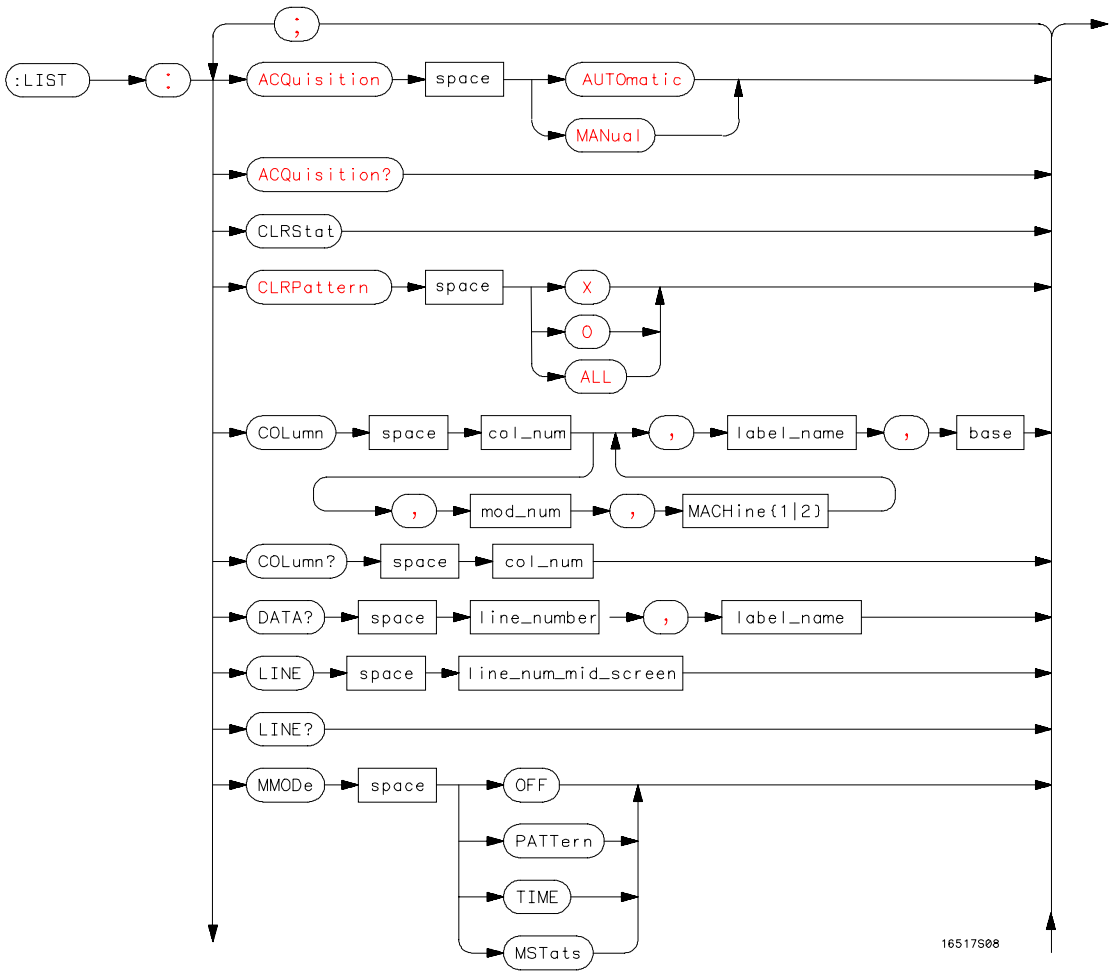
List Menu Commands

Introduction

The LIST menu commands allow access to the LIST subsystem and contains the following commands:

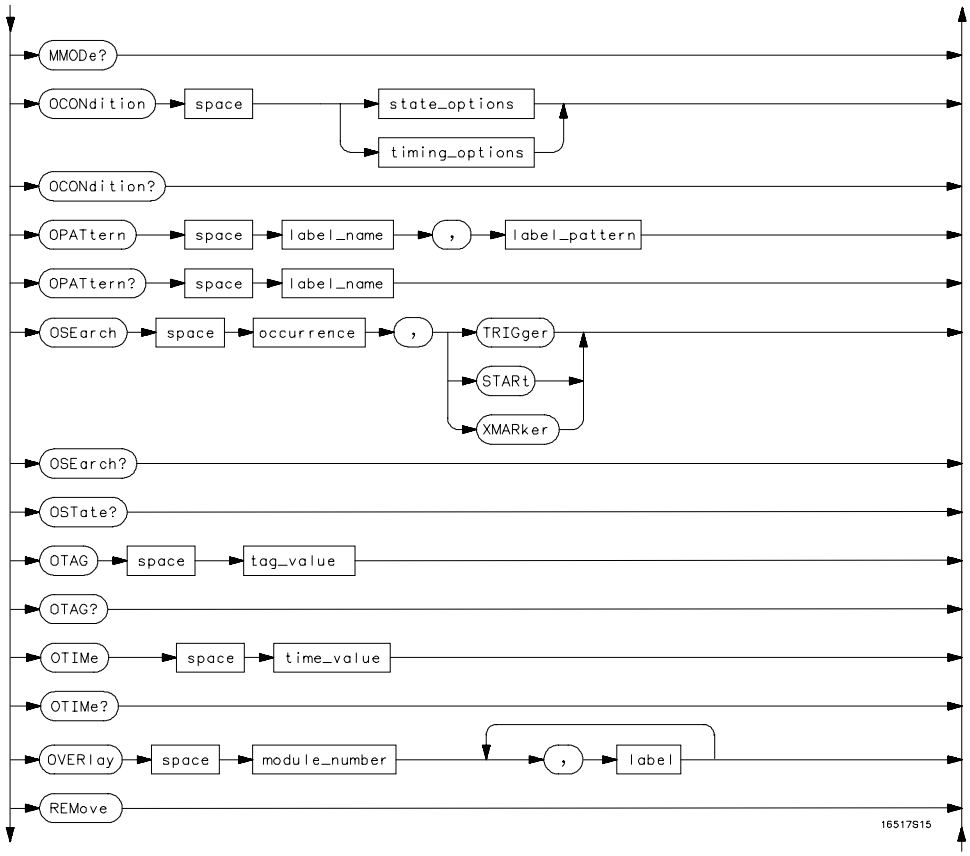
- ACQuisition
- CLRPattern
- CLRStat
- COLumn
- DATA
- LINE
- MMODe
- OCONdition
- OPATtern
- OSEarch
- OSTate
- OTAG
- OTIMe
- OVERlay
- REMove
- RUNTil
- SAMPclk
- SHOW
- SOFFset
- SPERiod
- TAVerage
- TMAXimum
- TMINimum
- TPOSition
- VRUNs
- XCONdition
- XOTag
- XOTime
- XPATtern
- XSEarch
- XSTate
- XTAG
- XTIMe

Figure 5-1



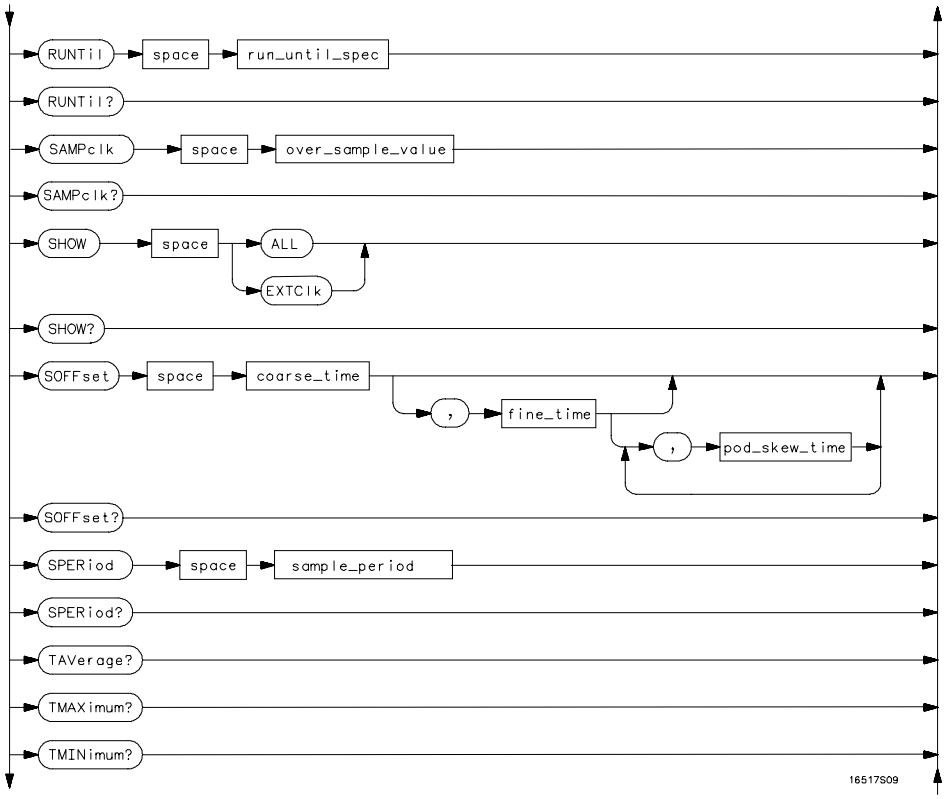
LIST Subsystem Syntax Diagram

Figure 5-1 (continued)



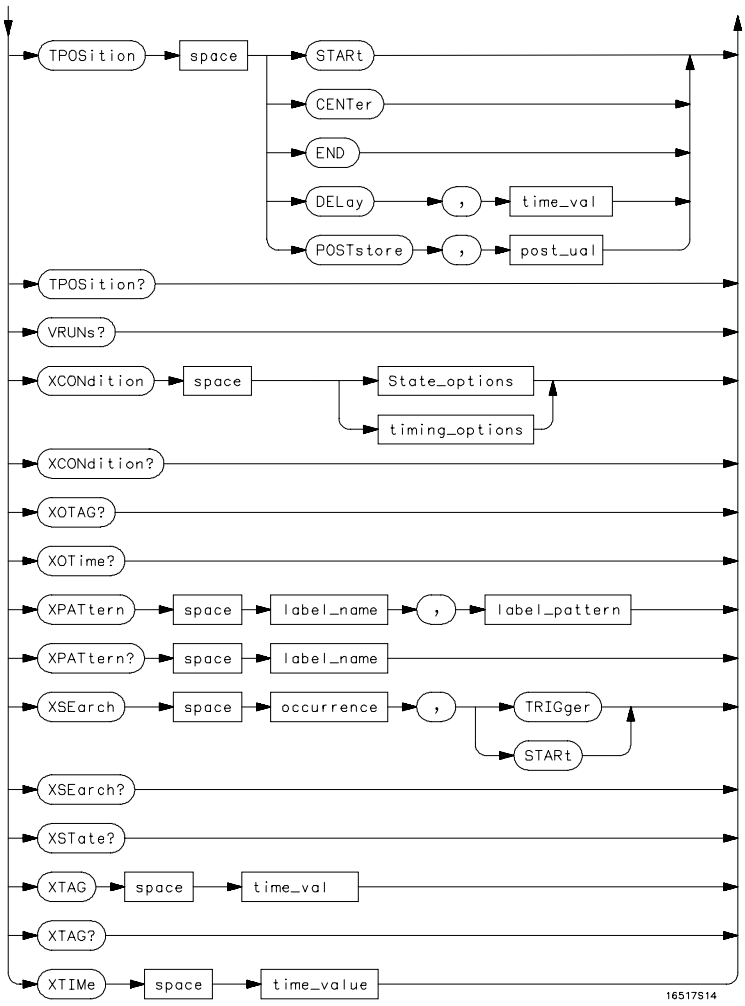
LIST Subsystem Syntax Diagram (continued)

Figure 5-1 (continued)



LIST Subsystem Syntax Diagram (continued)

Figure 5-1 (continued)



LIST Subsystem Syntax Diagram (continued)

Table 5-1 LIST Parameter Values

Parameter	Values
col_num	integer from 1 to 61
label_name	string of up to 6 alphanumeric characters
base	{BINary|HEXadecimal|OCTal|DECimal|TWOS| ASCii|SYMBol} for labels or {ABSolute|RELative} for time tags
module_number	{1|2|3|4|5|6|7|8|9|10}
line_number	integer from –131071 to +131071
line_num_mid_screen	integer from –131071 to +131071
state_options	{ENTering|LEAVing|CLOCk}
timing_options	{ENTering|LEAVing}
label_pattern	"{#B{0|1|X} . . . | #Q{0|1|2|3|4|5|6|7|X} . . . | #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . | {0|1|2|3|4|5|6|7|8|9} . . . }"
occurrence	integer from –131071 to +131071
time_value	real number
run_until_spec	{OFF|LT, <value>|GT, <value>|INRange, <value>, <value>|OUTRange, <value>,<value> |EQUal|NEQual}
GT	greater than
LT	less than
value	real number
coarse_time	a real number from –5 ns to +5 ns in increments of 200 ps
fine_time	an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns
pod_skew_time	an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns
sample_period	a real number from 250 ps to 524.29 µs depending on mode
time_val	a real number from either (2 × sample period) or 16 ns whichever is greater to (1048574 × sample period)
post_val	an integer from 1 to 99 representing percentage

LIST

Selector

`:LIST`

The LIST selector is used as part of a compound header to access those commands normally found in the Timing Listing menu. It always follows the SELECT(n) command when you first access the module. It must precede any command you wish to send to the LIST subsystem unless you send combined command messages (see "Combined Commands in the Same Subsystem" in Chapter 1 of the *Agilent Technologies 16500B/16501A Programmer's Guide*).

Example

```
OUTPUT XXX;":LIST:LINE 256"
```

ACQuisition (Timing Mode Only)

Command

```
:LIST:ACQuisition {AUTOmatic|MANual}
```

The ACQuisition command allows you to specify the acquisition mode for the timing analyzer. This command results in an error (–211, Legal Command but settings conflict) if the STATe or FASTtiming is selected.

Example

```
OUTPUT XXX;":LIST:ACQUISITION AUTOMATIC"
```

Query

```
:LIST:ACQuisition?
```

The ACQuisition query returns the current acquisition mode specified. This query results in an error (–211, Legal Command but settings conflict) if the STATe or FASTtiming is selected.

Returned Format

```
[:LIST:ACQuisition] {AUTOmatic|MANual}<NL>
```

Example

```
OUTPUT XXX;":LIST:ACQUISITION?"
```

CLRPattern

Command `:LIST:CLRPattern {X|O|ALL}`

The CLRPattern command allows you to clear the patterns in the Specify Patterns menu, which are the patterns used for pattern searches.

Example

```
OUTPUT XXX;":LIST:CLRPATTERN O"
```

CLRStat

Command `:LIST:CLRStat`

The CLRStat command allows you to clear the waveform statistics without having to stop and restart the acquisition.

Example

```
OUTPUT XXX;":LIST:CLRSTAT"
```

COLumn

Command

```
:LIST:COLumn <col_num>[, <module_num>,
MACHine{1|2}], <label_name>, <base>
```

The COLumn command allows you to configure the list display by assigning a label name and base to one of the 61 vertical columns in the menu. A column number of 1 refers to the left most column. When a label is assigned to a column it replaces the original label in that column.

When the label name is "TIME," the TIME column is assumed and the next parameter must specify RELative or ABSolute.

`<col_num>` integer from 1 to 61

`<module_num>` {1|2|3|4|5|6|7|8|9|10}

`<label_name>` a string of up to 6 alphanumeric characters

`<base>` {BINary|HEXadecimal|OCTal|DECimal|TWOS|ASCii|SYMBol} for labels
or
{ABSolute|RELative} for time

Example

```
OUTPUT XXX;":LIST:COLUMN 4,1,'A',HEX"
```

Query

```
:LIST:COLumn? <col_num>
```

The COLumn query returns the column number, module number, label name, and base for the specified column.

Returned Format

```
[:LIST:COLumn] <col_num>,<module_num>, MACHine{1|2},
<label_name>, <base><NL>
```

Example

```
OUTPUT XXX;":LIST:COLUMN? 4"
```

DATA

Query `:LIST:DATA? <line_number>, <label_name>`

The DATA query returns the value at a specified line number for a given label. The format will be the same as the one shown in the Listing display.

Returned Format `[:LIST:DATA] <line_number>, <label_name>, <label_pattern><NL>`

`<line_number>` integer from –131071 to +131071

`<label_name>` string of up to 6 alphanumeric characters

`<label_pattern>` "{#B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Example

```
OUTPUT XXX;":LIST:DATA? 512, 'Lab1'"
```

LINE

Command `:LIST:LINE <line_num_mid_screen>`

The LINE command allows you to scroll the listing vertically. The command specifies the state line number relative to the trigger that the analyzer highlights at the center of the screen.

`<line_num_mid_screen>` integer from –131071to +131071

Example

```
OUTPUT XXX;":LIST:LINE 0"
```

Query `:LIST:LINE?`

The LINE query returns the line number for the state currently in the box at the center of the screen.

Returned Format `[:LIST:LINE] <line_num_mid_screen><NL>`

Example

```
OUTPUT XXX;":LIST:LINE?"
```

MMODe

Command `:LIST :MMODe{OFF|PATTern|TIME|MSTats}`

The MMODe (Marker Mode) command selects the mode that controls marker placement and the display of the marker readouts. When PATTern is selected, the markers will be placed on patterns. When TIME is selected, the markers move to specific time values. In MSTats, the markers are placed on patterns, but the readouts will be time statistics.

Example

```
OUTPUT XXX; ":LIST:MMODE TIME"
```

Query `:LIST:MMODe?`

The MMODe query returns the current marker mode.

Returned Format `[:LIST:MMODe] {OFF|PATTern|TIME|MSTats}<NL>`

Example

```
OUTPUT XXX;":LIST:MMODE?"
```

OCONdition

Command

```
:LIST:OCONdition
{<state_options>|<timing_options>}
```

The OCONdition command specifies where the O marker is placed. The O marker can be placed on the entry or exit point of the OPATtern when in the PATTern or STATistics marker modes. An additional option in the oversampling state mode is CLOCk which places the O marker on externally clocked states only, ignoring the oversampled states.

`<state_options>` {ENTering|LEAVing|CLOCk}

`<timing_options>` {ENTering|LEAVing}

Examples

```
OUTPUT XXX; ":LIST:OCONDITION ENTERING"
OUTPUT XXX; ":LIST:OCONDITION CLOCK"
```

Query

```
:LIST:OCONdition?
```

The OCONdition query returns the current setting.

Returned Format

```
[:LIST:OCONdition] {<state_options>|<timing_options>}<NL>
```

Example

```
OUTPUT XXX;":LIST:OCONDITION?"
```

OPATtern

Command `:LIST:OPATtern <label_name>, <label_pattern>`

The OPATtern command allows you to construct a pattern recognizer term for the O marker which is then used with the OSEarch criteria and OCONdition when placing the marker on patterns. Since this command deals with only one label at a time, a complete specification could require several invocations.

When the value of a pattern is expressed in binary, it represents the bit values for the label inside the pattern recognizer term. In whatever base is used, the value must be between 0 and $(2^{32} - 1)$, since a label may not have more than 32 bits. Because the <label_pattern> parameter may contain don't cares, it is handled as a string of characters rather than a number.

`<label_name>` string of up to 6 alphanumeric characters

`<label_pattern>` "{#B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Example

```
OUTPUT XXX; ":LIST:OPATTERN 'A', '511'"
```

Query `:LIST:OPATtern? <label_name>`

The OPATtern query, in pattern marker mode, returns the O pattern marker specification for a given label name. In the time marker mode, the query returns the pattern under the O marker for a given label. If the O marker is not placed on valid data, don't cares (X) are returned.

Returned Format `[:LIST:OPATtern] <label_name>, <label_pattern><NL>`

Example

```
OUTPUT XXX;":LIST:OPATTERN? 'A'"
```

OSEarch

Command `:LIST:OSEarch <occurrence>, <origin>`

The OSEarch command defines the search criteria for the O marker which is then used with the associated OPATtern recognizer specification and the OCONdition when moving markers on patterns. The origin parameter tells the marker to begin a search at the start of acquired data, at the trigger, or at the X marker. An occurrence of 0 places a marker on the selected origin. With a negative occurrence, the marker searches before the origin. With a positive occurrence, the marker searches after the origin.

`<origin>` {STARt|TRIGger|XMARker}

`<occurrence>` integer from –131071 to +131071

Example

```
OUTPUT XXX; ":LIST:OSEARCH +10,TRIGGER"
```

Query `:LIST:OSEarch?`

The OSEarch query returns the search criteria for the O marker.

Returned Format `[:LIST:OSEarch] <occurrence>,<origin><NL>`

Example

```
OUTPUT XXX;":LIST:OSEARCH?"
```

OSTate

Query `:LIST:OSTate?`

The OSTate query returns the line number in the listing where the O marker resides (–131071 to +131071). If data is not valid, the query returns 500000.

Returned Format `[:LIST:OSTate] <state_num><NL>`

`<state_num>` an integer from –131071 to +131071, or 500000

Example

```
OUTPUT XXX;":LIST:OSTATE?"
```

OTAG

Command `:LIST:OTAG <time_value>`

The OTAG command specifies the time tag value on which the O Marker should be placed. If there is no valid acquired data, no action is performed.

`<time_value>` real number

Example

```
:OUTPUT XXX;":LIST:OTAG 40.0E–6"
```

Query `:LIST:OTAG?`

The OTAG query returns the O Marker position in time. If the acquired data is not valid, the query returns 9.9E37.

Returned Format `[:LIST:OTAG] <time_value><NL>`

Example

```
OUTPUT XXX;":LIST:OTAG?"
```

OTIMe

Command `:LIST:OTIMe <time_value>`

The OTIMe command positions the O marker in time when the marker mode is TIME. If data is not valid, the command performs no action.

`<time_value>` real number

Example

```
OUTPUT XXX; ":LIST:OTIME 30.0E–6"
```

Query `:LIST:OTIMe?`

The OTIMe query returns the O marker position in time. If data is not valid, the query returns 9.9E37.

Returned Format `[:LIST:OTIMe] <time_value><NL>`

Example

```
OUTPUT XXX;":LIST:OTIME?"
```

OVERlay

Command

```
:LIST:OVERlay <col_num>, <module_num>,
MACHine{1|2}, <label_name>
```

The OVERlay command allows you to add time-correlated labels from other modules or machines to the listing.

`<col_num>` integer from 1 to 61

`<module_num>` {1|2|3|4|5|6|7|8|9|10}

`<label_name>` a string of up to 6 alphanumeric characters

Example

```
OUTPUT XXX;":WAVEFORM:OVERLAY 4, 3, MACHINE2, 'DATA'"
```

REMove

Command

```
:LIST:REMove
```

The REMove command removes all labels, except the leftmost label and the Time label, from the listing menu.

Example

```
OUTPUT XXX;":LIST:REMOVE"
```

RUNTil

Command

```
:LIST:RUNTil <run_until_spec>
```

The RUNTil (run until) command allows you to define a stop condition when the run mode is repetitive. Specifying OFF causes the analyzer to make runs until either the display's Stop field is touched or the STOP command is issued.

There are four conditions based on the time between the X and O markers. Using this difference in the condition is effective only when time markers have been turned on (see the MMODe command in the Waveform Menu Commands, chapter 4). These four conditions are as follows:

- Less Than (LT) a specified time value.
- Greater Than (GT) a specified time value.
- In the range (INRange) between two time values.
- Out of the range (OUTRange) between two time values.

End points for the INRange and OUTRange should be at least 250 ps apart since this is the minimum time at which data is sampled.

When Compare is allocated to this module, there are two conditions that are based on a comparison of the acquired state data and the compare data image. The analyzer will run until one of the following conditions is true:

- Every channel of every label has the same value (EQUal) unless unmasked.
- Any channel of any label that is not masked has a differenct value (NEQual).

`<run_until_spec>` {OFF|LT,<value>|GT, <value>|INRange, <value>,<value>|OUTRange, <value>,<value>|EQUal|NEQual}

`<value>` real number

Examples

```
OUTPUT XXX;":LIST:RUNTIL GT, 800.0E-6"
OUTPUT XXX;":LIST:RUNTIL INRANGE, 4.5, 5.5"
```

Query `:LIST:RUNTil?`

The RUNTil query returns the current stop criteria.

Returned Format `[:LIST:RUNTil] <run_until_spec><NL>`

Example

```
OUTPUT XXX;":LIST:RUNTIL?"
```

SAMPclk (State Mode Only)

Command `:LIST:SAMPclk <over_sample_value>`

The SAMPclk command allows you to specify the current oversampling value from 1 to 32 samples per clock depending on the current value of the external clock period. The <over_sample_value> times the sample clock cannot exceed 2 GHz. For example, at a clock speed of 100 MHz the maximum <over_sample_value> is 16 (100 × 16 = 1.6 GHz). Additionally, at the slowest clock rate (20 MHz), the maximum <over_sample_value> can be 32 (32 × 20 MHz = 640 MHz). This command results in an error (–211, Legal command but settings conflict) if the State mode is not selected.

`<over_sample_value>` An integer from 1 to 32 in powers of 2

Example

```
OUTPUT XXX;":LIST:SAMPLCK 16"
```

Query `:LIST:SAMPclk?`

The SAMPclk query returns the current value of oversampling.

Returned Format `[:LIST:SAMPclk] <over_sample_value><NL>`

Example

```
OUTPUT XXX;":LIST:SAMPLCLK?"
```

SHOW (State Mode Only)

Command `:LIST:SHOW {ALL|EXTClk}`

The SHOW command allows you to specify which states will be displayed in the listing menu. The ALL option displays all acquired states. The EXTClk option displays only the externally clocked states in the oversampled mode. This command results in an error (–211, Legal Command but settings conflict) if a Timing mode is selected.

Example

```
OUTPUT XXX;":LIST:SHOW ALL"
```

Query `:LIST:SHOW?`

The SHOW query returns the current setting.

Returned Format `[:LIST:SHOW] {ALL|EXTClk}<NL>`

Example

```
OUTPUT XXX;":LIST:SHOW?"
```

SOFFset (State Mode Only)

Command

```
:LIST:SOFFset <coarse_time>[,<fine_time>
[, <pod_skew_time1>, ..., <pod_skew_time10>]]
```

The SOFFset command allows you to specify the offset between the external clock and the internal sample clock in the State mode. An error (–211, Legal command but settings conflict) results if the State mode is not selected.

`<coarse_time>` a real number from –5 ns to +5 ns in increments of 200 ps

`<fine_time>` an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns

`<pod_skew_time1...10>` an absolute real number that is within ±500 ps of <coarse_time> with a maximum of ±5 ns

Examples

```
OUTPUT XXX;":LIST:SOFFSET –1.2E–9"
OUTPUT XXX;":LISTFORM:SOFFSET 1.1E–9, 1.3E–9, 1.25E–9,
1.25E-9"
```

Query

```
:LIST:SOFFset?
```

The SOFFset query returns the current setting for the clock offset. If the SOFFset setting is only set with <coarse_time>, the query returns one time value. If the setting is set with <coarse_time>, <fine_time>, and <pod_skew_time>, the query returns <coarse_time>, <fine_time> followed by a time value for each pod.

Returned Format

```
[:LIST:SOFFset]  <coarse_time>[,<fine_time>,
<pod_skew_time1>, ...,<pod_skew_time10>]<NL>
```

Example

```
OUTPUT XXX;":LIST:SOFFSET?"
```

SPERiod

Command `:LIST:SPERiod <sample_period>`

The SPERiod command allows you to set the sample period for the next acquisition of the timing analyzer in the wide timing mode. If the fast timing mode is selected, the sample period is always 250 ps. When this command is sent, the acquisition mode is automatically set to manual (see TRIGger:ACQuisition in chapter 3). The value you specify for this command is rounded to the nearest allowable setting.

This command results in an error (–211, Legal command but settings conflict) if the analyzer is in the state mode or the fast timing mode.

`<sample_period>` real number from 500 ps to 65.536 µs

Example

```
OUTPUT XXX;":LIST:SPERIOD 50E-9"
```

Query `:LIST:SPERiod?`

The SPERiod query returns the sample period current acquisition. If there is no valid data, the query returns 9.9E37.

Returned Format `[:LIST:SPERiod] <sample_period><NL>`

Example

```
OUTPUT XXX;":LIST:SPERIOD?"
```

TAVerage

Query `:LIST:TAVerage?`

The TAVerage query returns the value of the average time between the X and O markers when the marker mode is MSTats. If there is no valid acquired data, the query returns 9.9E37.

Returned Format `[:LIST:TAVerage] <time_value><NL>`

`<time_value>` real number

Example

```
OUTPUT XXX;":LIST:TAVERAGE?"
```

TMAXimum

Query `:LIST:TMAXimum?`

The TMAXimum query returns the value of the maximum time between the X and O markers when the marker mode is MSTats. If there is no valid acquired data, the query returns 9.9E37.

Returned Format `[:LIST:TMAXimum] <time_value><NL>`

`<time_value>` real number

Example

```
OUTPUT XXX;":LIST:TMAXIMUM?"
```

TMINimum

Query `:LIST:TMINimum?`

The TMINimum query returns the value of the minimum time between the X and O Markers when the marker mode is MSTats. If the acquired data is not valid, the query returns 9.9E37.

Returned Format `[:LIST:TMINimum] <time_value><NL>`

`<time_value>` real number

Example

```
OUTPUT XXX;":LIST:TMINIMUM?"
```

TPOSition

Command `:LIST:TPOSition {STARt|CENTer|END|DELay, <time_val>|POSTstore, <post_value>}`

The TPOSition (trigger position) command allows you to set the trigger at the start, center, end or at any position in the trace using delay or poststore. Delay is specified as a real number representing the time between the trigger and the first acquired sample. In the state mode, if the external clock period is greater than 16 ns, the poststore clock runs at the external clock frequency and the delay limits are 0 ns to (1048574 × external clock period). If the external clock period is less than 16 ns, the external clock is divided down by a power of 2 such that the period of the poststore clock is a minimum of 16 ns.

Otherwise, in the timing mode with the sequencer running at 500 MHz, the poststore clock runs at the same rate as the sample clock or *sample period*×2^x where x is great enough so that the minimum poststore clock period is 16 ns.

Poststore is defined as 1 to 99 percent with a poststore of 99 percent being the same as start position and a poststore 1 percent being the same as an end trace.

If the analyzer is a timing analyzer, this command will automatically set the acquisition mode to MANUAL (see "TRIGger:ACQuisition" on page 3-11).

`<time_val>` real number from 0 to (1048574 × external clock period) or divided down sample period, or divided down external clock period.

`<post_value>` integer from 1 to 99 representing percentage of poststore.

Example

```
OUTPUT XXX;":LIST:TPOSITION CENTER"
```

Query

```
:LIST:TPOSition?
```

The TPOSition query returns the current trigger position setting. If the timer resource is turned off in that sequence level, the query returns 9.9E37.

Returned Format

```
[:LIST:TPOSition] {STARt|CENTer|END|DELay,
<time_val>|POSTstore, <post_value>}<NL>
```

Example

```
OUTPUT XXX;":LIST:TPOSITION?"
```

VRUNs

Query `:LIST:VRUNs?`

The VRUNs query returns the number of valid runs and total number of runs made when the marker mode is MSTats. Valid runs are those where the pattern search for both the X and O markers was successful, resulting in valid delta time measurements.

Returned Format `[:LIST:VRUNs] <valid_runs>, <total_runs><NL>`

`<valid_runs>` zero or positive integer

`<total_runs>` zero or positive integer

Example

```
OUTPUT XXX;":LIST:VRUNS?"
```

XCONdition

Command `:LIST:XCONdition {<state_options>|<timing_options>}`

The XCONdition command specifies where the X marker is placed. The X marker can be placed on the entry or exit point of the XPATtern when in the PATTern or STATistics marker modes. An additional option in state mode is CLOCk, which places the X marker on externally clocked states only, ignoring the oversampled states.

`<state_options>` {ENTering|LEAVing|CLOCk}

`<timing_options>` {ENTering|LEAVing}

Examples

```
OUTPUT XXX; ":LIST:XCONDITION ENTERING"
OUTPUT XXX; ":LIST:XCONDITION CLOCK"
```

Query `:LIST:XCONdition?`

The XCONdition query returns the current setting.

Returned Format `[:LIST:XCONdition] {<state_options>|<timing_options>}<NL>`

Example

```
OUTPUT XXX;":LIST:XCONDITION?"
```

XOTag

Query `:LIST:XOTag?`

The XOTag query returns the time from the X marker to the O marker. If the acquired data is not valid, the query returns 9.9E37.

Returned Format `[:LIST:XOTag] <XO_time><NL>`

`<time_time>` real number

Example

```
OUTPUT XXX;":LIST:XOTAG?"
```

XOTime

Query `:LIST:XOTime?`

The XOTime query returns the time from the X marker to the O marker. If the acquired data is not valid, the query returns 9.9E37.

Returned Format `[:LIST:XOTime] <value><NL>`

`<value>` real number

Example

```
OUTPUT XXX;":LIST:XOTIME?"
```

XPATtern

Command `:LIST:XPATtern <label_name>,<label_pattern>`

The XPATtern command allows you to construct a pattern recognizer term for the X marker which is then used with the XSEarch criteria and XCONdition when placing the marker on patterns. Since this command deals with only one label at a time, a complete specification could require several invocations.

When the value of a pattern is expressed in binary, it represents the bit values for the label inside the pattern recognizer term. Whatever base is used, the value must be between 0 and $2^{32} - 1$, since a label may not have more than 32 bits. Because the <label_pattern> parameter may contain don't cares, it is handled as a string of characters rather than a number.

`<label_name>` string of up to 6 alphanumeric characters

`<label_pattern>` "{#B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Example `OUTPUT XXX; ":LIST:XPATTERN 'A','511'"`

Query `:LIST:XPATtern? <label_name>`

The XPATtern query, in pattern marker mode, returns the pattern specification for a given label name. In the time marker mode, the query returns the pattern under the X marker for a given label. If the X marker is not placed on valid data, don't cares (X) are returned.

Returned Format `[:LIST:XPATtern] <label_name>, <label_pattern><NL>`

Example `OUTPUT XXX;":LIST:XPATTERN? 'A'"`

XSEarch

Command `:LIST:XSEarch <occurrence>, <origin>`

The XSEarch command defines the search criteria for the X marker which are then used with the associated XPATtern recognizer specification and the XCONdition when placing the marker on patterns. The origin parameter tells the marker to begin a search either at the start of acquired data or at the trigger. The occurrence parameter determines which occurrence of the XPATtern recognizer specification, relative to the origin, the marker actually searches for. An occurrence of 0 (zero) places a marker on the origin. With a negative occurrence, the marker searches before the origin. With a positive occurrence, the marker searches after the origin.

`<origin>` {TRIGger|STARt}

`<occurrence>` integer from –131071 to +131071

Example `OUTPUT XXX; ":LIST:XSEARCH,+10,TRIGGER"`

Query `:LIST:XSEarch?`

The XSEarch query returns the search criteria for the X marker.

Returned Format `[:LIST:XSEarch] <occurrence>,<origin><NL>`

Example

```
OUTPUT XXX;":LIST:XSEARCH?"
```

XSTate

Query `:LIST:XSTate?`

The XSTate query returns the line number in the listing where the X marker resides (–131071 to +131071). If data is not valid, the query returns 500000.

Returned Format `[:LIST:XSTate] <state_num><NL>`

`<line_number>` an integer from –131071 to +131071, or 500000

Example

```
OUTPUT XXX;":LIST:XSTATE?"
```

XTAG

Command

```
:LIST:XTAG <time_value>
```

The XTAG command specifies the time tag value on which the X Marker should be placed. If there is no valid acquired data, no action is performed.

`<time_value>` real number

Example

```
OUTPUT XXX;":LIST:XTAG 40.0E-6"
```

Query

```
:LIST:XTAG?
```

The XTAG query returns the X Marker position in time. If there is no valid acquired data, the query returns 9.9E37.

Returned Format

```
[:LIST:XTAG] <time_value><NL>
```

Example

```
OUTPUT XXX;":LIST:XTAG?"
```

XTIMe

Command `:LIST:XTIMe <time_value>`

The XTIMe command positions the X marker in time when the marker mode is TIME. If the acquired data is not valid, the command performs no action.

`<time_value>` real number

Example

```
OUTPUT XXX; ":LIST:XTIME 40.0E-6"
```

Query `:LIST:XTIMe?`

The XTIMe query returns the X marker position in time. If the acquired data is not valid, the query returns 9.9E37.

Returned Format `[:LIST:XTIMe] <time_value><NL>`

Example

```
OUTPUT XXX;":LIST:XTIME?"
```

6

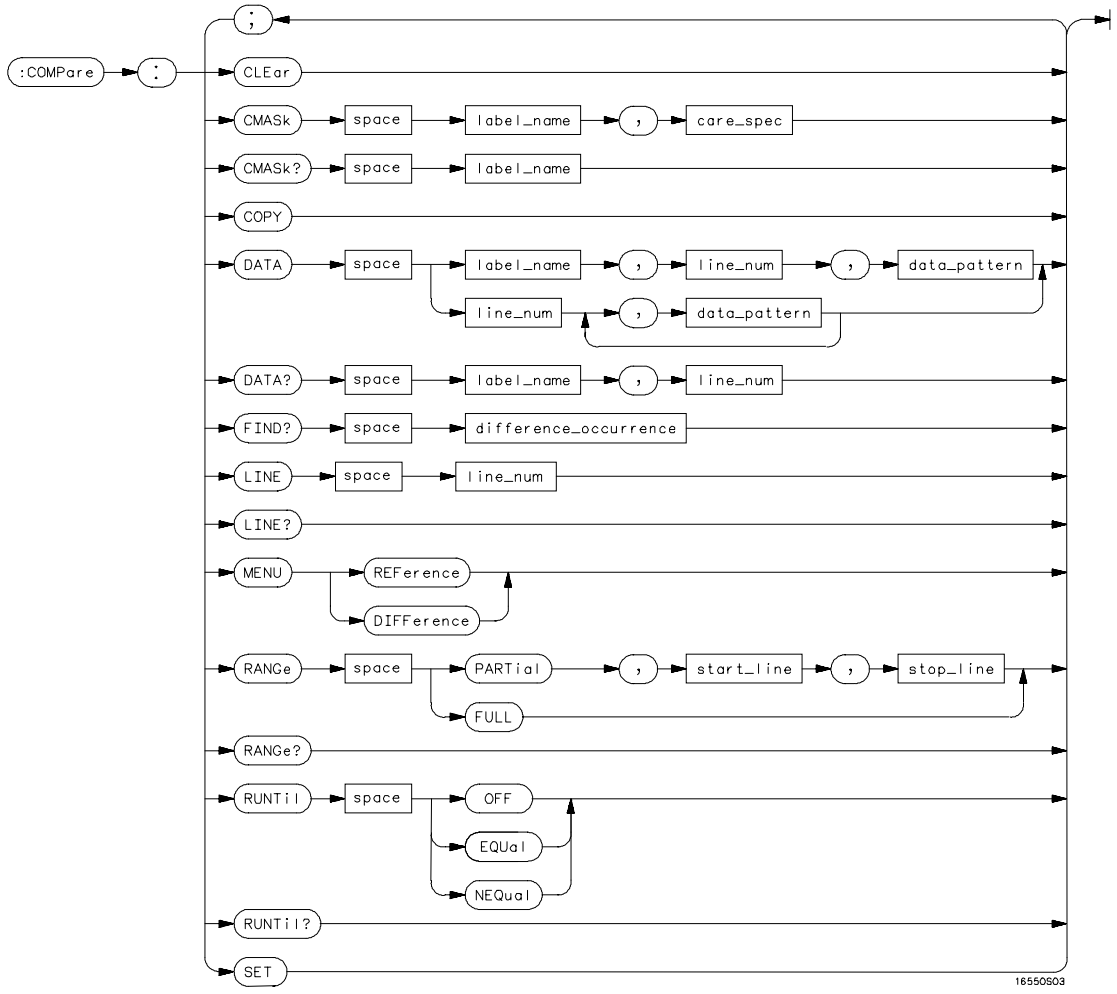
Compare Menu Commands

Introduction

Commands in the state COMPare subsystem provide the ability to do a bit-by-bit comparison between the acquired data listing and a compare reference data image. The commands are:

- CLEar
- CMASk
- COPY
- DATA
- FIND
- LINE
- MENU
- RANGe
- RUNTil
- SET

Figure 6-1



COMPare Subsystem Syntax Diagram

Table 6-1 **Compare Parameter Values**

Parameter	Values
label_name	string of up to 6 characters
care_spec	string of characters "{*|.}..."
*	care
.	don't care
line_num	integer from –131071 to +131071
data_pattern	"{B{0|1|X} . . . | #Q{0|1|2|3|4|5|6|7|X} . . . | #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . | {0|1|2|3|4|5|6|7|8|9} . . . }"
difference_occurence	integer from 1 to 131071
start_line	integer from –131071 to +131071
stop_line	integer from <start_line> to +131071

COMPare

Selector `:COMPare`

The COMPare selector is used as part of a compound header to access those commands found in the Compare menu. It always follows the SELECT(n) command when you first access the module. It must precede any command you wish to send to the COMPare subsystem unless you send combined command messages (see "Combined Commands in the Same Subsystem" in Chapter 1 of the *Agilent Technologies 16500B/16501A Programmer's Guide*).

Example

```
OUTPUT XXX;":MACHINE1:COMPARE:FIND? 819"
```

CLEar

Command `:COMPare:CLEar`

The CLEar command clears all "don't cares" in the reference listing and replaces them with whatever data was present before, if the "don't cares" were the result of the SET command. Otherwise, the CLEar command replaces the "don't cares" with zeros (see SET command).

Example

```
OUTPUT XXX;":COMPARE:CLEAR"
```

CMASk

Command `:COMPare:CMASk <label_name>, <care_spec>`

The CMASk (Compare Mask) command allows you to set the bits in the channel mask for a given label in the compare listing image to "compares" or "don't compares."

`<label_name>` A string of up to 6 alphanumeric characters

`<care_spec>` A string of characters "{*|.}..." (32 characters maximum)

`<*>` An indicator that tells the logic analyzer that it cares about this bit.

`<.>` An indicator that tells the logic analyzer that it does not care about this bit (don't care).

Example

```
OUTPUT XXX;":COMPARE:CMASK 'DATA', '*.**..**'"
```

Query `:COMPare:CMASk? <label_name>`

The CMASk query returns the state of the bits in the channel mask for a given label in the compare listing image.

Returned Format `[:COMPare:CMASk] <label_name>, <care_spec>`

Example

```
OUTPUT XXX;":COMPARE:CMASK? 'DATA'"
```

COPY

Command `:COMPare:COPY`

The COPY command copies the current acquired Listing into the Compare Reference image. It does not affect the compare range or channel mask settings.

Example

```
OUTPUT XXX;":COMPARE:COPY"
```

DATA

Command

```
:COMPare:DATA {<label_name>, <line_num>,
<data_pattern>|<line_num>, <data_pattern>
[, <data_pattern>]... }
```

The DATA command allows you to edit the compare reference image for a given label and state row. When DATA is sent to an instrument where no compare reference is defined (such as at power-up), all other data in the image is set to "don't cares."

Not specifying the <label_name> parameter allows you to write data patterns to more than one label for the given line number. The first pattern is placed in the left-most label, with the following patterns being placed in a left-to-right fashion (as seen on the Compare display). Specifying more patterns than there are labels simply results in the extra patterns being ignored.

Because "don't cares" (Xs) are allowed in the data pattern, it must always be expressed as a string. You may still use different bases, but don't cares cannot be used in a decimal number.

`<label_name>` A string of up to 6 alphanumeric characters

`<line_num>` An integer from –131071 to +131071

`<data pattern>` A string in one of the following forms:
"{B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Examples

```
OUTPUT XXX;":COMPARE:DATA 'CLOCK', 42, '#B011X101X'"
OUTPUT XXX;":COMPARE:DATA 'OUT3', 0, '#HFF40'"
OUTPUT XXX;":COMPARE:DATA 129, '#BXX00', '#B1101', '#B10XX'"
OUTPUT XXX;":COMPARE:DATA –511, '4', '64', '16', 256', '8',
'16'"
```

Query `:COMPare:DATA? <label_name>, <line_num>`

The DATA query returns the value of the compare listing image for a given label and state row.

Returned Format `[:COMPare:DATA] <label_name>, <line_num>, <data_pattern><NL>`

Example

```
OUTPUT XXX;":COMPARE:DATA? 'DATA', 512"
```

FIND

Query `:COMPare:FIND? <difference_occurrence>`

The FIND query is used to get the line number of a specified difference occurence (first, second, third, etc) within the current compare range, as dictated by the RANGe command (see page 6-10). A difference is counted for each line where at least one of the current labels has a discrepancy between its acquired state data listing and its masked reference image.

Invoking the FIND query updates both the Listing and Compare displays so that the line number returned is in the center of the screen.

Returned Format `[:COMPare:FIND] <difference_occurrence>, <line_num><NL>`

`<difference_occurrence>` integer from 1 to 131071

`<line_num>` integer from –131071 to +131071

Example

```
OUTPUT XXX;":COMPARE:FIND? 26"
```

LINE

Command

```
:COMPare:LINE <line_num>
```

The LINE command allows you to center the compare listing data about a specified line number.

`<line_num>` An integer from –131071 to +131071

Example

```
OUTPUT XXX;":COMPARE:LINE –511"
```

Query

```
:COMPare:LINE?
```

The LINE query returns the current line number specified.

Returned Format

```
[:COMPare:LINE] <line_num>}<NL>
```

Example

```
OUTPUT XXX;":COMPARE:LINE?"
```

MENU

Command `:COMPare:MENU {REFerence|DIFFerence}`

The MENU command allows you to display the reference or the difference listings in the Compare menu.

Example

```
OUTPUT XXX;":COMPARE:MENU REFERENCE"
```

RANGe

Command `:COMPare:RANGe {FULL|PARTial, <start_line>, <stop_line>}`

The RANGe command allows you to define the boundaries for the comparison. The range entered must be a subset of the lines in the acquired data memory.

`<start_line>` integer from –131071 to +131071

`<stop_line>` integer from <start_line> to +131071

Examples

```
OUTPUT XXX;":COMPARE:RANGE PARTIAL, -511, 512"
OUTPUT XXX;":COMPARE:RANGE FULL"
```

Query

```
:COMPare:RANGe?
```

The RANGe query returns the current boundaries for the comparison.

Returned Format

```
[::COMPare:RANGe] {FULL|PARTial, <start_line>,
<stop_line>}<NL>
```

Example

```
OUTPUT 707;":COMPARE:RANGE?"
```

RUNTil

Command

```
:COMPare:RUNTil {OFF|LT,<value>|GT,
<value>|INRange, <value>,<value>|OUTRange,
<value>,
<value>|EQUal|NEQual}
```

The RUNTil (run until) command allows you to define a stop condition when the run mode is repetitive. Specifying OFF causes the analyzer to make runs until either the display's Stop field is touched or the STOP command is issued.

There are four conditions based on the time between the X and O markers. Using this difference in the condition is effective only when time markers have been turned on (see the MMODe command in the Waveform Menu Commands, chapter 4). These four conditions are as follows:

- The difference is less than (LT) some value.
- The difference is greater than (GT) some value.
- The difference is inside some range (INRange).
- The difference is outside some range (OUTRange).

End points for the INRange and OUTRange should be at least 250 ps apart since this is the minimum time resolution of the time tag counter.

There are two conditions which are based on a comparison of the acquired state data and the compare reference image. You can run until one of the following conditions is true:

- Every channel of every label has the same value (EQUal).
- Any channel of any label has a different value (NEQual).

`<value>` real number

Example

```
OUTPUT XXX;":COMPARE:RUNTIL EQUAL"
```

Query `:COMPare:RUNTil?`

The RUNTil query returns the current stop criteria for the comparison when running in repetitive trace mode.

Returned Format

```
[:COMPare:RUNTil] {OFF| LT,<value>|GT,<value>lINRange,
<value>,<value>|OUTRange,<value>,<value>|EQUal|NEQual}<NL>
```

Example

```
OUTPUT XXX;":COMPARE:RUNTIL?"
```

SET

Command `:COMPare:SET`

The SET command sets every state in the reference listing to "don't care." If you send the SET command by mistake you can immediately send the CLEar command to restore the previous data. This is the only time the CLEar command will not replace "don't cares" with zeros.

Example

```
OUTPUT XXX;":COMPARE:SET"
```

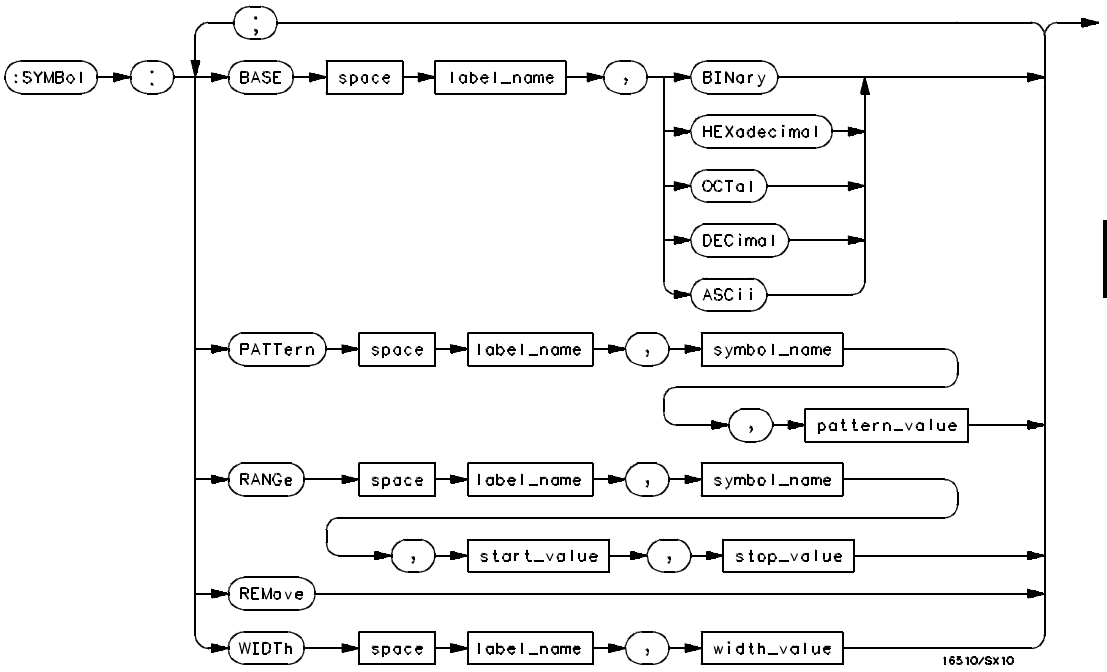
Symbol Subsystem Commands

Introduction

The SYMBol subsystem contains the commands that allow you to define symbols in the computer and download them to the 16517A/18A logic analyzer. The commands in this subsystem are:

- BASE
- PATTern
- RANGe
- REMove
- WIDTh

Figure 7-1



SYMBol Subsystem Syntax Diagram

Table 7-1 **SYMBol Parameter Values**

Parameter	Values
label_name	string of up to 6 alphanumeric characters
symbol_name	string of up to 16 alphanumeric characters
pattern_value	"{#B{0|1|X} . . . | #Q{0|1|2|3|4|5|6|7|X} . . . | #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . | {0|1|2|3|4|5|6|7|8|9} . . . }"
start_value	"{#B{0|1} . . . | #Q{0|1|2|3|4|5|6|7} . . . | #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F} . . . | {0|1|2|3|4|5|6|7|8|9} . . . }"
stop_value	"{#B{0|1} . . . | #Q{0|1|2|3|4|5|6|7} . . . | #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F} . . . | {0|1|2|3|4|5|6|7|8|9} . . . }"
width_value	integer from 1 to 16

SYMBol

Selector `:SYMBol`

The SYMBol selector is used as a part of a compound header to access the commands used to create symbols. It always follows the SELECT(n) command when you first access the module. It must precede any command you wish to send to the SYMBol subsystem unless you send combined command messages (see "Combined Commands in the Same Subsystem" in Chapter 1 of the *Agilent Technologies 16500B/16501A Programmer's Guide*).

Example

```
OUTPUT XXX;":SYMBOL:BASE 'DATA', BINARY"
```

BASE

Command

```
:SYMBol:BASE <label_name>, <base_value>
```

The BASE command sets the base in which symbols for the specified label will be displayed in the symbol menu. It also specifies the base in which the symbol offsets are displayed when symbols are used.

> BINary is not available for labels with more than 20 bits assigned. In this case the base will default to HEXadecimal.

`<label_name>` string of up to 6 alphanumeric characters

`<base_value>` {BINary|HEXadecimal|OCTal|DECimal|ASCii}

Example

```
OUTPUT XXX;":SYMBOL:BASE 'DATA', HEXADECIMAL"
```

PATTern

Command

```
:SYMBol:PATTern <label_name>, <symbol_name>,
<pattern_value>
```

The PATTern command allows you to create a pattern symbol for the specified label. Because don't cares (X) are allowed in the pattern value, it must always be expressed as a string. You may still use different bases, but don't cares cannot be used in a decimal number.

`<label_name>` string of up to 6 alphanumeric characters

`<symbol_name>` string of up to 16 alphanumeric characters

`<pattern_value>` "{#B{0|1|X} . . . |
#Q{0|1|2|3|4|5|6|7|X} . . . |
#H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F|X} . . . |
{0|1|2|3|4|5|6|7|8|9} . . . }"

Example

```
OUTPUT XXX;":SYMBOL:PATTERN 'STAT', 'MEM_RD', '#H01XX'"
```

RANGe

Command

```
:SYMBol:RANGe <label_name>, <symbol_name>,
<start_value>, <stop_value>
```

The RANGe command allows you to create a range symbol containing a start value and a stop value for the specified label. The values may be in binary (#B), octal (#Q), hexadecimal (#H) or decimal (default). Don't cares are not allowed in any base.

<label_name>	string of up to 6 alphanumeric characters
<symbol_name>	string of up to 16 alphanumeric characters
<start_value>	"{#B{0|1} . . . | #Q{0|1|2|3|4|5|6|7} . . . | #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F} . . . | {0|1|2|3|4|5|6|7|8|9} . . . }"
<stop_value>	"{#B{0|1} . . . | #Q{0|1|2|3|4|5|6|7} . . . | #H{0|1|2|3|4|5|6|7|8|9|A|B|C|D|E|F} . . . | {0|1|2|3|4|5|6|7|8|9} . . . }"

Example

```
OUTPUT XXX;":SYMBOL:RANGE 'STAT', 'IO_ACC', '0', '#H000F'"
```

REMove

Command `:SYMBol:REMove`

The REMove command deletes all symbols from a specified module.

Example

```
OUTPUT XXX;":SYMBOL:REMOVE"
```

WIDTh

Command `:SYMBol:WIDTh <label_name>, <width_value>`

The WIDTh command specifies the number of characters in which the symbol names will be displayed when symbols are used.

> The WIDTh command does not affect the displayed length of the symbol offset value.

`<label_name>` string of up to 6 alphanumeric characters

`<width_value>` integer from 1 to 16

Example

```
OUTPUT XXX;":SYMBOL:WIDTH 'DATA', 9 "
```

8

DATA and SETup Commands

Introduction

The DATA and SETup commands are SYSTem commands that allow you to send and receive block data between the 16517A/18A and a computer. Use the DATA instruction to transfer acquired timing and state data, and the SETup instruction to transfer instrument configuration data. This is useful for:

- Re-loading to the logic analyzer
- Processing data later
- Archiving acquired data
- Processing data in the computer

This chapter explains how to use these commands.

The format and length of block data depends on the instruction being used, the configuration of the instrument, and the amount of acquired data. The length of the data block can be up to 655,520 bytes in the 16517A/18A.

The SYSTem:DATA section describes each part of the block data as it will appear when using the DATA instruction. The beginning byte number, the length in bytes, and a description is given for each part of the block data. This is intended to be used primarily for processing of data in the computer.

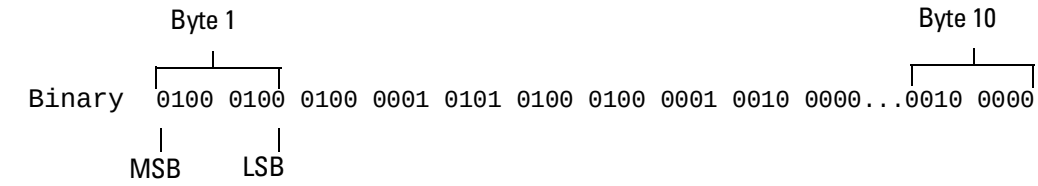
Do not change the block data in the computer if you intend to send the block data back into the logic analyzer for later processing. Changes made to the block data in the computer could have unpredictable results when sent back to the logic analyzer.

Data Format

To understand the format of the information within the block data, there are four important things to keep in mind.

- Data is sent to the computer in binary form.
- Each byte, as described in this chapter, contains 8 bits.
- The first bit of each byte is the MSB (most significant bit).
- Byte descriptions are printed in binary, decimal, or ASCII depending on how the data is described.

For example, the first ten bytes that describe the section name contain a total of 80 bits as follows:




```
Decimal  68 65 84 65 32 32 32 32 32 32
  ASCII  DATA space space space space space space
```

:SYSTem:DATA

Command `:SYSTem:DATA <block_data>`

The SYSTem:DATA command transmits acquired data from the computer into the acquisition memory of the 16517A/18A logic analyzer.

The block data consists of a variable number of bytes containing information previously captured by the acquisition chips. The format is described in the "Acquisition Data Description" section later in this chapter. Since no parameter checking is performed, out-of-range values could cause instrument lockup; therefore, care should be taken when transferring the data string into the logic analyzer.

The <block_data> parameter can be broken down into a <block_length_specifier> and a variable number of <section>'s.

The <block_length_specifier> always takes the form #8DDDDDDDD. Each D represents a digit (ASCII characters "0" through "9"). The value of the eight digits represents the total length of the block (all sections). For example, if the total length of the block is 655,520 bytes, the block length specifier would be "#800655520".

Each <section> consists of a <section header> and <section data>. The <section data> format varies for each section. For the DATA instruction, there is only one <section>, which is composed of a data preamble followed by the acquisition data. This section has a variable number of bytes depending on configuration and amount of acquired data.

`<block_data>` <block_length_specifier><section>[<section>...]

`<block_length_specifier>` #8<length>

`<length>` The total length of all sections in byte format (must be represented with 8 decimal digits)

`<section>` <section_header><section_data>

`<section_header>` 16 bytes, described in "Section Header Description," on page 8-6.

`<section_data>` Format depends on the specific section.

Example

```
OUTPUT XXX;":SYSTEM:DATA " <block_data>
```

> The total length of a section is 16 (for the section header) plus the length of the section data. So when calculating the value for <length>, don't forget to include the length of the section headers.

Query

```
:SYSTem:DATA?
```

The SYSTem:DATA query returns the block data to the computer. The data received as a result of the SYSTem:DATA query reflect the configuration of the analyzer when the last run was performed. Any changes made since then through either front-panel operations or programming commands do not affect the stored configuration that accompanies the data.

Returned Format

```
[:SYSTem:DATA] <block_data><NL>
```

Example

See "Transferring the logic analyzer acquired data" on page 9-11 in chapter 9, "Programming Examples" for an example.

Section Header Description

The section header uses bytes 1 through 16 (this manual begins counting at 1; there is no byte 0). The 16 bytes of the section header are as follows:

Byte Position	
1	10 bytes - Section name ("DATA space space space space space space" in ASCII for the DATA section).
11	1 byte - Reserved
12	1 byte - Module ID (0000 0100 binary or 4 decimal for the 16517A/18A)
13	4 bytes - Length of section in number of bytes that, when converted to decimal, specifies the number of bytes contained in the section.

Section Data

For the SYSTem:DATA command, the <section data> parameter consists of two parts: the data preamble and the acquisition data. These are described in the following two sections.

Data Preamble Description

The block data is organized as 144 bytes of preamble information, followed by the acquisition data. The preamble contains all the information necessary to describe the data itself, but not necessarily how it was acquired. For instance, the sample rate, the number of pods, and the number of samples are contained in the preamble. The values stored in the preamble represent the captured data currently stored in this structure and not the current analyzer configuration. For example, the mode of the data (byte 21) may be STATE, while the current setup of the analyzer is TIMING.

The preamble (bytes 17 through 16) consists of the following 144 bytes:

17	2 bytes - Instrument ID (always 16517 decimal for the 16517A/18A)
19	2 bytes - Revision Code for the preamble format

21 1 byte - Machine mode, one of the following decimal values:

1 = TIMING
2 = STATE

22 1 byte - Channel mode, one of the following decimal values:

0 = Full Channel Mode
1 = Half Channel Mode

23 1 byte - Number of pods, a decimal number representing the number of pods on cards connected to and including the Master card. The maximum value is 10.

24 1 byte - Master card, a decimal number representing the relative position of the Master card, starting with the value of 1. For example, with Slave cards in slots B and D and the Master card in C, this field will equal 2.

25 1 byte - Trigger found - This byte returns a true if the trigger point was found when the data was acquired. A decimal 1 indicates true.

1 = TRUE
0 = FALSE

26 1 byte - Prestore valid - This byte returns a true if the Prestore interval elapsed when the data was acquired. A decimal 1 indicates true.

1 = TRUE
0 = FALSE

27 1 byte - Measurement complete - This byte returns a true if the measurement ran to completion (the Poststore interval elapsed) when the data was acquired. A decimal 1 indicates true.

1 = TRUE
0 = FALSE

28 1 byte - Unused.

29 4 bytes - Number of valid samples - A decimal integer representing the number of samples that were acquired on each channel. If equal to zero, there is no valid data.

33 1 byte - Armed by - a decimal integer indicating the arming source.

0 = indicates the module was armed independently or by the IMB
1 = indicates the module was armed by the rear-panel SMB connector

34 1 byte - Unused.

35 1 byte - Clock edge - a decimal integer representing which clock edge was specified for the External Clock in the FORMAT menu when the data was acquired (STATE mode only).

0 = RISING
1 = FALLING

36 1 byte - Module Event Status Register - a binary copy of the GPIB Module Event Status Register at the time of data acquisition. The bit assignments are:

Bit number	Status
7	Not used
6	Not used
5	1 = STATE mode Run was done with External Clock out of spec 0 = STATE mode Run was done with External Clock in spec
4	1 = Run was done using default skew values 0 = Run was done with pod skews nulled out
3	1 = One or more Pattern searches failed 0 = Pattern searches did not fail
2	1 = Trigger found 0 = Trigger not found
1	1 = Repetitive Run Until condition was satisfied 0 = Repetitive Run Until condition was not satisfied
0	1 = Measurement ran to completion 0 = Measurement did not run to completion

37 4 bytes - Trigger point - a decimal number representing the sample number that corresponds to the trigger point. This byte is valid only if byte 25 = 1. For an acquisition with a delayed trigger (trigger point precedes the first sample) this byte will be 0.

41 2 bytes - Unused

43 2 bytes - Samples per external clock - a decimal integer representing the number of samples taken per external clock period. This field is valid only in the State mode and when oversampling. Normally it is set to 1.

45 80 bytes - Clock offset - ten decimal integers representing the number of picoseconds that the internal sample clock was offset from the external clock edge. This data is valid only in STATE mode. A positive value means that the internal sampling was delayed from the external clock edge.

These 80 bytes are arranged as an array of ten 64-bit values. The first two values are for the pods on the master card, followed by values for the pods on the expansion cards in top-to-bottom slot order. Pod 1 precedes pod 2 for each pair.

Example The following 64 bits in binary would equal 2,000 picoseconds or, 2 nanoseconds:

```
00000000 00000000 00000000 00000000 00000000 00000000 00000111 11010000
```

125 8 bytes - Sample period - a decimal integer representing the sample period, in femtoseconds.

Example The following 64 bits in binary would equal 1,000,000 femtoseconds, or 1 ns:

```
00000000 00000000 00000000 00000000 00000000 00001111 01000010 01000000
```

133 8 bytes - Trigger delay - a decimal integer representing the number of picoseconds that the trigger point precedes the first sample when using a delayed trigger. If this value is zero, then there was no delayed trigger, and bytes 37 through 40 determine the actual trigger point. This byte is valid only if byte 25 = 1.

141 20 bytes - Unused

Acquisition Data Description

The acquisition data section consists of a variable number of bytes depending on a how many cards are installed and the acquisition mode. The first eight bytes (bytes 161 through 168) contain the value of the real-time clock when the data was acquired. The acquired data resides in the rest of the data block from byte 169 through byte n. There are eight bytes following byte n (numbered n + 1 through n + 8) that are unused.

Time Stamp Data

When running in an Agilent Technologies 16500B, this time will be valid, with the year field set to the number of years after 1990. In an Agilent Technologies 16500A mainframe, which has no real-time clock, the year will be set to 255, to signify an invalid time.

161 1 byte - A decimal integer representing year

162 1 byte - A decimal integer representing month

163 1 byte - A decimal integer representing the day of the month

164 1 byte - A decimal integer representing the day of the week

165 1 byte - A decimal integer representing the hour in 24 hour format

166 1 byte - A decimal integer representing minutes

167 1 byte - A decimal integer representing seconds

168 1 byte - Unused

Acquired Data

The 16517A/18A has two 8-bit pods on each card, and up to five cards may be internally connected together, for a maximum of 80 channels. The data array is always written out in the order of the slots in which the cards are installed, from top to bottom. Unlike many of the other Agilent Technologies 16500 system modules, the data from the master card (16517A) is not placed in any special position in the array. Within each card the data from pod 2 always precedes the data from pod 1. The data are arranged so that all of the first samples across all pods are together, followed by all of the next samples.

The format of the data depends on the channel mode setting (byte 22). If the analyzer is in the Full Channel Mode, then the first p bytes, where p is the number of pods (byte 23), were all acquired at the same time. When pointing to a sample from a particular pod, the next sample from the same pod is offset by the number of pods (byte 23). For example, assume an 16517A master card is installed in slot C, and two 16518A expansion cards are installed in slots B and D. With this configuration, the first six bytes were all acquired in the first sample period, followed by the next six bytes in the second sample period and so on. The first two bytes are from the card in slot B, the next two from the master card in slot C, and the next two from the card in slot D. The pattern then repeats itself the number of times specified by the number of valid samples (bytes 29 through 32).

			bit numbers 7 x x x x x x 0
169	slot B	sample 0	pod 2
170	slot B	sample 0	pod 1
171	slot C	sample 0	pod 2
172	Slot C	sample 0	pod 1
173	slot D	sample 0	pod 2
174	slot D	sample 0	pod 1
175	slot B	sample 1	pod 2
176	slot B	sample 1	pod 1
177	slot C	sample 1	pod 2
.	.	.	.
.	.	.	.
.	.	.	.
n			

In the Half Channel Mode, only the lowest 4 bits of each pod are used. Each byte consists of the data from both pods on a card. The data is packed into each byte, with the 4 bits from pod 2 in the upper bits and the 4 bits from pod 1 in the lower bits. When pointing to a sample from a particular pod, the next sample from the same pod is offset by a value equal to $\frac{number\ of\ pods}{2}$ (byte 23). Using the same example card configuration, the first byte will be the first sample from the card in slot B, followed by the first sample from the master card in slot C, and then the sample from the card in slot D. Again, the pattern repeats itself the number of times specified by the number of valid samples (bytes 29 through 32).

			bit numbers 7 x x x x x x 0
169	slot B	sample 0	pod 2 | pod 1
170	slot C	sample 0	pod 2 | pod 1
171	slot D	sample 0	pod 2 | pod 1
172	Slot B	sample 1	pod 2 | pod 1
173	slot C	sample 1	pod 2 | pod 1
174	slot D	sample 1	pod 2 | pod 1
175	slot B	sample 2	pod 2 | pod 1
176	slot C	sample 2	pod 2 | pod 1
177	slot D	sample 2	pod 2 | pod 1
.	.	.	. | .
.	.	.	. | .
.	.	.	. | .
n			

In the Full Channel Mode, the maximum number of samples per channel is 65536 (64K). In the Half Channel Mode, the maximum number of samples per channel is 128K. Therefore, the number of bytes in an acquisition that runs to completion is the same in either mode. In Full Channel Mode, the number of valid bytes is *number of valid samples* × *number of pods* (bytes 29-32 times byte 23). In the Half Channel Mode, the number of valid bytes is $\frac{\textit{valid number of samples} \times \textit{number of pods}}{2}$ (bytes 29-32 times byte 23)/ 2)), but the number of valid samples may be twice as large.

SYSTem:SETup

Command `:SYStem:SETup <block_data>`

The SYStem:SETup command configures the logic analyzer module as defined by the block data sent by the computer. The length of the configuration data block can be up to 9,872,178 bytes in the 16517A/18A; however, the typical number of bytes is closer to 1,000,000 bytes.

There are four data sections which are always returned. These are the strings which would be included in the section header:

"CONFIG "
"DISPLAY "
"BIG_ATTRIB"
"MACRO "

Additionally, the following sections may also be included, depending on the current configuration:

"SYMBOLS "
"COMPARE "

`<block_data>`	<block_length_specifier><section>[<section>...]
`<block_length_specifier>`	#8<length>
`<length>`	The total length of all sections in byte format (must be represented with 8 digits)
`<section>`	<section_header><section_data>
`<section_header>`	16 bytes in the following format: 10 bytes for the section name 1 byte reserved 1 byte for the module ID code (4 for the 16517A/18A logic analyzer) 4 bytes for the length of section data in number of bytes that, when converted to decimal, specifies the number of bytes contained in the section.
`<section_data>`	Format depends on the section.

> The total length of a section is 16 (for the section header) plus the length of the section data. So when calculating the value for <length>, don't forget to include the length of the section headers.

Example

```
OUTPUT XXX;"SETUP " <block_data>
```

Query

```
:SYStem:SETup?
```

The SYStem:SETup query returns a block of data that contains the current configuration to the computer.

Returned Format

```
[:SYStem:SETup] <block_data><NL>
```

Example

See "Transferring the logic analyzer configuration" on page 9-9 in Chapter 9, "Programming Examples" for an example.

Part 3

9 Programming Examples 9-1

Programming Examples

9

Programming Examples

Introduction

This chapter contains short, usable, and tested program examples that cover the most asked for examples. The examples are written in HP BASIC 6.2.

- Making a timing analyzer measurement
- Making a state analyzer measurement
- Transferring logic analyzer configuration between the logic analyzer and the computer
- Transferring logic analyzer data between the logic analyzer and the computer
- Checking for measurement completion
- Making a Compare Measurement
- Using the COMPARE:DATA? Query

Making a timing analyzer measurement

This program sets up the logic analyzer to make a simple timing analyzer measurement. This example can be used with E2433-60006 Logic Analyzer Training board to acquire and display the output of the ripple counter. It can also be modified to make any timing analyzer measurement.

```
10    ! ****************** TIMING ANALYZER EXAMPLE ******************
20    !                for for the Agilent Technologies 16517A Logic Analyzer
30    !
40    ! ************************************************************
50    ! Select the module slot in which the 16517A is installed.  In
60    ! this example, the 16517A is in slot B of the mainframe.
70    !
80    OUTPUT 707;":SELECT 2"
90    !
100   ! ************************************************************
110   ! Configure the 16517A as a timing analyzer in the fast-timing
120   ! mode.
130   !
150   OUTPUT 707;":FORMAT:TYPE FASTTIMING"
170   !
180   ! ************************************************************
190   ! Make a label "COUNT," give the label a positive polarity, and
200   ! assign the lower 4 bits to both pods.
210   !
220   OUTPUT 707;":FORMAT:REMOVE ALL"
230   OUTPUT 707;":FORMAT:LABEL 'COUNT',POS,#B1111,#B1111"
250   ! ************************************************************
260   ! Specify FF hex for resource term PATT1.
280   !
290   OUTPUT 707;":TRIGGER:PATTERN 'PATT1', 'COUNT', '#H80'"
300   !
310   ! ************************************************************
320   ! Remove any previously inserted labels, insert the "COUNT."
330   ! label, change the seconds-per-division to 100 ns, and display the
340   ! waveform menu.
350   !
360   OUTPUT 707;":WAVEFORM:REMOVE"
370   OUTPUT 707;":WAVEFORM:INSERT 'COUNT', ALL"
380   OUTPUT 707;":WAVEFORM:RANGE 1E-6"
390   OUTPUT 707;":MENU 2,2"
```

```
400   !
410   ! ***************************************************************
420   ! Run the timing analyzer in single mode.
430   !
440   OUTPUT 707;":RMODE SINGLE"
450   OUTPUT 707;":START"
460   !
470   ! ***************************************************************
480   ! Set the marker mode (MMODE) to pattern so that time tags are available
490   ! for marker measurements.  Place the X-marker on 40 hex and the O-
500   ! marker on 10 hex.  Then tell the timing analyzer to find the first
510   ! occurrence of 40h after the trigger and the first occurrence of 10h
520   ! after the X-marker is found.
530   !
540   OUTPUT 707;":WAVEFORM:MMODE PATTERN"
550   !
560   OUTPUT 707;":WAVEFORM:XPATTERN 'COUNT','#H40'"
570   OUTPUT 707;":WAVEFORM:OPATTERN 'COUNT','#H10'"
580   !
590   OUTPUT 707;":WAVEFORM:XCONDITION ENTERING"
600   OUTPUT 707;":WAVEFORM:OCONDITION ENTERING"
610   !
620   OUTPUT 707;":WAVEFORM:XSEARCH +1, TRIGGER"
630   OUTPUT 707;":WAVEFORM:OSEARCH +1, XMARKER"
640   !
650   ! ***************************************************************
660   ! Turn the longform and headers on, dimension a string for the query
670   ! data, sending the XOTIME query and print the string containing the
680   ! XOTIME query data.
690   !
700   OUTPUT 707;":SYSTEM:LONGFORM ON"
710   OUTPUT 707;":SYSTEM:HEADER ON"
720   !
730   DIM Mtime$[100]
740   OUTPUT 707;":WAVEFORM:XOTIME?"
750   ENTER 707;Mtime$
760   PRINT Mtime$
770   END
```

Making a state analyzer measurement

This state analyzer program selects the 16517A logic analyzer, displays the configuration menu, defines a state analyzer, displays the state trigger menu, sets a state trigger for multilevel triggering. This program then starts a single acquisition measurement while checking for measurement completion.

```
10    ! ******************** STATE ANALYZER EXAMPLE ************************
20    !                  for the Agilent Technologies 16517A Logic Analyzer
30    !
40    ! ****************** SELECT THE 16517A MODULE **********************
50    ! Select the module slot in which the 16517A is installed.  In this
60    ! example, the 16517A is in slot B of the mainframe.
70    !
80    CLEAR SCREEN
90    OUTPUT 707;"*CLS"
100   OUTPUT 707;":SELECT 2"
110   !
120   ! ******************** CONFIGURE THE STATE ANALYZER **********************
130   !
140   OUTPUT 707;":FORMAT:TYPE STATE"
150   OUTPUT 707;":MENU 2,0"
160   !
170   ! ****************** SETUP THE FORMAT SPECIFICATION ********************
180   ! Make a label "SCOUNT," give the label a positive polarity, and
190   ! assign the 8 bits to both pods.
200   !
210   OUTPUT 707;":FORMAT:REMOVE ALL"
220   OUTPUT 707;":FORMAT:LABEL 'SCOUNT', POS, 255, 255"
230   !
```

```
240   ! ****************** SETUP THE TRIGGER SPECIFICATION *******************
250   ! The trigger specification will use four sequence levels with the trigger
260   ! level on level four.  Resource terms PATT1 through PATT4 will be
270   ! used to store only desired counts from the 8-bit ripple counter.
280   !
290   ! Display the trigger menu.
300   !
310   OUTPUT 707;":MENU 2,1"
320   !
330   ! Create a 4-level trigger specification.
340   !
350   OUTPUT 707;":TRIGGER:SEQUENCE 4"
360   !
370   ! Define pattern terms PATT1, PATT2, PATT3, AND PATT4 to be 4096, 6656,
380   ! 8192 and 65280 respectively.
390   !
400   OUTPUT 707;":TRIGGER:PATTERN 'PATT1','SCOUNT','4096'"
410   OUTPUT 707;":TRIGGER:PATTERN 'PATT2','SCOUNT','6656'"
420   OUTPUT 707;":TRIGGER:PATTERN 'PATT3','SCOUNT','8192'"
430   OUTPUT 707;":TRIGGER:PATTERN 'PATT4','SCOUNT','65280'"
440   !
450   ! ***************** CONFIGURE SEQUENCE LEVEL 1 **************************
460   ! Store ANYSTATE in level 1 and Then find resource term "PATT1" once.
470   !
480   OUTPUT 707;":TRIGGER:FIND1 'PATT1',1,2"
490   !
500   ! ***************** CONFIGURE SEQUENCE LEVEL 2 **************************
510   ! Store ANYSTATE in level 2 and Then find resource term "PATT2" once.
520   !
530   OUTPUT 707;":TRIGGER:FIND2 'PATT2',1,3"
540   !
550   ! ***************** CONFIGURE SEQUENCE LEVEL 3 **************************
560   ! Store ANYSTATE in level 3 and Then find term "PATT3" once.
570   !
580   OUTPUT 707;":TRIGGER:FIND3 'PATT3',1,4"
590   !
```

```
600   ! ***************** CONFIGURE SEQUENCE LEVEL 4 *************************
610   ! Store ANYSTATE and then trigger on the combination terms (PATT3 or
620   ! PATT4) in level 4.
630   !
640   OUTPUT 707;":TRIGGER:FIND4 '(PATT3 OR PATT4)',1,TRIGGER"
650   !
660   ! ***************** SPECIFY OVERSAMPLE CLOCK ****************************
670   ! Specify an over-sample clock of 16.
680   !
690   OUTPUT 707;":TRIGGER:SAMPCLK 16"
700   !
710   ! ***************** START ACQUISITION ***********************************
720   ! Place the instrument in single run mode then determine when
730   ! the run is complete.
740   !
750   OUTPUT 707;":RMODE SINGLE"
760   OUTPUT 707;"*CLS"
770   OUTPUT 707;":SYSTEM:HEADER OFF"
780   OUTPUT 707;":SYSTEM:LONGFORM OFF"
790   Status=0
800   OUTPUT 707;":START"
810   !
820   ! ****************** CHECK FOR MEASUREMENT COMPLETE **********************
830   ! Enable the MESR register and query the register for a measurement
840   ! complete condition.
850   !
860   OUTPUT 707;"*WAI"
870   OUTPUT 707;":MESE2 1"
880   OUTPUT 707;":MESR2?"
890   ENTER 707;Status
900   PRINT Status
910   CLEAR SCREEN
920   IF Status=0 THEN GOTO 880
930   PRINT TABXY(30,15);"Measurement is complete"
940   !
```

```
950   ! ********** SELECT ONLY EXTERNALLY CLOCKED STATES FOR DISPLAY ***********
960   !
970   ! Use the SHOW command to display only the states that were externally
980   ! clocked in the oversampled mode.
990   !
1000  OUTPUT 707;":LIST:SHOW EXTCLK"
1010  !
1020  !
1030  ! ********************** VIEW THE RESULTS ****************************
1040  ! Display the State Listing and select a line number in the listing that
1050  ! allows you to see the beginning of the listing on the logic analyzer
1060  ! display.
1070  !
1080  OUTPUT 707;":LIST:COLUMN 1, 'SCOUNT', DECIMAL"
1090  OUTPUT 707;":MENU 2,3"
1100  OUTPUT 707;":LIST:LINE 0"
1110  !
1120  END
```

Transferring the logic analyzer configuration

This program uses the SYSTem:SETup query to transfer the configuration of the logic analyzer to your computer. This program also uses the SYSTem:SETup command to transfer a logic analyzer configuration from the computer back to the logic analyzer. The configuration data will set up the logic analyzer according to the data. It is useful for getting configurations for setting up the logic analyzer by the computer. This query differs from the SYSTem:DATA query because it only transfers the configuration and not the acquired data. The SYSTem:SETup command differs from the SYSTem:DATA command because it only transfers the configuration and not acquired data.

```
10    ! **************  SETUP QUERY PROGRAM EXAMPLE  *****************
20    !                    for the Agilent Technologies 16517A
30    !
40    !
50    ! Transfer the logic analyzer setup to the computer.
60    !
70    ! Select the slot number in which the logic analyzer is located.
80    !
90    !           ************ NOTE ***************
100   !           This program assumes the logic
110   !           analyzer is located in slot B.
120   !           ********************************
130   !
140   OUTPUT @Comm;":SELECT 2"
150   !
160   ! ********************************************************************
170   ! Setup the logic analyzer so that unnecessary information is not sent
180   ! to the computer.
190   !
200   OUTPUT 707;"*CLS"
210   OUTPUT 707;":EOI ON"
220   OUTPUT 707;":SYSTEM:LONGFORM OFF"
230   OUTPUT 707;":SYSTEM:HEAD OFF"
240   !
250   ! ********************************************************************
260   ! Dimension the strings needed for the Setup data.
```

```
270   !
280   DIM Block$[32000]
290   DIM Specifier$[2]
300   DIM Blocklength$[8]
310   !
320   ! ********************** SETUP QUERY *********************************
330   !
340   ! Send the SETUP Query and enter the query buffer data into the strings.
350   !
360   OUTPUT 707;":SYSTEM:SETUP?"
370   ENTER 707 USING "#,2A";Specifier$
380   ENTER 707 USING "#,8A";Blocklength$
390   ENTER 707 USING "-K";Block$
400   !
410   ! The SETUP query has been sent to the logic analyzer and the setup
420   ! string has been entered into the computer.
430   !
440   PRINT "The SETUP Block length is ";Blocklength$
450   !
460   ! ***********************************************************************
470   ! Convert the Blocklength string data to an integer.  This integer is
480   ! used to specify how may ASCII bytes will be sent back to the logic
490   ! analyzer when the SETUP command is issued in the next section of this
500   ! program.
510   !
520   Blocklength=IVAL(Blocklength$,10)
540   ! ********************** SETUP COMMAND *******************************
550   !
560   ! Return the setup data to the logic analyzer.
570   !
580   OUTPUT 707;":SELECT 2"
590   !
600   OUTPUT 707 USING "#,14A";":SYSTEM:SETUP "
610   OUTPUT 707 USING "#,2A";Specifier$
620   OUTPUT 707 USING "#,8A";Blocklength$
630   OUTPUT 707 USING ""&VAL$(Blocklength)&"A";Block$
640   !
650   END
```

Transferring the logic analyzer acquired data

This program uses the SYSTem:DATA query to transfer acquired data to your computer. It is useful for getting acquired data for setting up the logic analyzer by the computer at a later time. This query differs from the SYSTem:SETup query because it transfers only the acquired data.

This program also uses the SYSTem:DATA command to transfer the logic analyzer data from the computer back to the logic analyzer and load the analyzer with the acquired data. The SYSTem:DATA command differs from the SYSTem:SETup command because it transfers both the configuration and the acquired data.

> You should always precede the SYSTem:DATA query and command with the SYSTem:SETup query and command if the acquired data depends on a specific configuration. If you are only interested in the acquired data for post processing in the computer and the data is not dependent on the configuration, you can use the SYSTem:DATA query and command alone.

```
10    ! ****************** DATA COMMAND AND QUERY EXAMPLE *******************
20    !                           for the Agilent Technologies 16517A
30    !
40    ! ********************* CREATE TRANSFER BUFFER ************************
50    !
60    ASSIGN @Buff TO BUFFER [170000]
70    !
80    ! ***************** INITIALIZE GPIB DEFAULT ADDRESS *********************
90    !
100   REAL Address
110   Address=@Comm
120   ASSIGN @Comm TO Address
130   !
140   CLEAR SCREEN
150   !
```

```
160   ! ************* INTITIALIZE VARIABLE FOR NUMBER OF BYTES *****************
170   ! The variable "Numbytes" contains the number of bytes in the buffer.
180   !
190   REAL Numbytes
200   Numbytes=0
210   !
220   ! ************** RE-INITIALIZE TRANSFER BUFFER POINTERS ******************
230   !
240   CONTROL @Buff,3;1
250   CONTROL @Buff,4;0
260   !
270   ! ********************** SEND THE DATA QUERY *************************
280   OUTPUT @Comm;":SYSTEM:HEADER ON"
290   OUTPUT @Comm;":SYSTEM:LONGFORM ON"
300   OUTPUT @Comm;"SELECT 2"
310   OUTPUT @Comm;":SYSTEM:DATA?"
320   !
330   ! ******************** ENTER THE BLOCK DATA HEADER ********************
340   ! Enter the block data header in the proper format.
350   !
360   ENTER @Comm USING "#,B";Byte
370   PRINT CHR$(Byte);
380   WHILE Byte<>35
390     ENTER @Comm USING "#,B";Byte
400     PRINT CHR$(Byte);
410   END WHILE
420   ENTER @Comm USING "#,B";Byte
430   PRINT CHR$(Byte);
440   Byte=Byte-48
```

```
450   IF Byte=1 THEN ENTER @Comm USING "#,D";Numbytes
460   IF Byte=2 THEN ENTER @Comm USING "#,DD";Numbytes
470   IF Byte=3 THEN ENTER @Comm USING "#,DDD";Numbytes
480   IF Byte=4 THEN ENTER @Comm USING "#,DDDD";Numbytes
490   IF Byte=5 THEN ENTER @Comm USING "#,DDDDD";Numbytes
500   IF Byte=6 THEN ENTER @Comm USING "#,DDDDDD";Numbytes
510   IF Byte=7 THEN ENTER @Comm USING "#,DDDDDDD";Numbytes
520   IF Byte=8 THEN ENTER @Comm USING "#,DDDDDDDD";Numbytes
530   PRINT Numbytes
540   !
550   ! ******************* TRANSFER THE DATA ******************************
560   ! Transfer the data from the logic analyzer to the buffer.
570   !
580   TRANSFER @Comm TO @Buff;COUNT Numbytes,WAIT
590   !
600   ENTER @Comm USING "-K";Length$
610   PRINT "LENGTH of Length string is";LEN(Length$)
620   !
630   PRINT "**** GOT THE DATA ****"
640   PAUSE
650   ! ******************** SEND THE DATA *********************************
660   ! Make sure buffer is not empty.
670   !
680   IF Numbytes=0 THEN
690     PRINT "BUFFER IS EMPTY"
700     GOTO 1160
710   END IF
720   !
730   ! ********************* SEND THE DATA COMMAND *************************
740   ! Send the Setup command
750   !
760   OUTPUT @Comm USING "#,14A";":SYSTEM:DATA #"
770   PRINT "SYSTEM:DATA command has been sent"
780   PAUSE
790   !
```

```
800   ! ******************** SEND THE BLOCK DATA ***************************
810   ! Send the block data header to the 16517A in the proper format.
820   !
830   Byte=LEN(VAL$(Numbytes))
840   OUTPUT @Comm USING "#,B";(Byte+48)
850   IF Byte=1 THEN OUTPUT @Comm USING "#,A";VAL$(Numbytes)
860   IF Byte=2 THEN OUTPUT @Comm USING "#,AA";VAL$(Numbytes)
870   IF Byte=3 THEN OUTPUT @Comm USING "#,AAA";VAL$(Numbytes)
880   IF Byte=4 THEN OUTPUT @Comm USING "#,AAAA";VAL$(Numbytes)
890   IF Byte=5 THEN OUTPUT @Comm USING "#,AAAAA";VAL$(Numbytes)
900   IF Byte=6 THEN OUTPUT @Comm USING "#,AAAAAA";VAL$(Numbytes)
910   IF Byte=7 THEN OUTPUT @Comm USING "#,AAAAAAA";VAL$(Numbytes)
920   IF Byte=8 THEN OUTPUT @Comm USING "#,AAAAAAAA";VAL$(Numbytes)
930   !
940   ! ********************** SAVE BUFFER POINTERS ************************
950   ! Save the transfer buffer pointer so it can be restored after the
960   ! transfer.
970   !
980   STATUS @Buff,5;Streg
990   !
1000  ! ****************** TRANSFER DATA TO THE 16517 ********************
1010  ! Transfer the data from the buffer to the 16517A.
1020  !
1030  TRANSFER @Buff TO @Comm;COUNT Numbytes,WAIT
1040  !
1050  ! ********************** RESTORE BUFFER POINTERS ***********************
1060  ! Restore the transfer buffer pointer
1070  !
1080  CONTROL @Buff,5;Streg
1090  !
1100  ! ******************** SEND TERMINATING LINE FEED **********************
1110  ! Send the terminating linefeed to properly terminate the data string.
1120  !
1130  OUTPUT @Comm;""
1140  !
1150  PRINT "**** SENT THE DATA ****"
1160  END
```

Checking for measurement completion

This program can be appended to or inserted into another program when you need to know when a measurement is complete. If it is at the end of a program it will tell you when measurement is complete. If you insert it into a program, it will halt the program until the current measurement is complete.

This program is also in the state analyzer example program in the "Making a State Analyzer Measurement" topic earlier in this chapter. It is included in the state analyzer example program to show how it can be used in a program to halt the program until measurement is complete.

```
300   ! ***************** MEASUREMENT COMPLETE EXAMPLE ***********************
310   !                 for the Agilent Technologies 16517A Logic Analyzer
320   !
330   ! This program can be appended to or inserted into another program when
340   ! you need to know when a measurement is complete.  If it is at the end
350   ! of a program it will tell you when measurement is complete.  If you
360   ! insert it into a program, it will halt the program until the current
370   ! measurement is complete.
380   !
390   ! In this example, the module installed in slot B is being checked for
400   ! measurement complete.
410   !
420   ! ****************** CHECK FOR MEASUREMENT COMPLETE **********************
430   ! Enable the MESR register and query the register for a measurement
440   ! complete condition.
450   !
460   OUTPUT 707;":SYSTEM:HEADER OFF"
470   OUTPUT 707;":SYSTEM:LONGFORM OFF"
480   !
490   Status=0
500   OUTPUT 707;":MESE2 1"
510   OUTPUT 707;":MESR2?"
520   ENTER 707;Status
530   !
```

```
540   ! Print the MESR register status.
550   !
560   CLEAR SCREEN
570   ! Repeat the MESR query until measurement is complete.
580   WAIT 1
590   IF Status=0 THEN GOTO 500
600   PRINT TABXY(30,15);"Measurement is complete"
610   !
620   END
```

Making a Compare Measurement

This program example acquires a state listing, copies the listing to the compare reference listing, acquires another state listing, and compares both listings to find differences.

This program is written in such a way you can run it with the Agilent Technologies E2433-60006 Logic Analyzer Training Board.

```
10    ! *********** STATE COMPARE EXAMPLE *******************************
20    !            for the Agilent Technologies 16517A Logic Analyzer
30    !
40    !
50    !************** SELECT THE 16517A MODULE ***********************
60    ! Select the module slot in which the 16517A is installed.  In this
70    ! example, the 16517A is in slot B of the mainframe.
80    !
90    OUTPUT 707;":SELECT 2"
100   OUTPUT 707;":SYSTEM:HEADER OFF"
110   OUTPUT 707;":SYSTEM:LONGFORM OFF"
120   !
130   !************** CONFIGURE THE STATE ANALYZER ***********************
140   ! Configure the 16517A as a state analyzer.
150   !
160   OUTPUT 707;":FORMAT:TYPE STATE"
170   !
180   ! ***************************************************************
190   ! Remove all labels previously set up, make a label "SCOUNT," specify
200   ! positive logic, and assign the 8 bits to both pods of the label.
210   !
220   OUTPUT 707;":FORMAT:REMOVE ALL"
230   OUTPUT 707;":FORMAT:LABEL 'SCOUNT', POS, 255,255"
240   !
250   ! ***************************************************************
260   ! Specify the falling edge for the clock.
270   !
280   OUTPUT 707;":FORMAT:CLOCK FALLING"
290   !
300   ! ***************************************************************
310   ! Specify two sequence levels, the trigger sequence level, specify
320   ! 4096 for the PATT1 term which will be the trigger term, and store
330   ! no states until the trigger is found.
340   !
```

```
350   OUTPUT 707;":TRIGGER:SEQUENCE 2"
360   OUTPUT 707;":TRIGGER:PATTERN 'PATT1','SCOUNT','4096'"
370   OUTPUT 707;":TRIGGER:FIND1 'ANYSTATE', 2, 2"
380   OUTPUT 707;":TRIGGER:FIND2 'PATT1',1,TRIGGER"
390   OUTPUT 707;":MENU 2,1"
400   !
410   ! ****************************************************************
420   ! Change the displayed menu to the state listing and start the state
430   ! analyzer in repetitive mode.
440   !
450   OUTPUT 707;":MENU 2,3"
460   OUTPUT 707;":RMODE REPETITIVE"
470   OUTPUT 707;":START"
480   !
490   ! ****************************************************************
500   ! The logic analyzer is now running in the repetitive mode
510   ! and will remain in repetitive until the STOP command is sent.
520   !
530   PRINT "The logic analyzer is now running in the repetitive mode"
540   PRINT "and will remain in repetitive until the STOP command is sent."
550   PRINT
560   PRINT "Press CONTINUE"
570   PAUSE
580   !
590   !****************************************************************
600   ! Stop the acquisition and copy the acquired data to the compare reference
610   ! listing.
620   !
630   OUTPUT 707;":STOP"
640   OUTPUT 707;":MENU 2,5"
650   OUTPUT 707;":COMPARE:MENU REFERENCE"
660   OUTPUT 707;":COMPARE:COPY"
670   !
680   ! The logic analyzer acquistion is now stopped, the Compare menu
690   ! is displayed, and the data is now in the compare reference
700   ! listing.
710   !
720   !****************************************************************
730   ! Display line 32767 of the compare listing and start the analyzer
740   ! in a repetitive mode.
750   !
760   OUTPUT 707;":COMPARE:LINE 32767"
770   OUTPUT 707;":START"
780   !
```

```
790   ! Line 32767 of the listing is now displayed at center screen
800   ! in order to show the last four states acquired.  In this
810   ! example, the last four states are stable.  However, in some
820   ! cases, the end points of the listing may vary thus causing
830   ! a false failure in compare.  To eliminate this problem, a
840   ! partial compare can be specified to provide predicable end
850   ! points of the data.
860   !
870   PRINT "Press CONTINUE to send the STOP command."
880   PAUSE
890   OUTPUT 707;":STOP"
900   !
910   !************************************************************************
920   ! The end points of the compare can be fixed to prevent false failures.
930   ! In addition, you can use partial compare to compare only sections
940   ! of the state listing you are interested in comparing.
950   !
960   OUTPUT 707;":COMPARE:RANGE PARTIAL, 0, 508"
970   !
980   ! The compare range is now from line 0 to +508
990   !
1000  !************************************************************************
1010  ! At this point another system-under-test would be connected so that new
1020  ! data would be acquired and compared to see which states are different.
1030  ! If you are using the E2433-60006 Training Kit, you can simulate a
1040  ! different system-under-test by disconnecting the clock lead and
1050  ! reacquiring data.
1060  !
1070  PRINT "Disconnect the clock lead from the training board so that the"
1080  PRINT "data changes, reacquire the data and compare which states are
different."
1090  !
1100  PRINT "Press CONTINUE when you have finished disconnecting the clock."
1110  !
1120  PAUSE
1130  !
1140  !************************************************************************
1150  ! Start the logic analyzer to acquire new data and then stop it to compare
1160  ! the data.  When the acquistion is stopped, the Compare Listing Menu will
1170  ! be displayed.
1180  !
1190  OUTPUT 707;":START"
1200  OUTPUT 707;":STOP"
1210  OUTPUT 707;":MENU 2,5"
1220  !
```

```
1230  !*************************************************************************
1240  ! Dimension strings in which the compare find query (COMPARE:FIND?)
1250  ! enters the line numbers and error numbers.
1260  !
1270  DIM Line$[20]
1280  DIM Error$[4]
1290  DIM Comma$[1]
1300  !
1310  ! *************************************************************************
1320  ! Display the Difference listing.
1330  !
1340  OUTPUT 707;":COMPARE:MENU DIFFERENCE"
1350  !
1360  ! *************************************************************************
1370  ! Loop to query all 508 possible errors.
1380  !
1390  FOR Error=1 TO 508
1400  !
1410  ! Read the compare differences
1420  !
1430  OUTPUT 707;":COMPARE:FIND? "&VAL$(Error)
1440  PRINT Error
1450  ! *************************************************************************
1460  ! Format the Error$ string data for display on the controller screen.
1470  !
1480  IF Error>99 THEN GOTO 1620
1490  IF Error>9 THEN GOTO 1590
1500  !
1510  ENTER 707 USING "#,1A";Error$
1520  ENTER 707 USING "#,1A";Comma$
1530  ENTER 707 USING "K";Line$
1540  Error_return=IVAL(Error$,10)
1550  IF Error_return=0 THEN GOTO 1860
1560  !
1570  GOTO 1650
1580  !
1590  ENTER 707 USING "#,3A";Error$
1600  ENTER 707 USING "K";Line$
1610  GOTO 1650
1620  !
1630  ENTER 707 USING "#,4A";Error$
1640  ENTER 707 USING "K";Line$
1650  !
```

```
1660  ! ************************************************************************
1670  ! Test for the last error.  The error number of the last error is the same
1680  ! as the error number of the first number after the last error.
1690  !
1700  Error_line=IVAL(Line$,10)
1710  IF Error_line=Error_line2 THEN GOTO 1820
1720  Error_line2=Error_line
1730  !
1740  ! ************************************************************************
1750  ! Print the error numbers and the corresponding line numbers on the
1760  ! controller screen.
1770  !
1780  PRINT "Error number ",Error," is on line number ",Error_line
1790  !
1800  NEXT Error
1810  !
1820  PRINT
1830  PRINT
1840  PRINT "Last error found"
1850  GOTO 1890
1860  PRINT "No errors found"
1870  !
1880  !*****************************************************************
1890  !
1900  END
```

Using the COMPare:DATA? Query

This program example shows how to use the COMPare:DATA? query to print the compare reference listing.

```
10 DIM Label$[6], Response$[80]
15 PRINT "This program shows the values for a signal's Compare listing"
20 INPUT "Enter signal label: ", Label$
25 OUTPUT XXX;":SYSTEM:HEADER OFF"      !Turn headers off (from responses)
30 OUTPUT XXX;":COMPARE:RANGE?"
35 ENTER XXX; First, Last  !Read in the range's end-points
40 PRINT "LINE ", "VALUE of "; Label$
45 FOR State = First TO Last       !Print compare value for each state
50   OUTPUT XXX;"   :COMPARE:DATA? '" & Label$ & "'," & VAL$(State)
55   ENTER XXX; Response$
60   PRINT State, Response$
65   NEXT State
70 END
```

Index

A
ACCumulate command/query, 4–9
ACQuisition command/query, 3–11, 4–10, 5–9
ARMedby command/query, 3–11

B
BASE command, 7–5
Block data, 8–4
Block length specifier, 8–4
Block length specifier>, 8–13
Block length specifier>>, 8–4
BRANch command/query, 3–12 to 3–13

C
CARDcage query, 1–5
CENTer command, 4–10
CLEar command, 6–5
CLOCk command/query, 2–5
CLRPattern command, 4–11, 5–10
CLRStat command, 4–11, 5–10
CMASk command/query, 6–5
COLumn command/query, 5–11
command
  ACCumulate, 4–9
  ACQuisition, 3–11, 4–10, 5–9
  ARMedby, 3–11
  BASE, 7–5
  BRANch, 3–12
  CENTer, 4–10
  CLEar, 6–5
  CLOCk, 2–5
  CLRPattern, 4–11, 5–10
  CLRStat, 4–11, 5–10
  CMASk, 6–5
  COLumn, 5–11
  COMPare, 6–4
  COPY, 6–6
  DATA, 6–7, 8–4
  DELay, 4–12
  DURation, 3–15
  EDGE, 3–17
  FIND, 3–18
  FORMat, 2–4
  INSert, 4–13
  LABel, 2–6, 4–14
  LINE, 5–12, 6–9
  LIST, 5–8
  MENU, 1–6, 6–10
  MESE, 1–12
  MINus, 4–15
  MMODe, 4–16, 5–13
  OCONdition, 4–17, 5–14
  OPATtern, 4–18, 5–15
  OSEarch, 4–19, 5–16
  OTAG, 5–17
  OTIMe, 4–20, 5–18
  OVERlay, 4–21, 5–19
  PATTern, 3–20, 7–6
  PLUS, 4–22
  PRINt, 1–7
  RANGe, 4–23, 6–10, 7–6
  REMove, 2–7, 4–23, 5–19, 7–7
  REName, 3–21
  RMODe, 1–7
  RUNTil, 4–24, 5–20, 6–11
  SAMPclk, 3–22, 4–25, 5–21
  SELect, 1–3, 1–6
  SEQuence, 3–23
  SET, 6–12
  SETup, 8–13
  SETUPHOLDA, 3–24
  SETUPHOLDB, 3–25
  SETUPHOLDC, 3–26
  SHOW, 5–22
  SIZE, 4–26
  SOFFset, 2–8, 4–27, 5–23
  SPERiod, 3–27, 4–28, 5–24
  STARt, 1–6
  STOP, 1–7
  SYMBol, 7–4
  SYSTem:DATA, 8–2, 8–4
  SYSTem:PRINt, 1–7
  SYSTem:SETup, 8–2, 8–13
  THReshold, 2–9
  TIMER, 3–28
  TPOSition, 3–29, 4–30, 5–26
  TYPE, 2–10
  WIDTh, 7–8
  XCONdition, 4–32, 5–28
  XPATtern, 4–33, 5–30
  XSEarch, 4–34, 5–31
  XTAG, 5–33
  XTIMe, 4–35, 5–34
Command Set Organization, 1–8 to 1–10
compare measurement
  program example, 9–17
COMPare selector, 6–4
COMPare Subsystem, 6–1, 6–3 to 6–12
COMPare:DATA? query
  program example, 9–22
Complex qualifier, 3–9
COPY command, 6–6

D
DATA, 8–4
Data and Setup Commands, 8–1, 8–3 to 8–14
Data block
  Data preamble, 8–6
  Section data, 8–6
  Section header, 8–6
DATA command/query, 6–7
Data preamble, 8–6 to 8–9
DATA query, 5–12
DELay command/query, 4–12
DURation command/query, 3–15

E
EDGE command/query, 3–17
Examples
  program, 9–2

F
FIND query, 6–8
FORMat selector, 2–4

I
INSert command, 4–13
INTermodule Subsystem, 1–7

L
LABel command/query, 2–6, 4–14
LINE command/query, 5–12, 6–9
LIST selector, 5–8
LIST Subsystem, 5–1, 5–3 to 5–34

M
measurement complete program example, 9–15
MENU, 1–6
MENU command, 6–10
MESE command/query, 1–12
MESR query, 1–14

MINus command/query, 4–15
MMEMory Subsystem, 1–7
MMODe command/query, 4–16, 5–13
Module Status Reporting, 1–11

O
OCONdition command/query, 4–17, 5–14
OPATtern command/query, 4–18, 5–15
OSEarch command/query, 4–19, 5–16
OSTate query, 5–17
OTAG command/query, 5–17
OTIMe command/query, 4–20, 5–18
OVERlay command, 4–21, 5–19

P
PATTern command, 7–6
PATTern command/query, 3–20
PLUS command, 4–22
Preamble description, 8–6
program example
  checking for measurement completion, 9–15
  compare measurement, 9–17
  COMPare:DATA? query, 9–22
  state analyzer, 9–5
  SYSTem:DATA command, 9–11
  SYSTem:DATA query, 9–11
  SYSTem:SETup command, 9–9
  SYSTem:SETup query, 9–9
  timing analyzer, 9–3
  transferring configuration to analyzer, 9–9
  transferring configuration to the 9–9
  transferring setup and data to the analyzer, 9–11
  transferring setup and data to the computer, 9–11
Program Examples, 9–2

Q
Query
  ACCumulate, 4–9
  ACQuisition, 3–11, 4–10, 5–9
  ARMedby, 3–12
  BRANch, 3–14
  CARDcage, 1–5
  CLOCk, 2–5
  CMASk, 6–6
  COLumn, 5–11
  DATA, 5–12, 6–8, 8–5
  DELay, 4–12
  DURation, 3–16
  EDGE, 3–17
  ERRor, 1–7
  FIND, 3–19, 6–8
  LABel, 2–7
  LINE, 5–13, 6–9
  MENU, 1–6
  MESE, 1–12
  MESR, 1–14
  MMODe, 4–16, 5–13
  OCONdition, 4–17, 5–14
  OPATtern, 4–18, 5–15
  OSEarch, 4–19, 5–16
  OSTate, 5–17
  OTIMe, 4–20, 5–18
  PATTern, 3–21
  PRINt, 1–7
  RANGe, 4–23, 6–11
  REName, 3–21
  RMODe, 1–7
  RUNTil, 4–25, 5–21, 6–12
  SEQuence, 3–23
  SETup, 8–14
  SHOW, 5–22
  SOFFset, 2–8, 4–27, 5–23
  SPERiod, 3–27, 4–28, 5–24
  SYNC, 2–9
  SYSTem:DATA, 8–5
  SYSTem:ERRor, 1–7
  SYSTem:PRINt, 1–7
  SYStem:SETup, 8–14
  TAVerage, 4–29, 5–25
  Threshold, 2–10
  TIMER, 3–28
  TMAXimum, 4–29, 5–25
  TMINimum, 4–30, 5–26
  TPOSition, 3–30, 4–31, 5–27
  TYPE, 2–10
  VRUNs, 4–31, 5–28
  XCONdition, 4–32, 5–29
  XOTag, 5–29
  XOTime, 4–33, 5–30
  XPATtern, 4–34, 5–31
  XSEarch, 4–35, 5–32
  XSTate, 5–32
  XTAG, 5–33
  XTIMe, 4–35, 5–34

R
RANGe command, 7–6
RANGe command/query, 4–23, 6–10
REMove command, 2–7, 4–23, 5–19, 7–7
REName command/query, 3–21
RMODe, 1–7
RUNTil command/query, 4–24, 5–20, 6–11

S
SAMPclk command/query, 3–22, 4–25,
Section data, 8–6
Section data format, 8–4
Section header, 8–6
SELect, 1–6
SELect command, 1–3
SET command, 6–12
SETup, 8–13
SETUPHOLDA command, 3–24
SETUPHOLDB command, 3–25
SETUPHOLDC command, 3–26
SHOW command/query, 5–22
SIZE command/query, 4–26
SOFFset command/query, 2–8, 4–27, 5–23
SPERiod command/query, 3–27, 4–28,
STARt, 1–6
state analyzer
computer program example, 9–5
STOP, 1–7
Subsystem
  COMPare, 6–2
  LIST, 5–1, 5–3 to 5–34
  SYMBol, 7–1, 7–3 to 7–8
  TRIGger/TTRace, 3–1, 3–3 to 3–30
  WAVeform, 4–1, 4–3 to 4–35
SYMBol selector, 7–4
SYMBol Subsystem, 7–1, 7–3 to 7–8
SYNC command/query, 2–9
Syntax Diagram
  COMPare Subsystem, 6–3
  FORMat Subsystem, 2–3
  LIST Subsystem, 5–3
  SYMBol Subsystem, 7–3
  TRIGger Subsystem, 3–3

WAVeform Subsystem, 4–4 to 4–6
SYSTem:DATA, 8–4 to 8–5
SYSTem:DATA command program example, 9–11
SYSTem:DATA query program example, 9–11
SYSTem:ERRor, 1–7
SYSTem:PRINt, 1–7
SYStem:SETup, 8–13 to 8–14
SYSTem:SETup command program example, 9–9
SYSTem:SETup query program example, 9–9

T
TAVerage query, 4–29, 5–25
THReshold command/query, 2–9
timing analyzer
program example, 9–3
TMAXimum query, 4–29, 5–25
TMINimum query, 4–30, 5–26
TPOSition command/query, 3–29 to 3–30, 4–30, 5–26 to 5–27
TRIGger/TTRace Subsystem, 3–1, 3–3 to 3–30
TYPE command/query, 2–10

V
VRUNs query, 4–31, 5–28

W
WAVeform selector, 4–8
WIDTh command, 7–8

X
XCONdition command/query, 4–32, 5–28
XOTag query, 5–29
XOTime query, 4–33, 5–30
XPATtern command/query, 4–33, 5–30
XSEarch command/query, 4–34, 5–31
XSTate query, 5–32
XTAG command/query, 5–33
XTIMe command/query, 4–35, 5–34

© Copyright Agilent Technologies, 1987-2000
All Rights Reserved.

Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Document Warranty

The information contained in this document is subject to change without notice.

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

Agilent Technologies shall not be liable for errors contained herein or for damages in connection with the furnishing, performance, or use of this material.

Safety

This apparatus has been designed and tested in accordance with IEC Publication 348, Safety Requirements for Measuring Apparatus, and has been supplied in a safe condition. This is a Safety Class I instrument (provided with terminal for protective earthing). Before applying power, verify that the correct safety precautions are taken (see the following warnings). In addition, note the external markings on the instrument that are described under "Safety Symbols."

Warning

- Before turning on the instrument, you must connect the protective earth terminal of the instrument to the protective conductor of the (mains) power cord. The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. You must not negate the protective action by using an extension cord (power cable) without a protective conductor (grounding). Grounding one conductor of a two-conductor outlet is not sufficient protection.
- Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuseholders. To do so could cause a shock of fire hazard.
- Service instructions are for trained service personnel. To avoid dangerous electric shock, do not perform any service unless qualified to do so. Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.
- If you energize this instrument by an auto transformer (for voltage reduction), make sure the common terminal is connected to the earth terminal of the power source.
- Whenever it is likely that the ground protection is impaired, you must make the instrument inoperative and secure it against any unintended operation.
- Do not operate the instrument in the presence of flammable gasses or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.
- Do not install substitute parts or perform any unauthorized modification to the instrument.
- Capacitors inside the instrument may retain a charge even if the instrument is disconnected from its source of supply.
- Use caution when exposing or handling the CRT. Handling or replacing the CRT shall be done only by qualified maintenance personnel.

Safety Symbols



Instruction manual symbol: the product is marked with this symbol when it is necessary for you to refer to the instruction manual in order to protect against damage to the product.



Hazardous voltage symbol.



Earth terminal symbol: Used to indicate a circuit common connected to grounded chassis.

WARNING

The Warning sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a Warning sign until the indicated conditions are fully understood and met.

CAUTION

The Caution sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a Caution symbol until the indicated conditions are fully understood or met.

Agilent Technologies
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901

Product Warranty

This Agilent Technologies product has a warranty against defects in material and workmanship for a period of one year from date of shipment. During the warranty period, Agilent Technologies will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies.

For products returned to Agilent Technologies for warranty service, the Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to the Buyer. However, the Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

Agilent Technologies warrants that its software and firmware designated by Agilent Technologies for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent Technologies does not warrant that the operation of the instrument software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

No other warranty is expressed or implied. Agilent Technologies specifically disclaims the implied warranties of merchantability or fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are the buyer's sole and exclusive remedies. Agilent Technologies shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products.

For any assistance, contact your nearest Agilent Technologies Sales Office.

Certification

Agilent Technologies certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

About this edition

This is the second edition of the *Agilent Technologies 16517A/18A Programmer's Guide.*

Publication number
16517-97007, February 2000

Printed in USA.

Previous editions:

16517-97000, July 1993

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by you. The dates on the title page change only when a new edition is published.

A software or firmware code may be printed before the date. This code indicates the version level of the software or firmware of this product at the time the manual or update was issued. Many product updates do not require manual changes; and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

The following list of pages gives the date of the current edition and of any changed pages to that edition.

All pages original edition

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

Gold-standard solutions

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

We buy equipment

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

Learn more!

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

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

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

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

ARTISAN® TECHNOLOGY GROUP