Keysight 86100A/B/C/D Wide–Bandwidth Oscilloscope

(for programming the legacy instrument GUI. To program the FlexDCA GUI, refer to the 86100D help.)

KEYSIGHT
TECHNOLOGIES

Programmer's Guide

Notices

Manual Part Number

Edition

Warranty

Technology Licenses

Restricted Rights Legend

Safety Notices

CAUTION

A **CAUTION** notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in damage to the product or loss of important data. Do not proceed beyond a **CAUTION** notice until the indicated conditions are fully understood and met.

WARNING

A WARNING notice denotes a hazard. It calls attention to an operating procedure, practice, or the like that, if not correctly performed or adhered to, could result in personal injury or death. Do not proceed beyond a WARNING notice until the indicated conditions are fully understood and met.

Contents

1 Introduction

Who Should Use This Book

Use this book if you are programming an 86100D that is running the instrument's legacy GUI. To program an 86100D that is running the FlexDCA GUI, *do not use the commands in this book*. Instead, with the 86100D FlexDCA GUI displayed click Help > Contents. Then, click the "Programming" link to learn about the programming commands for FlexDCA.

Table 1   86100D GUIs

86100D Running Legacy GUI	86100D Running FlexDCA GUI

> **NOTE**   If you are using FlexDCA on a PC to control the 86100D with the legacy GUI, *do not use this book*. Instead send FlexDCA programming commands to FlexDCA and let FlexDCA control the 86100D. Use the commands documented in FlexDCA's help system.

KEYSIGHT
TECHNOLOGIES

Supported Firmware Versions

This edition of this book documents remote control of the instruments shown in the following table.

Table 2   Supported Instruments

Instrument	Firmware Vision
86100D	A.13.00 and below
86100C	A.10.80 and below
86100B	A.05.00 and below
86100A	A.05.00 and below

NOTE   Starting with firmware version A.12.00, the information in this book applies to an 86100D that is running in Legacy configuration. The instrument can also be operated in Standard configuration, in which case you must use the remote commands that are documented in the 86100D online help. Refer to the online help for information on the different 86100D configurations.

IEEE 488.2 (SCPI)

The programming syntax documented in this book conforms to the IEEE 488.2 Standard Digital Interface for Programmable Instrumentation and to the Standard Commands for Programmable Instruments (SCPI). For a listing of commands that are new or revised, refer to "New and Revised Commands" on page 9. If you are unfamiliar with programming instruments using the SCPI standard, refer to "Command Syntax" on page 16. For more detailed information regarding the GPIB, the IEEE 488.2 standard, or the SCPI standard, refer to the following books:

· International Institute of Electrical and Electronics Engineers. *IEEE Standard 488.1-1987, IEEE Standard Digital Interface for Programmable Instrumentation.* New York, NY, 1987.

· International Institute of Electrical and Electronics Engineers. *IEEE Standard 488.2-1987, IEEE Standard Codes, Formats, Protocols and Common commands For Use with ANSI/IEEE Std 488.1-1987.* New York, NY, 1987.

You can configure the instrument and transfer data between the instrument and a computer using GPIB (General Purpose Interface Bus) connection or SICL/LAN connection (firmware revision A08.00 and above).

SICL/LAN Support

The ability to control the instrument over SICL/LAN is a new feature introduced with revision A.08.00. For SICL/LAN support, use the Keysight IO Libraries Suite which is shipped on a disc with the instrument. This software includes the Keysight Connection Expert, which facilitates the sending of remote commands to the instrument by using a LAN device address. If you can not establish a LAN connection on the instrument, install the Keysight IO Libraries LAN patch. This patch is located on the instrument at C:\InfiniiumInstaller\AgtInstIoLanPatch.msi.

An IP address can be substituted instead of using domain names.

To create the device address within the Keysight Connection Expert,

1   Locate the instrument device address, which should look similar to the following examples:

    TCPIP0::10.0.0.5::inst0::INSTR

    TCPIP0::YourInstrument.YourDomain::inst0::INSTR

2   Right-click the instrument device address to view the shortcut menu and select Change Properties.

3   In the Advanced section, change the remote instrument name to gpib0,7. The device address should now be:

    TCPIP0::10.0.0.5::gpib0,7::INSTR

After configuring the Keysight Connection Expert with the above steps, sending commands to the instrument changes the instrument from local mode into remote mode, which is similar to GPIB control. If, however, the device address inst0 is used instead of gpib0,7 the instrument will not change from local to the remote mode and some dialog boxes may be presented during the SICL/LAN session that requires front-panel operation.

SICL/LAN support requires that two programs be unblocked by the instrument's firewall. If you upgraded the instrument firmware versions A.07.00 and below to revision A.08.00 and above, you might be prompted by a firewall application to block the Keysight Remote I/O Port Mapper Utility and the Keysight Remote I/O Server. If you decide to allow the features to be blocked, then remote control of the DCA over SICL/LAN will not be possible. We recommend that you select Unblock on these features. However, if you block these features, you can always reconfigure the firewall at a later time to allow SICL/LAN.

Some firewall applications might block an echo request (ping) from the Keysight Connection Expert version 15.0 and above. If a ping is blocked the "Instrument I/O on this PC" auto-detect function will not find the instrument even though it has been added and tested correctly under the Change Properties dialog box. To resolve this on the Microsoft Windows Firewall, .

For more information on communicating with the instrument using the Keysight's IO Libraries Suite, refer to the book *IO Libraries Suite Connectivity Guide with Getting Started*.

To upgrade instrument software

After you have obtained the software upgrade file for your instrument, perform the following steps to install the upgrade.

1   Copy the software upgrade file to a USB Flash Drive, external USB CD-RW drive, LAN folder, or other device so that the file will be available to copy to the instrument.

2   On the instrument's File menu, click Exit and then click Yes to exit the application.

3   On the Windows Start menu, click My Computer.

4   Select the D: drive and create a new folder. Give the new folder a meaningful name. For example, Software Upgrade.

5   Copy the upgrade file (.exe file extension) from an external memory device to your new folder.

6 Select the upgrade file to begin the installation. Click Next twice for the installation wizard to automatically uninstall the current version and install the newer version.

7 If you are prompted by a firewall application to block the Keysight Remote I/O Port Mapper Utility and the Keysight Remote I/O Server, select Unblock as shown in Figure 1 on page 10. See the introduction to this section for more information.

8 On the Windows desktop, double click the program icon to start the instrument.

Agilent Remote I/O Port Mapper Utility   Agilent Remote I/O Server
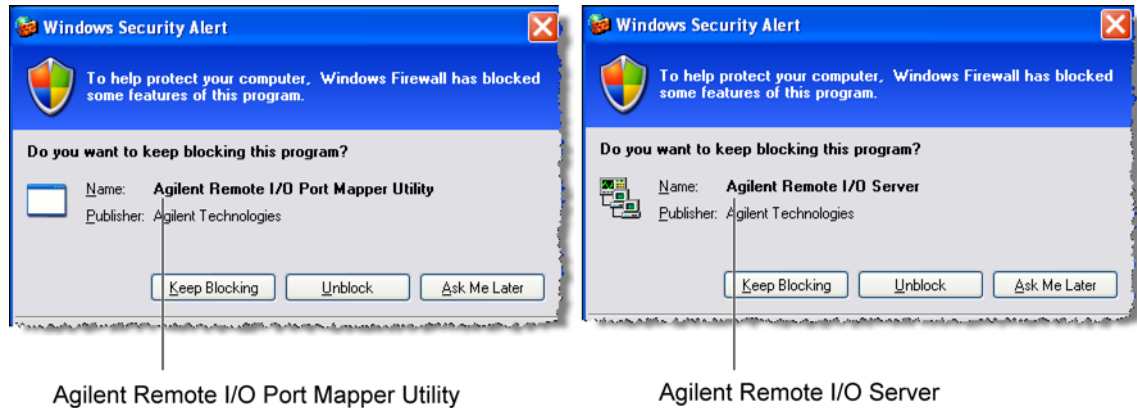
Figure 1   Example Windows Firewall Security Alerts

To configure the firewall

This procedure applies to instrument software revision A.08.00 and above. Although it describes the settings for the Windows Firewall, settings using a different firewall will be similar. These settings allow control of the instrument over SICL/LAN and allow the Keysight Connection Expert to locate the instrument.

1 On the instrument, click Help > About 86100C/D and confirm that software revision A.08.00 or above is installed.

2 Minimize the 86100C/D application to view the Windows desktop.

3 On the Start menu, click Control Panel.

4 If Category View is set, click Switch to Classic View.

5 Open Windows Firewall.

6 On the Exceptions tab, clear or select to unblock (allow) the Keysight Remote I/O Port Mapper Utility and the Keysight Remote I/O Server. These programs allow control of the instrument over SICL/LAN. If these utilities are not listed, click Add Program in the dialog box and add them using the following paths:

Keysight Remote I/O Port Mapper Utility found at C:\Program Files\Keysight\IO Libraries Suite\bin\portmap.exe

Keysight Remote I/O Server found at C:\Program Files\Keysight\IO Libraries Suite\bin\siclland.exe
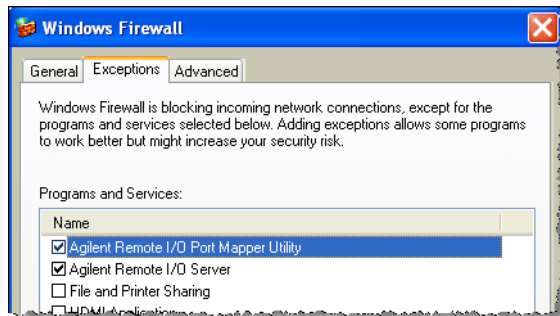
Figure 2   86100C/D SICL/LAN Programs

7   On the Windows Firewall, click the Advanced tab.

8   Click ICMP to open the ICMP Settings dialog box.

9   Clear or select Allow incoming echo request. Selecting this feature allows the Keysight Connection Expert's (version 15.0 and above) Instrument I/O on this PC to automatically find the instrument.
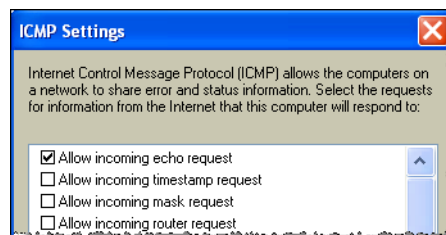
Figure 3   Allow Incoming Echo Request

Examples

Throughout this book, BASIC and ANSI C are used in the examples of individual commands. If you are using other languages, you will need to find the equivalents of BASIC commands like OUTPUT, ENTER, and CLEAR, to convert the examples. The instrument's GPIB address is configured at the factory to a value of 7. You must set the output and input functions of your programming language to send the commands to this address. You can change the GPIB address from the instrument's front panel.

Measurement Process

Figure 4 is a instrument block diagram that shows where the measurements are made on the acquired data and when the post-signal processing is applied to the data. The diagram is laid out serially for a visual perception of how the data is affected by the instrument.
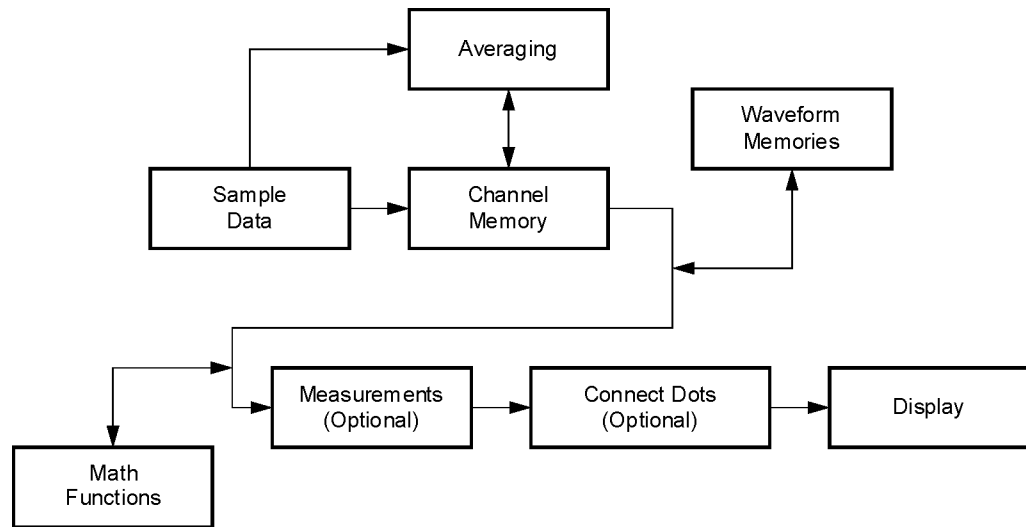
54800b01

Figure 4   Sample Data Processing

The sample data is stored in the channel memory for further processing before being displayed. The time it takes for the sample data to be displayed depends on the number of post processes you have selected. Averaging your sampled data helps remove any unwanted noise from your waveform.

You can store your sample data in the instrument's waveform memories for use as one of the sources in Math functions or to visually compare against a waveform that is captured at a future time. The Math functions allow you to apply mathematical operations on your sampled data. You can use these functions to duplicate many of the mathematical operations that your circuit may be performing to verify that your circuit is operating correctly. The measurements section performs any of the automated measurements that are available in the instrument. The measurements that you have selected appear at the bottom of the display. The Connect Dots section draws a straight line between sample data points, giving an analog look to the waveform. This is sometimes called linear interpolation.

The Command Tree

The command tree refers to the relationship of the commands to each other. The IEEE 488.2 common commands do not affect the position of the parser within the tree. A leading colon or a program message terminator (<NL> or EOI true on the last byte) places the parser at the root of the command tree. A leading colon is a colon that is the first character of a program header. Executing a subsystem command places you in that subsystem until a leading colon or a program message terminator is found. The commands in this instrument can be placed into three types: common commands, root level commands, and subsystem commands.

- Common commands (defined by IEEE 488.2) control functions that are common to all IEEE 488.2 instruments. These commands are independent of the tree and do not affect the position of the parser within the tree. *RST is an example of a common command.

- Root level commands control many of the basic functions of the instrument. These commands reside at the root of the command tree. They can always be parsed if they occur at the beginning of a program message or are preceded by a colon. Unlike common commands, root level commands place the parser back at the root of the command tree. AUTOSCALE is an example of a root level command.

- Subsystem commands are grouped together under a common node of the command tree, such as the TIMEBASE commands. Only one subsystem may be selected at a given time. When the instrument is initially turned on, the command parser is set to the root of the command tree and no subsystem is selected.

Command headers are created by traversing down the command tree. A legal command header from the command tree would be :TIMEBASE:RANGE. It consists of the subsystem followed by a command separated by colons. The compound header contains no spaces.

In the command tree, use the last mnemonic in the compound header as a reference point (for example, RANGE). Then find the last colon above that mnemonic (TIMEBASE:). That is the point where the parser resides. Any command below this point can be sent within the current program message without sending the mnemonics which appear above them (for example, REFERENCE).

Use a colon to separate two commands in the same subsystem.

```
OUTPUT 707;":CHANNEL1:RANGE 0.5;OFFSET 0"
```

The colon between CHANNEL1 and RANGE is necessary because CHANNEL1:RANGE specifies a command in a subsystem. The semicolon between the RANGE command and the OFFSET command is required to separate the two commands. The OFFSET command does not need CHANNEL1 preceding it because the CHANNEL1:RANGE command sets the parser to the CHANNEL1 node in the tree.

Command Syntax

In accordance with IEEE 488.2, the instrument's commands are grouped into "subsystems." Commands in each subsystem perform similar tasks. Starting with Chapter 5, "System Commands" each chapter covers a separate subsystem.

Sending a Command

It's easy to send a command to the instrument. Simply create a command string from the commands listed in this book, and place the string in your program language's output statement. For commands other than common commands, include a colon before the subsystem name. For example, the following string places the cursor on the peak laser line and returns the power level of this peak:

```
OUTPUT 720;":MEAS:SCAL:POW? MAX"
```

Commands can be sent using any combination of uppercase or lowercase ASCII characters. Instrument responses, however, are always returned in uppercase.

The program instructions within a data message are executed after the program message terminator is received. The terminator may be either a NL (new line) character, an EOI (End-Or-Identify) asserted in the GPIB interface, or a combination of the two. Asserting the EOI sets the EOI control line low on the last byte of the data message. The NL character is an ASCII linefeed (decimal 10). The NL (New Line) terminator has the same function as an EOS (End Of String) and EOT (End Of Text) terminator.

Short or Long Forms

Commands and queries may be sent in either long form (complete spelling) or short form (abbreviated spelling). The description of each command in this manual shows both versions; the extra characters for the long form are shown in lowercase. However, commands can be sent using any combination of uppercase or lowercase ASCII characters. Instrument responses, however, are always returned in uppercase. Programs written in long form are easily read and are almost self-documenting. Using short form commands conserves the amount of controller memory needed for program storage and reduces the amount of I/O activity.

The short form is the first four characters of the keyword, unless the fourth character is a vowel. Then the mnemonic is the first three characters of the keyword. If the length of the keyword is four characters or less, this rule does not apply, and the short form is the same as the long form.

For example:

```
:TIMEBASE:DELAY 1E-6 is the long form.
:TIM:DEL 1E-6 is the short form.
```

Table 3   Long and Short Command Forms

Long Form	Short Form	How the Rule is Applied
RANGE	RANG	Short form is the first four characters of the keyword.
PATTERN	PATT	Short form is the first four characters of the keyword.
DISK	DISK	Short form is the same as the long form.
DELAY	DEL	Fourth character is a vowel, short form is the first three characters.

White Space

White space is defined to be one or more characters from the ASCII set of 0 through 32 decimal, excluding 10 (NL). White space is usually optional, and can be used to increase the readability of a program.

Combining Commands

You can combine commands from the same subsystem provided that they are both on the same level in the subsystem's hierarchy. Simply separate the commands with a semi-colon (;). If you have selected a subsystem, and a common command is received by the instrument, the instrument remains in the selected subsystem. For example, the following commands turn averaging on, then clears the status information without leaving the selected subsystem.
```
":ACQUIRE:AVERAGE ON;*CLS;COUNT 1024"
```

You can send commands and program queries from different subsystems on the same line. Simply precede the new subsystem by a semicolon followed by a colon. Multiple commands may be any combination of compound and simple commands. For example:

```
:CHANNEL1:RANGE 0.4;:TIMEBASE:RANGE 1
```

Adding parameters to a command

Many commands have parameters that specify an option. Use a space character to separate the parameter from the command as shown in the following line:

```
OUTPUT 720;":INIT:CONT ON"
```
Separate multiple parameters with a comma (,). Spaces can be added around the commas to improve readability.

```
OUTPUT 720;":MEAS:SCAL:POW:FREQ? 1300, MAX"
```

String Arguments

Strings contain groups of alphanumeric characters which are treated as a unit of data by the instrument. You may delimit embedded strings with either single (') or double (") quotation marks. These strings are case-sensitive, and spaces act as legal characters just like any other character. For example, this command writes the line string argument to the instrument's advisory line:
```
:SYSTEM:DSP ""This is a message.""
```

Numbers

Some commands require number arguments. All numbers are expected to be strings of ASCII characters. You can use exponential notation or suffix multipliers to indicate the numeric value. The following numbers are all equal:

28 = 0.28E2 = 280E-1 = 28000m = 0.028K = 28E-3K

When a syntax definition specifies that a number is an integer, any fractional part is ignored and truncated. Using "mV" or "V" following the numeric voltage value in some commands will cause Error 138–Suffix not allowed. Instead, use the convention for the suffix multiplier.

Table 4   <suffix mult>

Value	Mnemonic	Value	Mnemonic
1E18	EX	1E-3	m
1E15	PE	1E-6	u
1E12	T	1E-9	n
1E9	G	1E-12	p

Table 4   <suffix mult>

Value	Mnemonic	Value	Mnemonic
1E6	MA	1E-15	f
1E3	K	1E-18	a

Table 5   <suffix unit>

Suffix	Referenced Unit
V	Volt
s	Second
W	Watt
BIT	Bits
dB	Decibel
%	Percent
Hz	Hertz

Infinity Representation

The representation for infinity for this instrument is 9.99999E+37. This is also the value returned when a measurement cannot be made.

Sequential and Overlapped Commands

IEEE 488.2 makes a distinction between sequential and overlapped commands. Sequential commands finish their task before the execution of the next command starts. Overlapped commands run concurrently. Commands following an overlapped command may be started before the overlapped command is completed. The common commands *WAI and *OPC may be used to ensure that commands are completely processed before subsequent commands are executed.

Queries

Command headers immediately followed by a question mark (?) are queries. After receiving a query, the instrument interrogates the requested subsystem and places the answer in its output queue. The answer remains in the output queue until it is read or until another command is issued. When read, the answer is transmitted across the bus to the designated listener (typically a computer). For example, the query:

```
:TIMEBASE:RANGE?
```

places the current time base setting in the output queue. In BASIC, the computer input statement:

```
ENTER < device address >;Range
```

passes the value across the bus to the computer and places it in the variable Range. You can use query commands to find out how the instrument is currently configured. They are also used to get results of measurements made by the instrument. For example, the command:

```
:MEASURE:RISETIME?
```

tells the instrument to measure the rise time of your waveform and place the result in the output queue. The output queue must be read before the next program message is sent. For example, when you send the query :MEASURE:RISETIME? you must follow it with an input statement. In BASIC, this is usually done with an ENTER statement immediately followed by a variable name. This statement reads the result of the query and places the result in a specified variable. If you send another command or query before reading the result of a query, the output buffer is cleared and the current response is lost. This also generates a query-interrupted error in the error queue. If you execute an input statement before you send a query, it will cause the computer to wait indefinitely.

If a measurement cannot be made because of the lack of data, because the source signal is not displayed, the requested measurement is not possible (for example, a period measurement on an FFT waveform), or for some other reason, 9.99999E+37 is returned as the measurement result. In TDR mode with ohms specified, the returned value is 838 MW.

You can send multiple queries to the instrument within a single program message, but you must also read them back within a single program message. This can be accomplished by either reading them back into a string variable or into multiple numeric variables. For example, you could read the result of the query :TIMEBASE:RANGE?;DELAY? into the string variable Results$ with the command: ENTER 707;Results$

When you read the result of multiple queries into string variables, each response is separated by a semicolon. For example, the response of the query :TIMEBASE:RANGE?;DELAY? would be:

```
<range_value>;<delay_value>
```

Use the following program message to read the query :TIMEBASE:RANGE?;DELAY? into multiple numeric variables:

```
ENTER 707;Result1,Result2
```

Definite-Length Block Response Data

Definite-length block response data allows any type of device-dependent data to be transmitted over the system interface as a series of 8-bit binary data bytes. This is particularly useful for sending large quantities of data or 8-bit extended ASCII codes. The syntax is a pound sign (#) followed by a non-zero digit representing the number of digits in the decimal integer. After the non-zero digit is the decimal integer that states the number of 8-bit data bytes being sent. This is followed by the actual data. For example, for transmitting 4000 bytes of data, the syntax would be:

```
#44000 <4000 bytes of data> <terminator>
```

The leftmost "4" represents the number of digits in the number of bytes, and "4000" represents the number of bytes to be transmitted.

Byte order can affect the ability of your programs to correctly interpret block data.

The byte order, or endianness, of returned block data differs between the Waveform and Measure subsystems. By default, the Waveform subsystem queries return block data in MSB (Most Significant Byte) first format. If needed, you can change the order to LSB (Least Significant Byte) first using the command "BYTeorder" on page 9.

The following Measure sybsystem queries return block data in LSB first format:

```
:MEASure:AMPLitutde:ISIVsbit?
:MEASure:AMPLitutde:ISIVsbit:BITS?
:MEASure:JITTer:DDJVsbit?
:MEASure:JITTer:DDJVsbit:BITS?
:MEASure:JITTer:EBITs?
:MEASure:JITTer:PATTern?
:MEASure:SINTegrity:PATTern?
```

Be aware that the Keysight IO Libraries Suite, by default, interprets received block data as MSB first format and there is no Measure subsystem command to change the byte order to LSB. When using these Measure subsystem queries, you *must* change the byte order received from MSB to LSB. For example, you could do one of the following:

- Open Keysight VEE's Advanced Instrument Properties dialog box, select the General tab, and change the byte order setting. However, using this method results in incorrect Waveform queries.

- Write a function to change the byte order in your program.

- Use a function already available in your authoring tool such as provided in Microsoft Excel.

Starting a Program

The commands and syntax for initializing the instrument are listed in Chapter 3, "Common Commands". Refer to your GPIB manual and programming language reference manual for information on initializing the interface. To make sure the bus and all appropriate interfaces are in a known state, begin every program with an initialization statement. For example, BASIC provides a CLEAR command which clears the interface buffer. When you are using GPIB, CLEAR also resets the instrument's parser. After clearing the interface, initialize the instrument to a preset state using the *RST command.

The AUTOSCALE command is very useful on unknown waveforms. It automatically sets up the vertical channel, time base, and trigger level of the instrument.

A typical instrument setup configures the vertical range and offset voltage, the horizontal range, delay time, delay reference, trigger mode, trigger level, and slope. An example of the commands sent to the instrument are:

```
:CHANNEL1:RANGE 16;OFFSET 1.00<terminator>
:SYSTEM:HEADER OFF<terminator>
:TIMEBASE:RANGE 1E-3;DELAY 100E-6<terminator>
```

This example sets the time base at 1 ms full-scale (100 ms/div), with delay of 100 ms. Vertical is set to 16V full-scale (2 V/div), with center of screen at 1V, and probe attenuation of 10.

The following program demonstrates the basic command structure used to program the instrument.

```
10 CLEAR 707 ! Initialize instrument interface
20 OUTPUT 707;"*RST" !Initialize instrument to preset state
30 OUTPUT 707;":TIMEBASE:RANGE 5E-4"! Time base to 500 us full scale
40 OUTPUT 707;":TIMEBASE:DELAY 25E-9"! Delay to 25 ns
50 OUTPUT 707;":TIMEBASE:REFERENCE CENTER"! Display reference at center
60 OUTPUT 707;":CHANNEL1:RANGE .16"! Vertical range to 160 mV full scale
70 OUTPUT 707;":CHANNEL1:OFFSET -.04"! Offset to -40 mV
80 OUTPUT 707;":TRIGGER:LEVEL,-.4"! Trigger level to -0.4
90 OUTPUT 707;":TRIGGER:SLOPE POSITIVE"! Trigger on positive slope
100OUTPUT 707;":SYSTEM:HEADER OFF"<terminator>
110OUTPUT 707;":DISPLAY:GRATICULE FRAME"! Grid off
120END
```

- Line 10 initializes the instrument interface to a known state and Line 20 initializes the instrument to a preset state.

- Lines 30 through 50 set the time base, the horizontal time at 500 ms full scale, and 25 ns of delay referenced at the center of the graticule.

- Lines 60 through 70 set the vertical range to 160 mV full scale and the center screen at -40 mV.

- Lines 80 through 90 configure the instrument to trigger at -0.4 volts with normal triggering.

- Line 100 turns system headers off.

- Line 110 turns the grid off.

The DIGITIZE command is a macro that captures data using the acquisition (ACQUIRE) subsystem. When the digitize process is complete, the acquisition is stopped. The captured data can then be measured by the instrument or transferred to the computer for further analysis. The captured data consists of two parts: the preamble and the waveform data record. After changing the instrument configuration, the waveform buffers are cleared. Before doing a measurement, the DIGITIZE command should be sent to ensure new data has been collected. You can send the DIGITIZE command with no parameters for a higher throughput. Refer to the DIGITIZE command in Chapter 4, "Root Level Commands" for details. When the DIGITIZE command is sent to an instrument, the specified channel's waveform is digitized with the current ACQUIRE parameters. Before sending the

:WAVEFORM:DATA? query to get waveform data, specify the WAVEFORM parameters. The number of data points comprising a waveform varies according to the number requested in the ACQUIRE subsystem. The ACQUIRE subsystem determines the number of data points, type of acquisition, and number of averages used by the DIGITIZE command. This allows you to specify exactly what the digitized information contains. The following program example shows a typical setup:

```
OUTPUT 707;":SYSTEM:HEADER OFF"<terminator>
OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"<terminator>
OUTPUT 707;":WAVEFORM:FORMAT BYTE"<terminator>
OUTPUT 707;":ACQUIRE:COUNT 8"<terminator>
OUTPUT 707;":ACQUIRE:POINTS 500"<terminator>
OUTPUT 707;":DIGITIZE CHANNEL1"<terminator>
OUTPUT 707;":WAVEFORM:DATA?"<terminator>
```

This setup places the instrument to acquire eight averages. This means that when the DIGITIZE command is received, the command will execute until the waveform has been averaged at least eight times. After receiving the :WAVEFORM:DATA? query, the instrument will start passing the waveform information when queried. Digitized waveforms are passed from the instrument to the computer by sending a numerical representation of each digitized point. The format of the numerical representation is controlled with the :WAVEFORM:FORMAT command and may be selected as BYTE, WORD, or ASCII. The easiest method of entering a digitized waveform depends on data structures, available formatting, and I/O capabilities. You must scale the integers to determine the voltage value of each point. These integers are passed starting with the leftmost point on the instrument's display. For more information, refer to Chapter 26, "Waveform Commands". When using GPIB, a digitize operation may be aborted by sending a Device Clear over the bus (for example, CLEAR 707).

NOTE   The execution of the DIGITIZE command is subordinate to the status of ongoing limit tests. (See commands ACQuire:RUNTil on page 126, MTEST:RUNTil on page 225, and LTEST:RUNTil on page 205.) The DIGITIZE command will not capture data if the stop condition for a limit test has been met.

Multiple Databases

Eye/Mask measurements are based on statistical data that is acquired and stored in the color grade/gray scale database. The color grade/gray scale database consists of all data samples displayed on the display graticule. The measurement algorithms are dependent upon histograms derived from the database. This database is internal to the instrument's applications. The color grade/gray scale database *cannot* be imported into an external database application.

If you want to perform an eye measurement, it is necessary that you first produce an eye diagram by triggering the instrument with a synchronous clock signal. Measurements made on a pulse waveform while in Eye/Mask mode will fail.

Firmware revision A.03.00 and later allows for multiple color grade/gray scale databases to be acquired and displayed simultaneously, including

·   all four instrument channels

·   all four math functions

·   one saved color grade/gray scale file

The ability to use multiple databases allows for the comparison of

·   channels to each other

·   channels to a saved color grade/gray scale file

·   functions to the channel data on which it is based

The advantage of acquiring and displaying channels and functions simultaneously is test times are greatly reduced. For example, the time taken to acquire two channels in parallel is approximately the same time taken to acquire a single channel.

Using Multiple Databases in Remote Programs

Most commands that control histograms, mask tests, or color grade data have additional optional parameters that were not available in firmware revisions prior to A.03.00. You can use the commands to control a single channel or add the argument APPend to enable more than one channel. The following example illustrates two uses of the CHANnel<n>:DISPlay command.

```
SYSTem:MODE EYE
CHANnel1:DISPlay ON
CHANnel2:DISPlay ON
```

The result using the above set of commands, is Channel 1 cleared and disabled while Channel 2 is enabled and displayed. However, by adding the argument APPend to the last command of the set, both Channels 1 and 2 will be enabled and displayed .

```
SYSTem:MODE EYE
CHANnel1:DISPlay ON
CHANnel2:DISPlay ON,APPend
```

For a example of using multiple databases, .

Downloading a Database

The general process for downloading a color grade/gray scale database is as follows:

1   Send the command :WAVEFORM:SOURCE CGRADE. This will select the color grade/gray scale database as the waveform source.

2   Issue :WAVeform:FORMat WORD. Database downloads only support word formatted data (16-bit integers).

3   Send the query :WAVeform:DATA? The data will be sent by means of a block data transfer as a two-dimensional array, 451 words wide by 321 words high (

Response Data" on page 21). The data is transferred starting with the upper left pixel of the display graticule, column by column, until the lower right pixel is transferred.

4   Send the command :WAVeform:XORigin to obtain the time of the left column.

5   Send the command :WAVeform:XINC to obtain the time increment of each column.

6   Send the command :WAVeform:YORigin to obtain the voltage or power of the vertical center of the database.

7   Send the command :WAVeform:YORigin to obtain the voltage or power of the incremental row.

The information from steps 4 through 7 can also be obtained with the command :WAVeform:PREamble.

Auto Skew

Another multiple database feature is the auto skew. You can use the auto skew feature to set the horizontal skew of multiple, active channels with the same bit rate, so that the waveform crossings align with each other. This can be very convenient when viewing multiple eye diagrams simultaneously. Slight differences between channels and test devices may cause a phase difference between channels. Auto skew ensures that each eye is properly aligned, so that measurements and mask tests can be properly executed.

In addition, auto skew optimizes the instrument trigger level. Prior to auto skew, at least one channel must display a complete eye diagram in order to make the initial bit rate measurement. Auto skew requires more data to be sampled; therefore, acquisition time during auto skew is slightly longer than acquisition time during measurements.

Files

When specifying a file name in a remote command, enclose the name in double quotation marks, such as "filename". If you specify a path, the path should be included in the quotation marks. All files stored using remote commands have file name extensions as listed in Table 6. You can use the full path name, a relative path name, or no path.

If you do not specify an extension when storing a file, or specify an incorrect extension, it will be corrected automatically according to the following rules:

· No extension specified: add the extension for the file type.

· Extension does not match file type: retain the filename, (including the current extension) and add the appropriate extension.

You do not need to use an extension when loading a file if you use the optional destination parameter. For example, :DISK:LOAD "STM1_OC3",SMASK automatically adds .msk to the file name. ASCII waveform files can be loaded only if the file name explicitly includes the .txt extension. Table 7 on page 24 shows the rules used when loading a specified file.

If you don't specify a directory when storing a file, the location of the file will be based on the file type. Table 8 on page 24 shows the default locations for storing files. On 86100C/D instruments, files are stored on the D: drive. On 86100A/B instruments, files are stored on the C: drive.

When loading a file, you can specify the full path name, a relative path name, or no path name. Table 9 on page 25 lists the rules for locating files, based on the path specified. Standard masks loaded from D:\Scope\masks. Files may be stored to or loaded from any path external drive or on any mapped network drive.

Table 6   File Name Extensions

File Type	File Name Extension	Command
Waveform - internal format	.wfm	"STORe" on page 19
Waveform - text format (Verbose, XY Verbose, or Y values)	.txt	"STORe" on page 19
Pattern Waveform	.csv	"PWAVeform:SAVE" on page 14
Setup	.set	"STORe" on page 19
Color grade - Gray Scale	.cgs	"STORe" on page 19
Jitter Memory	.jd	"STORe" on page 19
Screen image [*]	.bmp, .eps, .gif, .pcx, .ps, .jpg, .tif	"SIMage" on page 15
Mask	.msk, .pcm	"SAVE" on page 17
TDR/TDT	.tdr	"STORe" on page 19
MATLAB script	.m	"MATLab:SCRipt" on page 13
S-Parameter (Touchstone format)	.s1p, .s2p, .s4p	"SPARameter:SAVE" on page 17
S-Parameter (text format)	.txt	"SPARameter:SAVE" on page 17

*For .gif and .tif file formats, this instrument uses LZW compression/decompression licensed under U.S. patent No 4,558,302 and foreign counterparts. End user should not modify, copy, or distribute LZW compression/decompression capability. For .jpg file format, this instrument uses the .jpg software written by the Independent JPEG Group.

Table 7   Rules for Loading Files

File Name Extension	Destination	Rule
No extension	Not specified	Default to internal waveform format; add .wfm extension
Extension does not match file type	Not specified	Default to internal waveform format; add .wfm extension
Extension matches file type	Not specified	Use file name with no alterations; destination is based on extension file type
No extension	Specified	Add extension for destination type; default for waveforms is internal format (.wfm)
Extension does not match destination file type	Specified	Retain file name; add extension for destination type. Default for waveforms is internal format (.wfm)
Extension matches destination file type	Specified	Retain file name; destination is as specified

Table 8   Default File Locations

File Type	Default Location
Waveform - internal format, text format (Verbose, XY Verbose, or Y values),	D:\User Files\waveforms
Pattern Waveforms	D:\User Files\waveforms

Table 8   Default File Locations (continued)

File Type	Default Location
Setup	D:\User Files\setups
Color Grade - Gray Scale	D:\User Files\colorgrade-grayscale
Jitter Memory	D:\User Files\jitter data
Screen Image	D:\User Files\screen images
Mask	C:\Scope\masks (standard masks) D:\User Files\masks (user-defined masks)
TDR/TDT calibration data (software revision A.05.00 and below)	D:\User Files\TDR normalization
TDR/TDT calibration data (software revision A.06.00 and above)	D:\User Files\TDR calibration
MATLAB script	D:\User Files\MATLAB scripts
S-Parameters	D:\User Files\S-parameter data

Table 9   File Locations (Loading Files)

File Name	Rule
Full path name	Use file name and path specified
Relative path name	Full path name is formed relative to the present working directory, set with the command :DISK:CDIR. The present working directory can be read with the query :DISK:PWD?
File name with no preceding path	Add the file name to the default path (D:\User Files) based on the file type. (C drive on 86100A/B instruments.)

Status Reporting

Almost every program that you write will need to monitor the instrument for its operating status. This includes querying execution or command errors and determining whether or not measurements have been completed. Several status registers and queues are provided to accomplish these tasks. In this section, you'll learn how to enable and read these registers.

- Refer to Figure 5 on page 27 for an overall status reporting decision chart.
- See Figure 6 and Figure 7 to learn the instrument's status reporting structure which allows you to monitor specific events in the instrument.
- Table 10 on page 32 lists the bit definitions for each bit in the status reporting data structure.

The Status Byte Register, the Standard Event Status Register group, and the Output Queue are defined as the Standard Status Data Structure Model in IEEE 488.2-1987. IEEE 488.2 defines data structures, commands, and common bit definitions for status reporting. There are also instrument-defined structures and bits.

To monitor an event, first clear the event, then enable the event. All of the events are cleared when you initialize the instrument. To generate a service request (SRQ) interrupt to an external computer, enable at least one bit in the Status Byte Register. To make it possible for any of the Standard Event Status Register bits to generate a summary bit, the corresponding bits must be enabled. These bits are enabled by using the *ESE common command to set the corresponding bit in the Standard Event Status Enable Register. To generate a service request (SRQ) interrupt to the computer, at least one bit in the Status Byte Register must be enabled. These bits are enabled by using the *SRE common command to set the corresponding bit in the Service Request Enable Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register. For more information about common commands, see Chapter 3, "Common Commands".

Status Byte Register

The Status Byte Register is the summary-level register in the status reporting structure. It contains summary bits that monitor activity in the other status registers and queues. The Status Byte Register is a live register. That is, its summary bits are set and cleared by the presence and absence of a summary bit from other event registers or queues. If the Status Byte Register is to be used with the Service Request Enable Register to set bit 6 (RQS/MSS) and to generate an SRQ, at least one of the summary bits must be enabled, then set. Also, event bits in all other status registers must be specifically enabled to generate the summary bit that sets the associated summary bit in the Status Byte Register.

The Status Byte Register can be read using either the *STB? common command query or the GPIB serial poll command. Both commands return the decimal-weighted sum of all set bits in the register. The difference between the two methods is that the serial poll command reads bit 6 as the Request Service (RQS) bit and clears the bit which clears the SRQ interrupt. The *STB? query reads bit 6 as the Master Summary Status (MSS) and does not clear the bit or have any affect on the SRQ interrupt. The value returned is the total bit weights of all of the bits that are set at the present time.

Do you want to do status reporting?

no

yes

Reset the instrument and clear the status registers.
OUTPUT 707;"*RST"
OUTPUT 707;"*CLS"

Do you want to send a Service Request (SRQ) interrupt to the controller?

no

yes

Do you want to report events monitored by the Standard Event Status Register?

no

yes

Use the *ESE common command to enable the bits you want to use to generate a summary bit to the Status Byte Register.

Use the *ESE common command to enable the bits you want to generate the RQS/MSS bit to set bit 6 in the Status Byte Register and send an SRQ to the computer. If events are monitored by the Standard event Status Register also Enable ESB with *SRE command.

Activate the instrument function that you want to monitor.

When an interrupt occurs, read the Status Byte Register. Use the following: P=SPOLL(707)
        PRINT P

To read the Status Byte Register use the following:
OUTPUT 707;"*STB?"
ENTER 707;<variable>
PRINT <variable>
This reads the decimal value of the Status Byte Register.

Determine which bits in the Status Byte Register are set.

Use the following to read the Standard Event Status Register:
OUTPUT 707;"*ESR?
ENTER 707;<variable>
PRINT <variable>

Use the following to see if an operation is complete:
OUTPUT 707;"*OPC?
ENTER 707;<variable>
PRINT <variable>

Use the following to read the contents of the status byte:
OUTPUT 707;"*STB?
ENTER 707;<variable>
PRINT <variable>

END

54700b05

Figure 5   Status Reporting Decision Chart

The use of bit 6 can be confusing. This bit was defined to cover all possible computer interfaces, including a computer that could not do a serial poll. The important point to remember is that, if you are using an SRQ interrupt to an external computer, the serial poll command clears bit 6. Clearing bit 6 allows the instrument to generate another SRQ interrupt when another enabled event occurs. The only other bit in the Status Byte Register affected by the *STB? query is the Message Available bit (bit 4). If there are no other messages in the Output Queue, bit 4 (MAV) can be cleared as a result of reading the response to the *STB? query.

If bit 4 (weight = 16) and bit 5 (weight = 32) are set, a program would print the sum of the two weights. Since these bits were not enabled to generate an SRQ, bit 6 (weight = 64) is not set.
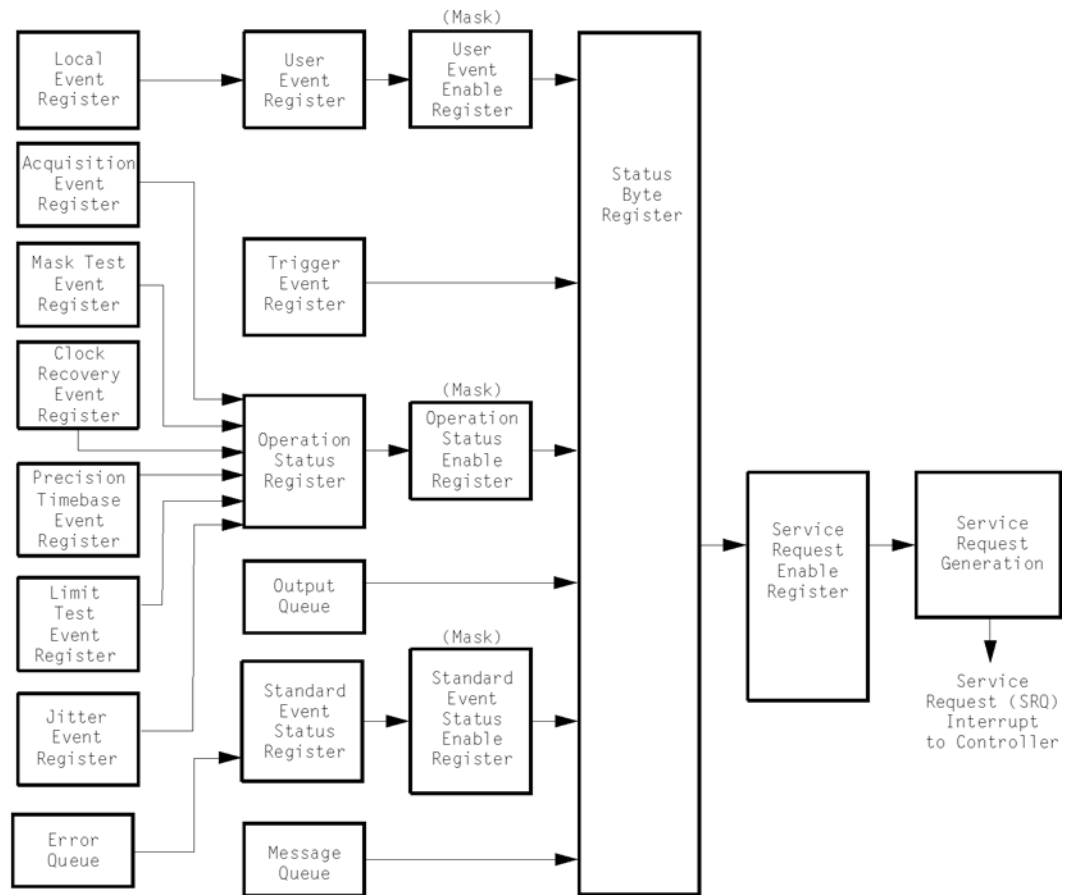
Figure 6   Status Reporting Overview
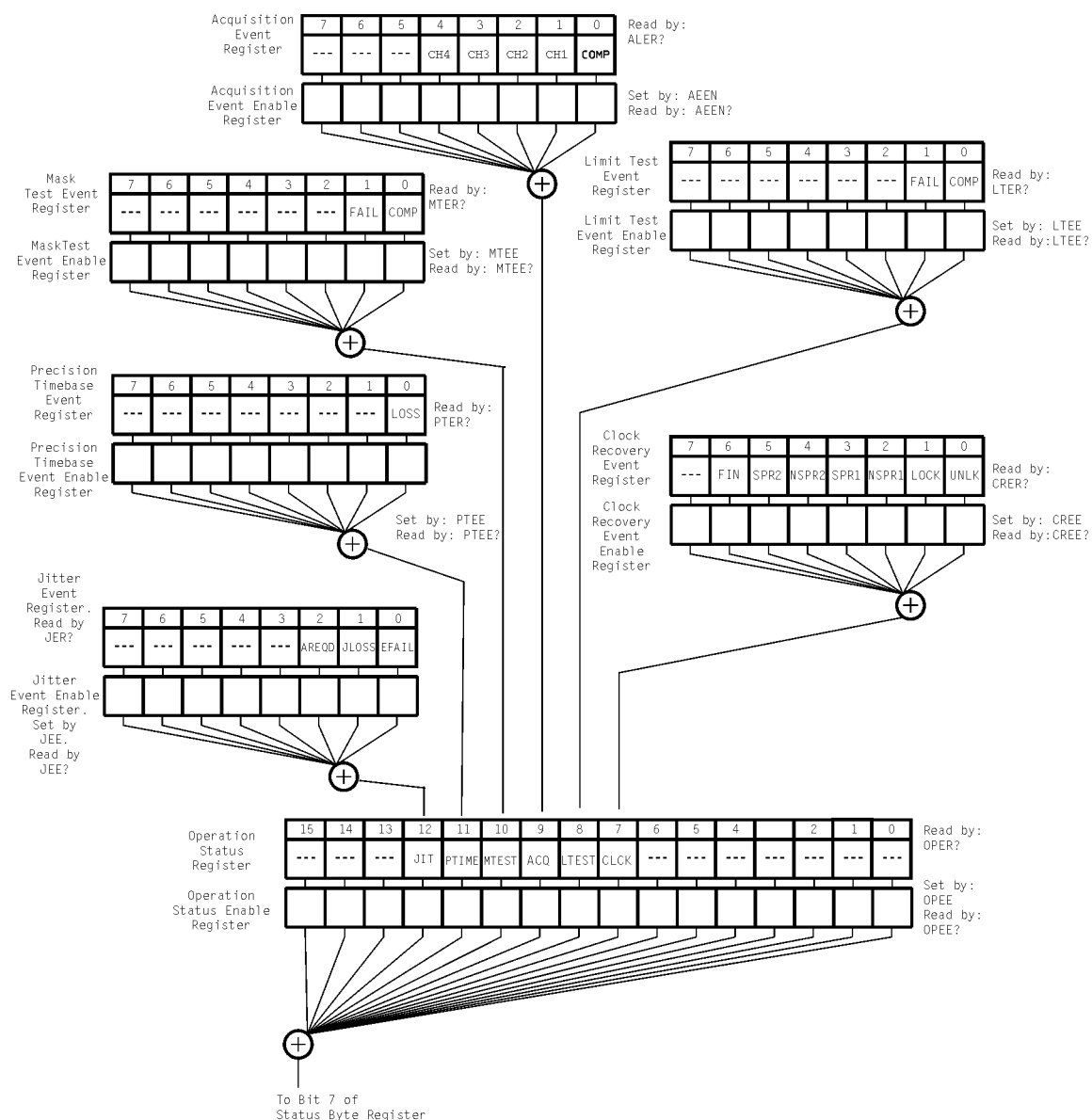
Figure 7   Status Reporting Data Structures
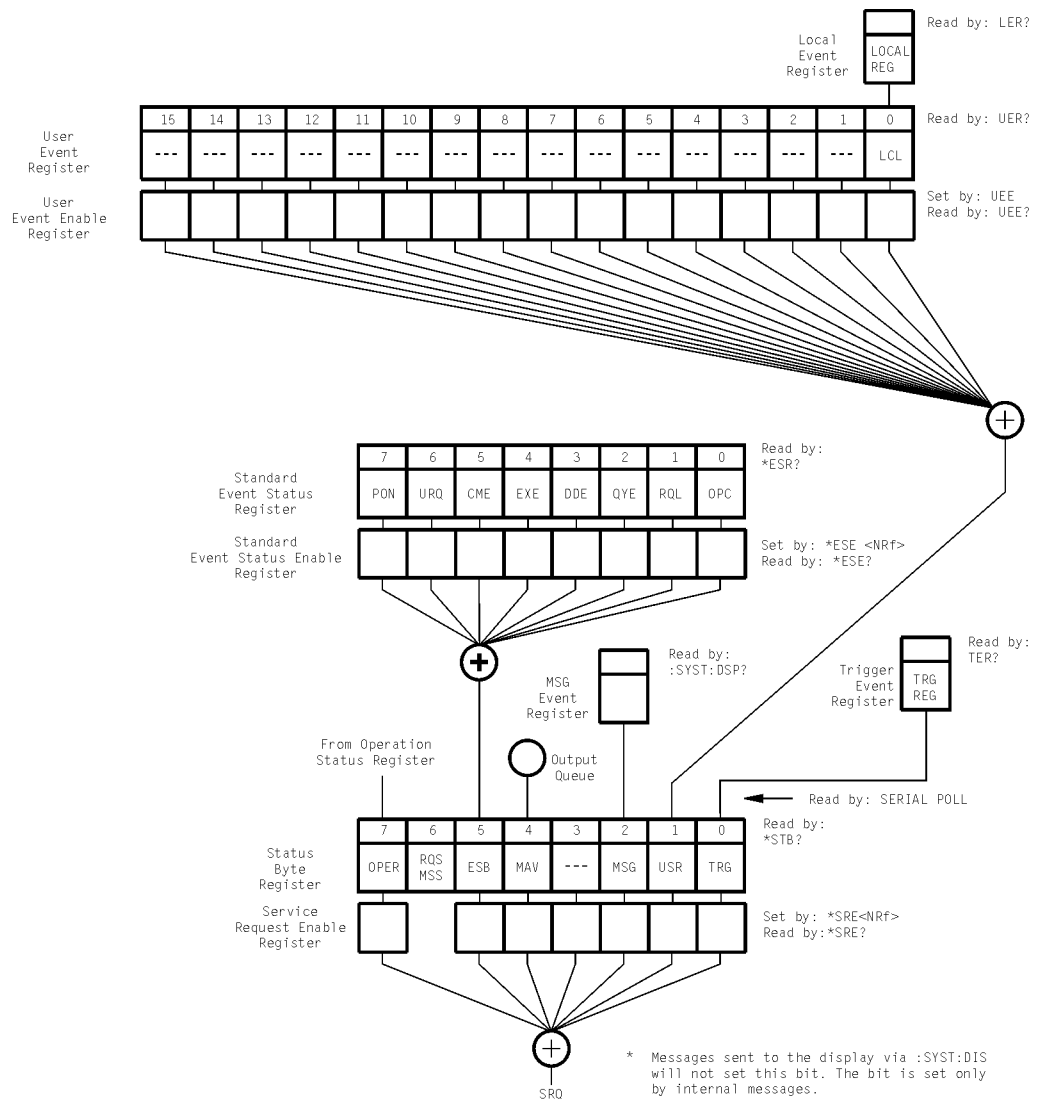
statdata2

Status Reporting Data Structures (continued)

This BASIC example uses the *STB? query to read the contents of the instrument's Status Byte Register when none of the register's summary bits are enabled to generate an SRQ interrupt.

```
10 OUTPUT 707;":SYSTEM:HEADER OFF;*STB?"!Turn headers off
20 ENTER 707;Result!Place result in a numeric variable
30 PRINT Result!Print the result
40 End
```

The next program prints 132 and clears bit 6 (RQS) of the Status Byte Register. The difference in the decimal value between this example and the previous one is the value of bit 6 (weight = 64). Bit 6 is set when the first enabled summary bit is set, and is cleared when the Status Byte Register is read by the serial poll command.

This example uses the BASIC serial poll (SPOLL) command to read the contents of the instrument's Status Byte Register.

```
10 Result = SPOLL(707)
20 PRINT Result
30 END
```

Use Serial Polling to Read the Status Byte Register. Serial polling is the preferred method to read the contents of the Status Byte Register because it resets bit 6 and allows the next enabled event that occurs to generate a new SRQ interrupt.

Service Request Enable Register

Setting the Service Request Enable Register bits enables corresponding bits in the Status Byte Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register. Bits are set in the Service Request Enable Register using the *SRE command, and the bits that are set are read with the *SRE? query. Bit 6 always returns 0. Refer to the Status Reporting Data Structures shown in Figure 7 on page 29. This example sets bit 4 (MAV) and bit 5 (ESB) in the Service Request Enable Register.
OUTPUT 707;"*SRE 48"

This example uses the parameter "48" to allow the instrument to generate an SRQ interrupt under the following conditions:

·   When one or more bytes in the Output Queue set bit 4 (MAV).

·   When an enabled event in the Standard Event Status Register generates a summary bit that sets bit 5 (ESB).

Trigger Event Register (TRG)

This register sets the TRG bit in the status byte when a trigger event occurs. The TRG event register stays set until it is cleared by reading the register or using the *CLS (clear status) command. If your application needs to detect multiple triggers, the TRG event register must be cleared after each one. If you are using the Service Request to interrupt a computer operation when the trigger bit is set, you must clear the event register after each time it is set.

Table 10   Status Reporting Bit Definition (Sheet 1 of 2)

Bit	Description	Definition
ACQ	Acquisition	Indicates that acquisition test has completed in the Acquisition Register.
AREQD	Autoscale Required	Indicates that a parameter change in Jitter Mode has made an autoscale necessary.
CLCK	CloCk	Indicates that one of the enabled conditions in the Clock Recovery Register has occurred.
CME	Command Error	Indicates if the parser detected an error.
COMP	Complete	Indicates the specified test has completed.
DDE	Device Dependent Error	Indicates if the device was unable to complete an operation for device dependent reasons.
EFAIL	Edge Characterization Fail	Indicates that the characterizing of edges in Jitter Mode has failed.
ESB	Event Status Bit	Indicates if any of the enabled conditions in the Standard Event Status Register have occurred.
EXE	Execution Error	Indicates if a parameter was out of range or was inconsistent with the current settings.
FAIL	Fail	Indicates the specified test has failed.
FIN	Finished	Indicates that a clock recovery relock operation has completed.
JLOSS	Pattern Synchronization Loss	Indicates that the pattern synchronization is lost in Jitter Mode.
LCL	Local	Indicates if a remote-to-local transition occurs.
LOCK	LOCKed	Indicates that a locked or trigger capture condition has occurred in the Clock Recovery Module.
LOSS	Time Reference Loss	Indicates the Precision Timebase (provided by the Keysight 86107A module) has detected a time reference loss due to a change in the reference clock signal.
LTEST	Limit Test	Indicates that one of the enabled conditions in the Limit Test Register has occurred.
MAV	Message Available	Indicates if there is a response in the output queue.
MSG	Message	Indicates if an advisory has been displayed.
MSS	Master Summary Status	Indicates if a device has a reason for requesting service.
MTEST	Mask Test	Indicates that one of the enabled conditions in the Mask Test Register has occurred.
NSPR1	No Signal Present Receiver 1	Indicates that the Clock Recovery Module has detected the loss of an optical signal on receiver one.
NSPR2	No Signal Present Receiver 2	Indicates that the Clock Recovery Module has detected the loss of an optical signal on receiver two.
OPC	Operation Complete	Indicates if the device has completed all pending operations.
OPER	Operation Status Register	Indicates if any of the enabled conditions in the Operation Status Register have occurred.
PON	Power On	Indicates power is turned on.
PTIME	Precision Timebase	Indicates that one of the enabled conditions in the Precision Timebase Register has occurred.
QYE	Query Error	Indicates if the protocol for queries has been violated.
RQL	Request Control	Indicates if the device is requesting control.

Table 10   Status Reporting Bit Definition (Sheet 2 of 2)

Bit	Description	Definition
RQS	Request Service	Indicates that the device is requesting service.
SPR1	Signal Present Receiver 1	Indicates that the Clock Recovery Module has detected an optical signal on receiver one.
SPR2	Signal Present Receiver 2	Indicates that the Clock Recovery Module has detected an optical signal on receiver two.
TRG	Trigger	Indicates if a trigger has been received.
UNLK	UNLoCKed	Indicates that an unlocked or trigger loss condition has occurred in the Clock Recovery Module.
URQ		Not used. Permanently set to zero.
USR	User Event Register	Indicates if any of the enabled conditions have occurred in the User Event Register.

Standard Event Status Register

The Standard Event Status Register (SESR) monitors the following instrument status events:

- PON – Power On
- CME – Command Error
- EXE – Execution Error
- DDE – Device Dependent Error
- QYE – Query Error
- RQC – Request Control
- OPC – Operation Complete

When one of these events occurs, the corresponding bit is set in the register. If the corresponding bit is also enabled in the Standard Event Status Enable Register, a summary bit (ESB) in the Status Byte Register is set. The contents of the Standard Event Status Register can be read and the register cleared by sending the *ESR? query. The value returned is the total bit weights of all of the bits set at the present time. If bit 4 (weight = 16) and bit 5 (weight = 32) are set, the program prints the sum of the two weights. This example uses the *ESR? query to read the contents of the Standard Event Status Register.

```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Turn headers off
20 OUTPUT 707;"*ESR?"
30 ENTER 707;Result!Place result in a numeric variable
40 PRINT Result!Print the result
50 End
```

Standard Event Status Enable Register

For any of the Standard Event Status Register (SESR) bits to generate a summary bit, you must first enable the bit. Use the *ESE (Event Status Enable) common command to set the corresponding bit in the Standard Event Status Enable Register. Set bits are read with the *ESE? query. Suppose your application requires an interrupt whenever any type of error occurs. The error status bits in the Standard Event Status Register are bits 2 through 5. The sum of the decimal weights of these bits is 60. Therefore, you can enable any of these bits to generate the summary bit by sending:
```
OUTPUT 707;"*ESE 60"
```

Whenever an error occurs, the instrument sets one of these bits in the Standard Event Status Register. Because the bits are all enabled, a summary bit is generated to set bit 5 (ESB) in the Status Byte Register. If bit 5 (ESB) in the Status Byte Register is enabled (via the *SRE command), a service request interrupt (SRQ) is sent to the external computer.

NOTE	Disabled SESR Bits Respond, but Do Not Generate a Summary Bit. Standard Event Status Register bits that are not enabled still respond to their corresponding conditions (that is, they are set if the corresponding event occurs). However, because they are not enabled, they do not generate a summary bit in the Status Byte Register.

User Event Register (UER)

This register hosts the LCL bit (bit 0) from the Local Events Register. The other 15 bits are reserved. You can read and clear this register using the UER? query. This register is enabled with the UEE command. For example, if you want to enable the LCL bit, you send a mask value of 1 with the UEE command; otherwise, send a mask value of 0.

Local Event Register (LCL)

This register sets the LCL bit in the User Event Register and the USR bit (bit 1) in the Status byte. It indicates a remote-to-local transition has occurred. The LER? query is used to read and to clear this register.

Operation Status Register (OPR)

This register hosts the CLCK bit (bit 7), the LTEST bit (bit 8), the ACQ bit (bit 9) and the MTEST bit (bit 10). The CLCK bit is set when any of the enabled conditions in the Clock Recovery Event Register have occurred. The LTEST bit is set when a limit test fails or is completed and sets the corresponding FAIL or COMP bit in the Limit Test Events Register. The ACQ bit is set when the COMP bit is set in the Acquisition Event Register, indicating that the data acquisition has satisfied the specified completion criteria. The MTEST bit is set when the Mask Test either fails specified conditions or satisfies its completion criteria, setting the corresponding FAIl or COMP bits in the Mask Test Events Register. The PTIME bit is set when there is a loss of the precision timebase reference occurs setting a bit in the Precision Timebase Events Register. The JIT bit is set in Jitter Mode when a bit is set in the Jitter Events Register. This occurs when there is a failure or an autoscale is needed. If any of these bits are set, the OPER bit (bit 7) of the Status Byte register is set. The Operation Status Register is read and cleared with the OPER? query. The register output is enabled or disabled using the mask value supplied with the OPEE command.

Acquisition Event Register (AER)

Bit 0 (COMP) of the Acquisition Event Register is set when the acquisition limits complete. The Acquisition completion criteria are set by the ACQuire:RUNtil command. Refer to "RUNTil" on page 13. The Acquisition Event Register is read and cleared with the ALER? query. Refer to "ALER?" on page 9.

Clock Recovery Event Register (CRER)

This register hosts the UNLK bit (bit 0), LOCK bit (bit 1), NSPR1 bit (bit 2), SPR1 bit (bit 3), NSPR2 bit (bit 4) and SPR2 bit (bit 5). Bit 0 (UNLK) of the Clock Recovery Event Register is set when an 83491/2/3/4/5/6A clock recovery module becomes unlocked or trigger loss has occurred. Bit 1 (LOCK) of the Clock Recovery Event Register is set when a clock recovery module becomes locked or a trigger capture has occurred. If an 83496A module is locked, sending the CRECovery:RELock command does not set UNLK bit (bit 0) or LOCK bit (bit 1). To determine if the RELock command has completed, use the CRECovery:LOCKed? query. Refer to "RELock" on page 23.

Bits 2 through 5 are valid only for modules that support the :SPResent command (refer to Table 28 on page 152 and "SPResent?" on page 24), which includes the 83491/2/3/4A and 86108A/B modules. Since these bits provide information on optical signals they are not effected by 83495/6A modules. Bit 2 (NSPR1) of the Clock Recovery Event Register is set when an clock recovery module transitions to no longer detecting an optical signal on receiver one. Bit 3 (SPR1) of the Clock Recovery Event Register is set when an clock recovery module transitions to detecting an optical signal on receiver one. Bit 4 (NSPR2) of the Clock Recovery Event Register is set when an clock recovery module transitions to no longer detecting an optical signal on receiver two. Bit 5 (SPR2) of

the Clock Recovery Event Register is set when an clock recovery module transitions to detecting an optical signal on receiver two. The Clock Recovery Event Register is read and cleared with the CRER? query. Refer to "CRER?" on page 13. When either of the UNLK, LOCK, NSPR1, SPR1, NSPR2 or SPR2 bits are set, they in turn set CLCK bit (bit 7) of the Operation Status Register. Results from the Clock Recovery Event Register can be masked by using the CREE command to set the Clock Recovery Event Enable Register. Refer to Refer to "CREE" on page 12 for enable and mask value definitions.

Limit Test Event Register (LTER)

Bit 0 (COMP) of the Limit Test Event Register is set when the Limit Test completes. The Limit Test completion criteria are set by the LTESt:RUN command. Refer to "RUNTil" on page 10. Bit 1 (FAIL) of the Limit Test Event Register is set when the Limit Test fails. Failure criteria for the Limit Test are defined by the LTESt:FAIL command. Refer to "FAIL" on page 7. The Limit Test Event Register is read and cleared with the LTER? query. Refer to "LTER?" on page 17. When either the COMP or FAIL bits are set, they in turn set the LTEST bit (bit 8) of the Operation Status Register. You can mask the COMP and FAIL bits, thus preventing them from setting the LTEST bit, by defining a mask using the LTEE command. Refer to "LTEE" on page 16. When the COMP bit is set, it in turn sets the ACQ bit (bit 9) of the Operation Status Register. Results from the Acquisition Register can be masked by using the AEEN command to set the Acquisition Event Enable Register to the value 0. You enable the COMP bit by setting the mask value to 1.

Jitter Event Register (JIT)

Bit 0 (EFAIL) of the Jitter Event Register is set when characterizing edges in Jitter Mode fails. Bit 1 (JLOSS) of the register is set when pattern synchronization is lost in Jitter Mode. Bit 2 (AREQD) of the register is set when a parameter change in Jitter Mode has made autoscale necessary. Bit 12 of the Operation Status Register (JIT) indicates that one of the enabled conditions in the Jitter Event Register has occurred. You can mask the EFAIL, JLOSS, and AREQD bits, thus preventing them from setting the JIT bit, by setting corresponding bits to zero using the JEE command. Refer to "JEE" on page 15.

Mask Test Event Register (MTER)

Bit 0 (COMP) of the Mask Test Event Register is set when the Mask Test completes. The Mask Test completion criteria are set by the MTESt:RUNTil command. Refer to "RUNTil" on page 16. Bit 1 (FAIL) of the Mask Test Event Register is set when the Mask Test fails. This will occur whenever any sample is recorded within any region defined in the mask. The Mask Test Event Register is read and cleared with the MTER? query. Refer to "MTER?" on page 19. When either the COMP or FAIL bits are set, they in turn set the MTEST bit (bit 10) of the Operation Status Register. You can mask the COMP and FAIL bits, thus preventing them from setting the MTEST bit, by setting corresponding bits to zero using the MTEE command. Refer to "MTEE" on page 18.

Precision Timebase Event Register (PTER)

The Precision Timebase feature requires the installation of the Keysight 86107A Precision Timebase Module. Bit 0 (LOSS) of the Precision Timebase Event Register is set when loss of the time reference occurs. Time reference is lost when a change in the amplitude or frequency of the reference clock signal is detected. The Precision Timebase Event Register is read and cleared with the PTER? query. Refer to "PTER?" on page 20. When the LOSS bit is set, it in turn sets the PTIME bit (bit 11) of the Operation Status Register. Results from the Precision Timebase Register can be masked by using the PTEE command to set the Precision Timebase Event Enable Register to the value 0. You enable the LOSS bit by setting the mask value to 1. Refer to "PTEE" on page 20.

Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. If the error queue overflows, the last error in the queue is replaced with error –350, "Queue overflow". Any time the queue overflows, the oldest errors remain in the queue, and the most recent error is discarded.

The length of the instrument's error queue is 30 (29 positions for the error messages, and 1 position for the "Queue overflow" message). The error queue is read with the SYSTEM:ERROR? query. Executing this query reads and removes the oldest error from the head of the queue, which opens a position at the tail of the queue for a new error. When all the errors have been read from the queue, subsequent error queries return 0, "No error." The error queue is cleared when any of the following occurs:

· When the instrument is powered up.

· When the instrument receives the *CLS common command.

· When the last item is read from the error queue.

For more information on reading the error queue, refer to the SYSTEM:ERROR? query in Chapter 5, "DATE   117". For a complete list of error messages, refer to "Error Messages" on page 46.

Output Queue

The output queue stores the instrument-to-computer responses that are generated by certain instrument commands and queries. The output queue generates the Message Available summary bit when the output queue contains one or more bytes. This summary bit sets the MAV bit (bit 4) in the Status Byte Register. The output queue may be read with the BASIC ENTER statement.

Message Queue

The message queue contains the text of the last message written to the advisory line on the screen of the instrument. The queue is read with the SYSTEM:DSP? query. Note that messages sent with the SYSTem:DSP command do not set the MSG status bit in the Status Byte Register.

Clearing Registers and Queues

The *CLS common command clears all event registers and all queues except the output queue. If *CLS is sent immediately following a program message terminator, the output queue is also cleared.

Interface Functions

The interface functions deal with general bus management issues, as well as messages that can be sent over the bus as bus commands. In general, these functions are defined by IEEE 488.1. The instrument is equipped with a GPIB interface connector on the rear panel. This allows direct connection to a GPIB equipped computer. You can connect an external GPIB compatible device to the instrument by installing a GPIB cable between the two units. Finger tighten the captive screws on both ends of the GPIB cable to avoid accidentally disconnecting the cable during operation. A maximum of fifteen GPIB compatible instruments (including a computer) can be interconnected in a system by stacking connectors. This allows the instruments to be connected in virtually any configuration, as long as there is a path from the computer to every device operating on the bus. The interface capabilities of this instrument, as defined by IEEE 488.1, are listed in the Table 11 on page 38.

CAUTION	Avoid stacking more than three or four cables on any one connector. Multiple connectors produce leverage that can damage a connector mounting.

GPIB Default Startup Conditions

The following default GPIB conditions are established during power-up: 1) The Request Service (RQS) bit in the status byte register is set to zero. 2) All of the event registers, the Standard Event Status Enable Register, Service Request Enable Register, and the Status Byte Register are cleared.

Command and Data Concepts

The GPIB has two modes of operation, command mode and data mode. The bus is in the command mode when the Attention (ATN) control line is true. The command mode is used to send talk and listen addresses and various bus commands such as group execute trigger (GET). The bus is in the data mode when the ATN line is false. The data mode is used to convey device-dependent messages across the bus. The device-dependent messages include all of the instrument specific commands, queries, and responses found in this manual, including instrument status information.

Communicating Over the Bus

Device addresses are sent by the computer in the command mode to specify who talks and who listens. Because GPIB can address multiple devices through the same interface card, the device address passed with the program message must include the correct interface select code and the correct instrument address.

Device Address = (Interface Select Code * 100) + (Instrument Address)

The examples in this manual assume that the instrument is at device address 707. Each interface card has a unique interface select code. This code is used by the computer to direct commands and communications to the proper interface. The default is typically "7" for GPIB interface cards. Each instrument on the GPIB must have a unique instrument address between decimal 0 and 30. This instrument address is used by the computer to direct commands and communications to the proper instrument on an interface. The default is typically "7" for this instrument. You can change the instrument address in the Utilities, Remote Interface dialog box.

NOTE	Do Not Use Address 21 for an Instrument Address. Address 21 is usually reserved for the Computer interface Talk/Listen address and should not be used as an instrument address.

Bus Commands

The following commands are IEEE 488.1 bus commands (ATN true). IEEE 488.2 defines many of the actions that are taken when these commands are received by the instrument. The device clear (DCL) and selected device clear (SDC) commands clear the input buffer and output queue, reset the parser, and clear any pending commands. If either of these commands is sent during a digitize operation, the digitize operation is aborted. The group execute trigger (GET) command arms the trigger. This is the same action produced by sending the RUN command. The interface clear (IFC) command halts all bus activity. This includes unaddressing all listeners and the talker, disabling serial poll on all devices, and returning control to the system computer.

Table 11   Interface Capabilities

Code	Interface Function	Capability
SH1	Source Handshake	Full Capability
AH1	Acceptor Handshake	Full Capability
T5	Talker	Basic Talker/Serial Poll/Talk Only Mode/. Unaddress if Listen Address (MLA)
L4	Listener	Basic Listener/Unaddresses if Talk Address (MTA)
SR1	Service Request	Full Capability
RL1	Remote Local	Complete Capability
PP1	Parallel Poll	Remote Configuration
DC1	Device Clear	Full Capability
DT1	Device Trigger	Full Capability
C0	Computer	No Capability
E2	Driver Electronics	Tri State (1 MB/SEC MAX)

Commands Unavailable in Jitter Mode

This section describes the commands that can generate errors when controlling the instrument in Jitter mode. This can be due to the command or one of its arguments that are not allowed in Jitter mode. Refer to the individual command reference for detailed information.

Measure Commands

- MATLab   270
- MATLab<N>:SCRipt   270
- MATLab<N>:ETENable   270
- MATLab<N>:ETEXt?   270

Waveform Files

Waveform and Color Grade/Gray Scale files cannot be saved or loaded in Jitter mode. The commands listed below produce a "Settings conflict" error when executed in Jitter Mode.

- STORe   173

    When used with sources other than SETup and JDMemory.

- STORe:WAVeform   114
- ACQuire:SWAVeform   128
- LTESt:SWAVeform   209
- MTESt:SWAVeform   231

Waveform Queries

Only jitter database waveforms may be set or queried in Jitter mode. Using the following command produces the error, "Signal or trigger source selection is not available".

- :WAVeform:DATA   345

Waveform Memory Load/Store

Waveforms cannot be saved into waveform memories in Jitter mode. All waveform memories are turned off when entering Jitter mode. The commands listed below produce a "Settings conflict" error when executed in Jitter mode.

- WMEMory<N>:LOAD   355
- WMEMory<N>:SAVE   356
- DISK:LOAD   167

    When used with sources other than SETup and JDMemory.

WAveform Memory Display

Waveform memories cannot be turned on in Jitter mode. The following command produces a "Settings conflict" error when executed in Jitter mode.

- WMEMory<N>:DISPlay   355

Waveform and Color Grade-Gray Scale Memory

The Waveform and Color Grade/Gray Scale memories cannot be turned on in Jitter mode. The following command produces an "Illegal parameter value" error when executed in Jitter mode.

- VIEW   115

When used with arguments other than JDMemory.

Timebase Scale And Delay

Scale and position controls on the Horizontal setup dialog are disabled in Jitter Mode. The following commands produce a "Settings conflict" error when executed in Jitter Mode:

- TIMebase:RANGe   332
- TIMebase:SCALe   333
- TIMebase:POSition   330

Channel Scale And Offset

Channel scale and offset controls are disabled in Jitter mode. The following commands produce a "Settings conflict" error when executed in Jitter Mode.

- CHANnel<N>:OFFSet   144
- CHANnel<N>:RANGe   146
- CHANnel<N>:SCALe   147

Acquisition Settings

Acquisition (Averaging) controls are disabled in Jitter mode. The following commands produce a "Settings conflict" error when executed in Jitter mode.

- ACQuire:AVERage   123
- ACQuire:BEST   124
- ACQuire:POINts   125

Histograms

Histograms are turned off when entering Jitter mode. The following commands produce a "Control is set to default" error.

- HISTogram:MODE   200
- VIEW   115

Software Skewing of Channels

All skew adjustments are disabled in jitter mode. The following commands produce a "Settings conflict" error when executed in Jitter mode.

- CALibrate:SKEW   139
- CALibrate:SKEW:AUTO   139

Error Messages

This chapter describes the error messages and how they are generated. Use the command "ERRor?" on page 9 to return an error number and message. The possible causes for the generation of the error messages are also listed in Table 12 on page 42.

Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. If the error queue overflows, the last error in the queue is replaced with error –350, "Queue overflow." Anytime the error queue overflows, the oldest errors remain in the queue, and the most recent error is discarded. The length of the instrument's error queue is 30 (29 positions for the error messages, and 1 position for the "Queue overflow" message). Reading an error from the head of the queue removes that error from the queue, and opens a position at the tail of the queue for a new error. When all errors have been read from the queue, subsequent error queries return 0, "No error."

The error queue is cleared when any of the following occur:

- the instrument is powered up,
- a *CLS command is sent,
- the last item from the queue is read, or
- the instrument is switched from talk only to addressed mode on the front panel.

Error Numbers

The error numbers are grouped according to the type of error that is detected.

- +0 indicates no errors were detected.
- –100 to –199 indicates a command error was detected.
- –200 to –299 indicates an execution error was detected.
- –300 to –399 indicates a device-specific error was detected.
- –400 to –499 indicates a query error was detected.
- +1 to +32767 indicates an instrument-specific error has been detected.

Refer to the Keysight 86100A/B/C online Help for instrument specific errors.

Command Error

An error number in the range –100 to –199 indicates that an IEEE 488.2 syntax error has been detected by the instrument's parser. The occurrence of any error in this class sets the command error bit (bit 5) in the event status register and indicates that one of the following events occurred:

- An IEEE 488.2 syntax error was detected by the parser. That is, a controller-to-instrument message was received that is in violation of the IEEE 488.2 standard. This may be a data element that violates the instrument's listening formats, or a data type that is unacceptable to the instrument.
- An unrecognized header was received. Unrecognized headers include incorrect instrument-specific headers and incorrect or unimplemented IEEE 488.2 common commands.
- A Group Execute Trigger (GET) was entered into the input buffer inside of an IEEE 488.2 program message.

Events that generate command errors do not generate execution errors, instrument-specific errors, or query errors.

Execution Error

An error number in the range –200 to –299 indicates that an error was detected by the instrument's execution control block. The occurrence of any error in this class causes the execution error bit (bit 4) in the event status register to be set. It also indicates that one of the following events occurred:

- The program data following a header is outside the legal input range or is inconsistent with the instrument's capabilities.
- A valid program message could not be properly executed due to some instrument condition.

Execution errors are reported by the instrument after expressions are evaluated and rounding operations are completed. For example, rounding a numeric data element will not be reported as an execution error. Events that generate execution errors do not generate command errors, instrument specific errors, or query errors.

Device- or Instrument-Specific Error

An error number in the range of –300 to –399 or +1 to +32767 indicates that the instrument has detected an error caused by an instrument operation that did not properly complete. This may be due to an abnormal hardware or firmware condition. For example, this error may be generated by a self-test response error, or a full error queue. The occurrence of any error in this class causes the instrument-specific error bit (bit 3) in the event status register to be set.

Query Error

An error number in the range –400 to –499 indicates that the output queue control of the instrument has detected a problem with the message exchange protocol. An occurrence of any error in this class causes the query error bit (bit 2) in the event status register to be set. An occurrence of an error also means one of the following is true:

- An attempt is being made to read data from the output queue when no output is either present or pending.
- Data in the output queue has been lost.

Table 12   Error Messages Returned by Instrument Parser (Sheet 1 of 7)

Error	Returned String	Description
208	Incident Wave not Subtracted	Incident wave not subtracted. Turn response ___ off and then on to restore. The blank space (___) represents a TDR/TDT response waveform (response 1 through response 4).One of the following settings changed after performing a TDR/TDT calibration: record length, timebase, or channel bandwidth. The incident waveform can no longer be subtracted until original settings have been restored.
191	Response Turned Off	Response ___ turned off: Time base, record length or bandwidth changed. The blank space (___) represents a TDR/TDT response waveform (response 1 through response 4). Timescale or bandwidth no longer match because there has been a change in either timebase, record length, or bandwidth. The TDR/TDT response waveform has been turned off because of this mismatch.
190	Execution not Possible	Execution not possible: Calibration is required. The operation requires the calibration of the TDR/TDT waveform. For example, TDR calibration parameters cannot be saved to a file before the calibration procedure is performed.
178	Measured RN is invalid	The current measured RN is invalid or questionable. To apply RN stabilization in Jitter Mode, you must first have a valid RN measurement. Pressing the Get Measured RN button in the Advanced Jitter tab while a questionable RN measurement is displayed results in this error message.
177	Defined lead/lag for one/zero level not found in pattern	Defined lead/lag (%n: %n) for one/zero level not found in pattern. Using closest (%n: %n).

Table 12   Error Messages Returned by Instrument Parser (Sheet 2 of 7)

Error	Returned String	Description
172	Automatic tap calculation failed	Automatic tap calculation failed: error message
164	No Time Reference Set	No time reference set: Reference clock not present or amplitude too small . The instrument fails to set the time reference when the reference clock amplitude is too small or not present.
163	Execution not Possible	Execution not possible: No valid ___ destination available. A valid TDR/TDT destination is not specified.
162	Execution not Possible	Execution not possible: Select TDR/TDT destination. . No TDR/TDT destination has been specified.
151	Unable to connect to MATLAB	Unable to connect to MATLAB. Improper or corrupted MATLAB installation.
147	Printer Error	Printer error: Install and select a default printer. The instrument was unable to locate the default printer.
141	Turn on Source for Specified Measurement	Turn on ___ for the ___ measurement. The first blank space (___) represents the source that is required for the specific measurement (for example, an optical channel). The second blank space (___) is replaced with the name of the measurement (for example, jitter).
140	Exceeded Maximum ASCII List Length	Exceeded maximum ASCII list length. An attempt was made to load a waveform in ASCII format into waveform memory. Waveform size exceeded ASCII record limit of 128K. Contents of the file may be corrupted; the waveform file can not be loaded.
139	Unable to normalize the equalizer tap values	Unable to normalize the equalizer tap values: __. During normalization, the tap values are adjusted so that the DC gain (the sum of the tap values) is one while preserving the relative magnitudes of the tap values.
135	Jitter Exceeds Measurable Range	Jitter exceeds measurable range for this signal. Reduce jitter or retard edge speeds.. The jitter analysis provided in Jitter Mode cannot accurately measure jitter if the combined RJ and PJ (δ-δ) exceeds the rise or fall time of the signal.
133	Unable to characterize edges: <*string*>	Sampling level is not in the valid range. In Jitter Mode, the jitter sampling level determines the active sample area for the measurements. The default is a value that is 50% of the logic highs and lows values. If you change this setting above or below the acceptable limits, this message appears. Enter a units value for the Jitter Sampling Level that is inside the minimum and maximum values shown on the message line.
131	Error Saving Mask	Error saving mask: only parametric custom masks can be saved. A remote command was executed attempting to save a standard mask.
130	Error Loading Mask	Error loading mask, ____. The custom mask cannot be loaded due to illegal values, structure, or commands contained in the mask file.
127	All Labels are in Use	All 32 labels are in use, delete an old label before adding a new one. A maximum of 32 labels can be used.
125	Header Information not Valid	Header information is not valid. Error when loading a waveform from text (ASCII) data.
120	Execution not possible: Calibration does not match mainframe.	Execution not possible: Calibration does not match mainframe. The instrument attempted to load mainframe timebase calibration data that does not match the current mainframe model number or serial number.
117	You must start the mask test	You must start the mask test prior to calculating auto margin. Without a running mask test, the instrument can not determine the auto margins.
116	Too Many Points Sent	Too many points sent

Table 12   Error Messages Returned by Instrument Parser (Sheet 3 of 7)

Error	Returned String	Description
115	Network Path not Found	The network path was not found. The network path may be unavailable or unmapped. For example, if you attempt to load or save a file to an unmapped or non-existent network path.
112	Unknown File Type	Unknown file type. The contents of the file do not match the expected format. The file may be corrupted or may not be the correct type.
85	Incompatible Setup	Incompatible setup. A previously saved setup is incompatible, possibly due to an instrument software change.
79	Probe Attenuation (or Gain) Exceeds Limits	Probe attenuation (or gain) exceeds calibration limits. If the probe is broken or if the probe connections are not securely fastened, the probe calibration process fails.
78	No Significant Asynchronous Components Present	No significant asynchronous components present. When using the Enhanced Jitter Analysis Software (Option 200), scanning for asynchronous PJ components can only be done if there are significant PJ frequencies detected in the aliased jitter spectrum. If there are no components, or if the components are too small to be accurately identified, scanning will not take place.
74	Mainframe Calibration Required	Execution is not possible: Mainframe calibration is required. The mainframe calibration is required when a change in the temperature of the mainframe exceeds 15C compared to the temperature of the last mainframe timebase calibration ($\Delta T > 15°C$).
72	Could not Save Calibration Factors	Could not save calibration factors: Service is required. Possible errors during calibration.
69	Calibration in Progress	Execution not possible while calibration is in progress. Unable to execute some remote commands during calibration.
68	Service Mainframe Timebase Uncalibrated	Service mainframe timebase is uncalibrated.
67	Right Module Uncalibrated	Right module is uncalibrated Calibration is recommended.
66	Left Module Uncalibrated	Left module is uncalibrated. Calibration is recommended.
65	Module Memory Contents Obsolete	Module memory contents obsolete: reinitialize ___ module. The blank spaces (___) represent the module model number. An error due to a recent software upgrade may have occurred.
64	Module not Supported	The ___ module is not supported. The blank spaces (___) represent the module model number. An error due to a recent software upgrade may have occurred.
62	Unable to Communicate	Unable to communicate with ___ module: remove and reinsert firmly. The instrument can not recognize the module. The blank space (___) indicates which module has the error (left or right).
61	Memory Error Occurred	Memory error occurred in ___ module: Try reinstalling module. The plug-in module memory is incorrect. The blank space (___) indicates which module has the error (left or right).
59	Action cannot be performed on Jitter Data Memory	Action cannot be performed on Jitter Data Memory. When Jitter Data Memory is viewed, the Run, Stop Single, Clear Display, or Auto Scale functions are unavailable.
52	Disconnect Probe from Module	Probe must be disconnected from module. During a module calibration, the probe must be disconnected from the module. This ensures an accurate calibration.
48	No Measurements for Limit Test	No measurements are on for limit test. Unable to perform a measurement limit test through GPIB when there are no active measurements.
47	No Mask Loaded	No mask loaded. Unable to perform a mask test when a mask is not selected.
46	No Valid Mask Test Sources	No valid mask test sources turned on. Unable to perform a mask test from a remote command when a valid source is not available.

Table 12   Error Messages Returned by Instrument Parser (Sheet 4 of 7)

Error	Returned String	Description
41	Waveform Data is Not Valid	Waveform data is not valid. Remote command error occurred when the instrument attempted to save a waveform to disk or read the waveform over GPIB.
40	Command Execution not Possible	Command execution is not possible on the selected waveform. Unable to perform remote command.
39	Function Cannot be Performed	Function cannot be performed on the selected waveform. The function is not defined for this waveform type; therefore it cannot be performed.
38	Measurement Cannot be Performed	Measurement cannot be performed on the selected waveform. The measurement is not defined for this waveform type, and cannot be made.
36	Autoscale not Completed	Autoscale not completed. Unable to perform a complete autoscale.
15	Execution not Possible	Execution is not possible. This message occurs when a remote command is sent to a value on a channel that does not have the feature. For example, this message will occur when you try to set the channel wavelength on an electrical channel.
14	System Software Error	Fatal system software error occurred: Please cycle power. The instrument is still operable. Normally, the address (defect diagnostic) where the error occurred is also displayed. Record this address to help in servicing the instrument.
12	Source not Available	Signal source is not available. Signal source may be currently unavailable. For example, if you activate markers using remote commands without having a signal source activated.
11	Date and Time Incorrect	System date and time are incorrect. This error occurs when loading a waveform file with an invalid date or time stamp.
7	Mask Test Align Failed	Mask test align failed. The mask test align algorithm was not able to detect a signal compatible with the installed mask. This can occur when there are not enough points on an edge or when the required edges are not present.
6	Unrecognizable Waveform Format	The file format is incompatible with the file open operation.
2	Uninstalled Option	The ___ option is not installed. The instrument was unable to execute a feature that requires an upgrade option that is not installed in the instrument.
0	No error	The error queue is empty. Every error in the queue has been read (SYSTEM:ERROR? query) or the queue was cleared by power-up or *CLS.
-100	Command error	This is the generic syntax error used if the instrument cannot detect more specific errors.
-101	Invalid character	A syntactic element contains a character that is invalid for that type.
-102	Syntax error	An unrecognized command or data type was encountered.
-103	Invalid separator	The parser was expecting a separator and encountered an illegal character.
-104	Data type error	The parser recognized a data element different than one allowed. For example, numeric or string data was expected but block data was received.
-105	GET not allowed	A Group Execute Trigger was received within a program message.
-108	Parameter not allowed	More parameters were received than expected for the header.
-109	Missing parameter	Fewer parameters were received than required for the header.
-112	Program mnemonic too long	The header or character data element contains more than twelve characters.

Table 12   Error Messages Returned by Instrument Parser (Sheet 5 of 7)

Error	Returned String	Description
-113	Undefined header	The header is syntactically correct, but it is undefined for the instrument. For example, *XYZ is not defined for the instrument.
-121	Invalid character in number	An invalid character for the data type being parsed was encountered. For example, a "9" in octal data.
-123	Exponent too large	Number is too large or too small to be represented internally.
-124	Too many digits	The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros.
-128	Numeric data not allowed	A legal numeric data element was received, but the instrument does not accept one in this position for the header.
-131	Invalid suffix	The suffix does not follow the syntax described in IEEE 488.2 or the suffix is inappropriate for the instrument.
-138	Suffix not allowed	A suffix was encountered after a numeric element that does not allow suffixes.
-141	Invalid character data	Either the character data element contains an invalid character or the particular element received is not valid for the header.
-144	Character data too long	
-148	Character data not allowed	A legal character data element was encountered where prohibited by the instrument.
-150	String data error	This error can be generated when parsing a string data element. This particular error message is used if the instrument cannot detect a more specific error.
-151	Invalid string data	A string data element was expected, but was invalid for some reason. For example, an END message was received before the terminal quote character.
-158	String data not allowed	A string data element was encountered but was not allowed by the instrument at this point in parsing.
-160	Block data error	This error can be generated when parsing a block data element. This particular error message is used if the instrument cannot detect a more specific error.
-161	Invalid block data	
-168	Block data not allowed	A legal block data element was encountered but was not allowed by the instrument at this point in parsing.
-170	Expression error	This error can be generated when parsing an expression data element. It is used if the instrument cannot detect a more specific error.
-171	Invalid expression	
-178	Expression data not allowed	Expression data was encountered but was not allowed by the instrument at this point in parsing.
-200	Execution error	This is a generic syntax error which is used if the instrument cannot detect more specific errors.
-220	Parameter error	Indicates that a program data element related error occurred.
-221	Settings conflict	Indicates that a legal program data element was parsed but could not be executed due to the current device state.
-222	Data out of range	Indicates that a legal program data element was parsed but could not be executed because the interpreted value is outside the legal range defined by the instrument.

Table 12   Error Messages Returned by Instrument Parser (Sheet 6 of 7)

Error	Returned String	Description
-223	Too much data	Indicates that a legal program data element of block, expression, or string type was received that contained more data than the instrument could handle due to memory or related instrument-specific requirements.
-224	Illegal parameter value	Used where exact value, from a list of possibles, was expected.
-225	Out of memory	The device has insufficient memory to perform the requested operation.
-231	Data questionable	Indicates that measurement accuracy is suspect.
-240	Hardware error	Indicates that a legal program command or query could not be executed because of a hardware problem in the device.
-241	Hardware missing	Indicates that a legal program command or query could not be executed because of missing device hardware; for example, an option was not installed, or current module does not have hardware to support command or query. Definition of what constitutes missing hardware is completely device-specific or module specific.
-250	Mass storage error	Indicates that a mass storage error occurred.
-251	Missing mass storage	Indicates that a legal program command or query could not be executed because of missing mass storage; for example, an option that was not installed.
-252	Missing media	Indicates that a legal program command or query could not be executed because of a missing media; for example, no disk.
-253	Corrupt media	Indicates that a legal program command or query could not be executed because of corrupt media; for example, bad disk or wrong format.
-254	Media full	Indicates that a legal program command or query could not be executed because the media was full; for example, there is no room on the disk.
-255	Directory full	Indicates that a legal program command or query could not be executed because the media directory was full.
-256	File name not found	Indicates that a legal program command or query could not be executed because the file name on the device media was not found; for example, an attempt was made to read or copy a nonexistent file.
-257	File name error	Indicates that a legal program command or query could not be executed because the file name on the device media was in error; for example, an attempt was made to copy to a duplicate file name.
-258	Media protected	Indicates that a legal program command or query could not be executed because the media was protected; for example, the write-protect tab on a disk was present.
-300	Service specific error	
-310	System error	Indicates that a system error occurred.
-340	Calibration failed	Indicates that a calibration has failed.
-350	Queue overflow	Indicates that there is no room in the error queue and an error occurred but was not recorded.
-400	Query error	This is the generic query error.
-410	Query INTERRUPTED	
-420	Query UNTERMINATED	
-430	Query DEADLOCKED	

Table 12   Error Messages Returned by Instrument Parser (Sheet 7 of 7)

Error	Returned String	Description
-440	Query UNTERMINATED after indefinite response	

Language Compatibility

This section lists Keysight 83480A commands that are not used in the 86100A/B/C/D.

Table 13   Keysight 83480A/54750A Commands Not Used in the Instrument (Sheet 1 of 6)

Programming Commands/Queries	Replacement Commands/Queries
Common Commands	
*LRN	SYSTEM:SETUP
Root Level Commands	
:AER?	No replacement
:ERASe	No replacement
:HEEN	:AEEN
:MENU	No replacement
:MERGe	No replacement
:STORe:PMEMory1	No replacement
:TEER	No replacement
System Commands :SYSTem	
:SYSTem:KEY	No replacement
Calibration Commands :CALibrate	
:CALibrate:FRAMe:CANCel	:CALibrate:CANcel
:CALibrate:FRAMe:CONTinue	:CALibrate:CONTinue
:CALibrate:FRAMe:DATA	No replacement
:CALibrate:FRAMe:DONE?	:CALibrate:STATus?
:CALibrate:FRAMe:MEMory?	No replacement
:CALibrate:PLUGin:ACCuracy	:CALibrate:MODule:STATus
:CALibrate:PLUGin:CANCel	:CALibrate:CANcel
:CALibrate:PLUGin:CONTinue	:CALibrate:CONTinue
:CALibrate:PLUGin:DONE?	:CALibrate:STATus?
:CALibrate:PLUGin:MEMory?	No replacement
:CALibrate:PLUGin:OFFSet	:CALibrate:MODule:OFFSet
:CALibrate:PLUGin:OPOWer	:CALibrate:MODule:OPOWer
:CALibrate:PLUGin:OPTical	:CALibrate:MODule:OPTical
:CALibrate:PLUGin:OWAVelength	:CALibrate:MODule:OWAVelength
:CALibrate:PLUGin:TIME?	:CALibrate:MODule:TIME?
:CALibrate:PLUGin:VERTical	:CALibrate:MODule:VERtical

Table 13   Keysight 83480A/54750A Commands Not Used in the Instrument (Sheet 2 of 6)

:CALibrate:PROBe	:CALibrate:PROBe CHANnel<N>
Channel Commands :CHANnel	
:CHANnel<N>:AUTOscale	:AUToscale
:CHANnel<N>:SKEW	:CALibrate:SKEW
Disk Commands :DISK	
:DISK:DATA?	No replacement
:DISK:FORMat	No replacement
Display Commands :DISPlay	
:DISPlay:ASSign	No replacement
:DISPlay:CGRade	:SYSTem:MODE EYE
:DISPlay:CGRade?	:SYSTem:MODE?
:DISPlay:COLumn	:DISPlay:LABel
:DISPlay:DATA	:WAVeform:DATA
:DISPlay:DWAVeform	No replacement
:DISPlay:FORMat	No replacement
:DISPlay:INVerse	:DISPlay:LABel
:DISPlay:LINE	:DISPlay:LABel
:DISPlay:MASK	No replacement
:DISPlay:ROW	:DISPlay:LABel
:DISPlay:SOURce	No replacement
:DISPlay:STRing	:DISPlay:LABel
:DISPlay:TEXT	:DISPlay:LABel:DALL
FFT Commands :FFT	
FFT is not available in the 86100A/B.	
Function Commands :FUNCtion	
:FUNCtion<N>:ADD	No replacement
:FUNCtion<N>:BWLimit	No replacement
:FUNCtion<N>:DIFFerentiate	No replacement
:FUNCtion<N>:DIVide	No replacement
:FUNCtion<N>:FFT	No replacement, FFT not available
:FUNCtion<N>:INTegrate	No replacement
:FUNCtion<N>:MULTiply	No replacement
:FUNCtion<N>:ONLY	:FUNCtion<N>:MAGNify

Table 13   Keysight 83480A/54750A Commands Not Used in the Instrument (Sheet 3 of 6)

Hardcopy Commands :HARDcopy	
:HARDcopy:ADDRess	:HARDcopy:DPRinte
:HARDcopy:BACKground	:HARDcopy:IMAGe INVert
:HARDcopy:BACKground?	No replacement
:HARDcopy:DESTination	No replacement
:HARDcopy:DEVice	No replacement
:HARDcopy:FFEed	No replacement
:HARDcopy:FILename	No replacement
:HARDcopy:LENGth	No replacement
:HARDcopy:MEDia	No replacement
Histogram Commands :HISTogram	
:HISTogram:RRATe	:DISPlay:RRATe
:HISTogram:RUNTil	:ACQuire:RUNTil
:HISTogram:SCALe	:HISTogram:SCALe:SIZE
:HISTogram:SCALe:OFFSet	:HISTogram:SCALe:SIZE
:HISTogram:SCALe:RANGe	:HISTogram:SCALe:SIZE
:HISTogram:SCALe:SCALe	:HISTogram:SCALe:SIZE
:HISTogram:SCALe:TYPE	:HISTogram:SCALe:SIZE
Limit Test Commands :LTESt	
:LTESt:SSCReen:DDISk:BACKground	:LTESt:SSCReen:IMAGe
:LTESt:SSCReen:DDISk:MEDia	No replacement
:LTESt:SSCReen:DDISk:PFORmat	No replacement
:LTESt:SSCReen:DPRinter:ADDRess	No replacement
:LTESt:SSCReen:DPRinter:BACKground	No replacement
:LTESt:SSCReen:DPRinter:MEDia	No replacement
:LTESt:SSCReen:DPRinter:PORT	No replacement
:LTESt:SSUMmary:ADDRess	No replacement
:LTESt:SSUMmary:MEDia	No replacement
:LTESt:SSUMmary:PFORmat	No replacement
:LTESt:SSUMmary:PORT	No replacement
Marker Commands :MARKer	
:MARKer:CURSor?	No replacement. Use individual queries.
:MARKer:MEASurement:READout	No replacement

Table 13   Keysight 83480A/54750A Commands Not Used in the Instrument (Sheet 4 of 6)

:MARKer:MODE	:MARKer:STATe
:MARKer:MODE?	No replacement
:MARKer:TDELta?	:MARKer:XDELta?
:MARKer:TSTArt	:MARKer:X1Position
:MARKer:TSTOp	:MARKer:X2Position
:MARKer:VDELta	:MARKer:YDELta
:MARKer:VSTArt	:MARKer:Y1Position
:MARKer:VSTOp	:MARKer:Y2Position
Mask Test Commands :MTESt	
:MTESt:AMASk:CReate	No replacement
:MTESt:AMASk:SOURce	No replacement
:MTESt:AMASk:UNITs	No replacement
:MTESt:AMASk:XDELta	No replacement
:MTESt:AMASk:YDELta	No replacement
:MTESt:AMODe	No replacement
:MTESt:COUNt:FWAVeforms?	MTESt:COUNt:HITS? TOTal
:MTESt:FENable	No replacement
:MTESt:MASK:DEFine	No replacement [a]
:MTESt:POLYgon:DEFine	No replacement [a]
:MTESt:POLYgon:DELete	No replacement [a]
:MTESt:POLYgon:MOVE	No replacement [a]
:MTESt:RECall	:MTESt:LOAD
:MTESt:SAVE	No replacement
:MTESt:SSCReen:DDISk:BACKground	:MTESt:SSCReen:IMAGe
:MTESt:SSCReen:DDISk:MEDia	No replacement
:MTESt:SSCReen:DDISk:PFORmat	No replacement
:MTESt:SSCReen:DPRinter	No replacement
:MTESt:SSCReen:DPRinter:ADDRess	No replacement
:MTESt:SSCReen:DPRinter:BACKground	No replacement
:MTESt:SSCReen:DPRinter:MEDia	No replacement
:MTESt:SSCReen:DPRinter:PFORmat	No replacement
:MTESt:SSCReen:DPRinter:PORT	No replacement
:MTESt:SSUMmary:ADDRess	No replacement

Table 13   Keysight 83480A/54750A Commands Not Used in the Instrument (Sheet 5 of 6)

:MTESt:SSUMmary:BACKground	No replacement
:MTESt:SSUMmary:MEDia	No replacement
:MTESt:SSUMmary:PFORmat	No replacement
:MTESt:SSUMmary:PORT	No replacement
Measure Commands :MEASure	
:MEASure:CGRade:ERCalibrate	:CALibrate:ERATio:STARt CHANnel<N>
:MEASure:CGRade:ERFactor	No replacement
:MEASure:CGRade:QFACtor	:MEASure:CGRade:ESN
:MEASure:FFT	No replacement. FFT not available.
:MEASure:HISTogram:HITS	Query only
:MEASure:HISTogram:MEAN	Query only
:MEASure:HISTogram:MEDian	Query only
:MEASure:HISTogram:M1S	Query only
:MEASure:HISTogram:M2S	Query only
:MEASure:HISTogram:OFFSET?	No replacement
:MEASure:HISTogram:PEAK	Query only
:MEASure:HISTogram:PP	Query only
:MEASure:PREShoot	No replacement
:MEASure:STATistics	No replacement. Statistics always on.
:MEASure:TEDGe	Query only
:MEASure:VLOWer	No replacement
:MEASure:VMIDdle	No replacement
:MEASure:VTIMe	Query only
:MEASure:VUPPer	No replacement
Timebase Commands :TIMebase	
:TIMebase:DELay	:TIMebase:POSition
:TIMebase:VIEW	No replacement
:TIMebase:WINDow:DELay	No replacement
:TIMebase:WINDow:POSition	No replacement
:TIMebase:WINDow:RANGe	No replacement
:TIMebase:WINDow:SCALe	No replacement
:TIMebase:WINDow:SOURce	No replacement
Trigger Commands :TRIGger	

Table 13   Keysight 83480A/54750A Commands Not Used in the Instrument (Sheet 6 of 6)

:TRIGger:SWEep	:TRIGger:SOURce FRUN
:TRIGger:SWEep?	:TRIGger:SOURce?
:TRIGger<N>:BWLimit	:TRIGger:BWLimit and :TRIGger:GATed
:TRIGger<N>:PROBe	:TRIGger:ATTenuation
Waveform Commands :WAVeform	
:WAVeform:COMPlete	No replacement
:WAVeform:COUPling	No replacement
:WAVeform:VIEW?	No replacement

[a] Refer to the Infiniium DCA Online Help to view information about defining custom masks.

2 Programming Examples

KEYSIGHT
TECHNOLOGIES

Listings of the C sample programs in this section include:

General Measurement Example

In this example, the main function inclues a call to init_IO() which initializes the instrument and interface so that the instrument can capture data and perform measurements on the data. At the start of the program, global symbols are defined which will be used to store and convert the digitized data to time and voltage values. In the transfer_data function, the header string (header_str) resembles the following string when the information is stripped off: #510225. The left-most "5" defines the number of digits that follow (10225). The example number "10225" is the number of points in the waveform. The information is stripped off of the header to get the number of data bytes that need to be read from the instrument. In the convert_data function, the data values are returned as digitized samples (sometimes called quantization levels or q-levels). These data values must be converted into voltage and time values. In the store_csv function, the time and voltage information of the waveform is stored in integer format, with the time stored first, followed by a comma, and the voltage stored second.

File: init.c

```
/* init. c */

/*
*   Command Order Example.  This program demonstrates the order of commands
*   suggested for operation of the Agilent 86100 analyzer via GPIB.
*   This program initializes the scope, acquires data, performs
*   automatic measurements, and transfers and stores the data on the
*   PC as time/voltage pairs in a comma-separated file format useful
*   for spreadsheet applications. It assumes a SICL INTERFACE exists
*   as 'gpib7' and an Agilent 86100 analyzer at address 7.
*   It also requires the cal signal attached to Channel 1.
*
*   See the README file on the demo disk for development and linking information.
*/

#include <stdio.h>      /* location of: printf ( ) */
#include <stdlib.h>     /* location of: atof(), atoi ( ) */
#include "hpibdecl.h"   /* prototypes, global declarations, constants */

void initialize ( );    /* initialize the scope */
void acquire_data ( );  /* digitize signal */
void auto_measurements ( );/* perform built-in automatic measurements */
void transfer_data ( ); /* transfers waveform data from scope to PC */
void convert_data ( );  /* converts data to time/voltage values */
void store_csv ( );     /* stores time/voltage pairs to comma-separated variable file format */


/* GLOBALS */
int count;
double xorg,xref,xinc;  /* values necessary for conversion of data */
double yorg,yref,yinc;
int Acquired_length;
char data [MAX_LENGTH]; /* data buffer */
double time_value [MAX_LENGTH];/* time value of data */
double volts [MAX_LENGTH];/* voltage value of data */

void main( void )
```

```
{
        /* initialize interface and device sessions */
        /* note: routine found in sicl_IO.c or natl_IO.c  */
        init_IO ( );


        initialize ( );  /* initialize the scope and interface and set up SRQ  */
        acquire_data ( );/* capture the data */
        auto_measurements ( );/* perform automated measurements on acquired data */
        transfer_data ( );/* transfer waveform data to the PC from scope */
        convert_data ( );/* convert data to time/voltage pairs */
        store_csv ( );   /* store the time/voltage pairs as csv file */
        close_IO ( );    /* close interface and device sessions */
                         /* note: routine found in sicl_IO.c or natl_IO.c  */
} /* end main ( ) */


/*
*  Function name:  initialize
*  Parameters:  none
*  Return value:  none
*  Description:  This routine initializes the analyzer for proper
*  acquisition of data. The instrument is reset to a known state and the
*  interface is cleared. System headers are turned off to allow faster
*  throughput and immediate access to the data values requested by queries.
*  The analyzer time base, channel, and trigger subsystems are then
*  configured. Finally, the acquisition subsystem is initialized.
*/

void initialize (  )
{
        write_IO ("*RST");    /* reset scope - initialize to known state */
        write_IO ("*CLS");    /* clear status registers and output queue */

        write_IO (":SYSTem:HEADer OFF"); /* turn off system headers */
    /* initialize time base parameters to center reference, 2 ms full-scale (200 us/div), and 20 us
    delay */
        write_IO (":TIMebase:REFerence CENTer;RANGe 2e-3;POSition 20e-6");
    /* initialize Channel1 1.6V full-scale (200 mv/div); offset -400mv */
        write_IO (":CHANnel1:RANGe 1.6;OFFSet -400e-3");
    /* initialize trigger info: channel1 signal on positive slope at 300mv */
        write_IO (":TRIGger:SOURce FPANel;SLOPe POSitive");
        write_IO (":TRIGger:LEVel-0.40");
    /* initialize acquisition subsystem */
        /* Real time acquisition - no averaging; record length 4096 */
        write_IO (":ACQuire:AVERage OFF;POINts 4096");
} /* end initialize ( ) */

/*
*  Function name:  acquire_data
*  Parameters: none
*  Return value:  none
*  Description:  This routine acquires data according to the current instrument settings.
*/
void acquire_data ( )
{
/*
*  The root level :DIGitize command is recommended for acquisition of new
*  data. It will initialize data buffers, acquire new data, and ensure that
*  acquisition criteria are met before acquisition of data is stopped.
*  The captured data is then available for measurements, storage, or transfer
*  to a PC.  Note that the display is automatically turned off by the
*  :DIGitize command and must be turned on to view the captured data.
*/
    write_IO (":DIGitize CHANnel1");
        write_IO (":CHANnel1:DISPlay ON");/* turn on channel 1 display which is turned off by the
    :DIGitize command */

} /*  end acquire_data() */

/*
*  Function name:  auto_measurements
*  Parameters:  none
*  Return value:  none
*  Description:  This routine performs automatic measurements of volts
*  peak-to-peak and period on the acquired data. It also demonstrates
*  two methods of error detection when using automatic measurements.
*/
```

```
void auto_measurements ( )
{
        float period, vpp;
        unsigned char vpp_str[16];
        unsigned char period_str[16];
        int bytes_read;

/*
*   Error checking on automatic measurements can be done using one of two methods.
*   The first method requires that you turn on results in the Measurements
*   subsystem using the command :MEASure:SEND ON.  When this is on, the analyzer
*   will return the measurement and a result indicator.  The result flag is zero
*   if the measurement was successfully completed, otherwise a non-zero value is
*   returned which indicates why the measurement failed. See the Programmer's Manual
*   for descriptions of result indicators.

*   The second method simply requires that you check the return value of the
*   measurement. Any measurement not made successfully will return with the value
*   +9.999E37. This could indicate that either the measurement was unable to be
*   performed, or that insufficient waveform data was available to make the
*   measurement.

* METHOD ONE - turn on results to indicate whether the measurement completed
* successfully. Note that this requires transmission of extra data from the scope.
*/

        write_IO (":MEASure:SEND ON");  /* turn results on */

        /* query -- volts peak-to-peak channel 1*/
        write_IO (":MEASure:VPP? CHANnel1");
      bytes_read = read_IO (vpp_str,16L);/* read in value and result flag */
      if (vpp_str[bytes_read-2] != '0')
            printf ("Automated vpp measurement error with result %c\n", vpp_str[bytes_read-2]);
        else
            printf ("VPP is %f\n", (float)atof (vpp_str));
      write_IO (":MEASure:PERiod? CHANnel1");/* period channel 1 */
        bytes_read = read_IO (period_str,16L);/* read in value and result flag */

        if (period_str[bytes_read-2] != '0')
            printf ("Automated period measurement error with result %c\n", period_str [bytes_read-2]);
        else
            printf ("Period is %f\n", (float) atof (period_str));


/* METHOD TWO - perform automated measurements and error checking with :MEAS:SEND OFF */

        period = (float) 0;
        vpp = (float) 0;

        /* turn off results */
        write_IO (":MEASure:SEND OFF");

        write_IO (":MEASure:PERiod? CHANnel1");/* period channel 1 */
        bytes_read = read_IO (period_str,16L);/* read in value and result flag */

        period = (float) atof (period_str);

        if ( period > 9.99e37 )
            printf ("\nPeriod could not be measured.\n");
        else
            printf ("\nThe period of channel 1 is %f seconds.\n", period );

        write_IO (":MEASure:VPP? CHANnel1");
        bytes_read = read_IO ( vpp_str,16L );

        vpp = (float) atof (vpp_str);

        if ( vpp > 9.99e37 )
            printf ("Peak-to-peak voltage could not be measured.\n");
        else
            printf ("The voltage peak-to-peak is %f volts.\n", vpp );

} /* end auto_measurements ( ) */

/*
*   Function name:  transfer_data
*   Parameters:  none
```

```
*  Return value:  none
*  Description:  This routine transfers the waveform conversion factors and waveform data to the PC.
*/

void transfer_data (  )
{
      int header_length;
      char header_str[8];
      char term;

      char xinc_str[32],xorg_str[32],xref_str[32];
      char yinc_str[32],yref_str[32],yorg_str[32];

      int bytes_read;

      /* waveform data source channel 1 */
      write_IO (":WAVeform:SOURce CHANnel1");
      /* setup transfer format */
      write_IO (":WAVeform:FORMat BYTE");
   /* request values to allow interpretation of raw data */
      write_IO (":WAVeform:XINCrement?");
      bytes_read = read_IO (xinc_str,32L);
      xinc = atof (xinc_str);

      write_IO (":WAVeform:XORigin?");
      bytes_read = read_IO (xorg_str,32L);
      xorg = atof (xorg_str);

      write_IO (":WAVeform:XREFerence?");
      bytes_read = read_IO (xref_str,32L);
      xref = atof (xref_str);

      write_IO (":WAVeform:YINCrement?");
      bytes_read = read_IO (yinc_str,32L);
      yinc = atof (yinc_str);

      write_IO (":WAVeform:YORigin?");
      bytes_read = read_IO (yorg_str,32L);
      yorg = atof (yorg_str);

      write_IO (":WAVeform:YREFerence?");
      bytes_read = read_IO (yref_str,32L);
      yref = atof (yref_str);

      write_IO (":WAVeform:DATA?");/* request waveform data */
      bytes_read = read_IO (data,1L); /* ignore leading # */
      bytes_read = read_IO (header_str,1L);/* input byte counter */
      header_length = atoi (header_str);

      /* read number of points - value in bytes */
      bytes_read = read_IO (header_str,(long)header_length);

      Acquired_length = atoi (header_str);/* number of bytes */

      bytes_read = read_IO (data,Acquired_length);  /* input waveform data */
      bytes_read = read_IO (&term,1L);/* input termination character */

} /* end transfer_data ( ) */

/*
*  Function name:  convert_data
*  Parameters:  none
*  Return value:  none
*  Description:  This routine converts the waveform data to time/voltage
*  information using the values that describe the waveform.  These values are
*  stored in global arrays for use by other routines.
*/

void convert_data ( )
{
      int i;

      for (i = 0; i < Acquired_length; i++)
      {
          time_value[i] = ((i - xref) * xinc) + xorg;    /* calculate time info */
          volts[i] = ((data[i] - yref) * yinc) + yorg;  /* calculate volt info */
      }
} /* end convert_data ( ) */
```

```
/*
*  Function name:  store_csv
*  Parameters:  none
*  Return value:  none
*  Description:  This routine stores the time and voltage information about
*  the waveform as time/voltage pairs in a comma-separated variable file
*  format.
*/

void store_csv ( )
{
      FILE *fp;
      int i;

      fp = fopen ("pairs.csv","wb");   /* open file in binary mode - clear file if already exists */
      if (fp != NULL)
      {
          for (i = 0; i < Acquired_length; i++)
          {
              /* write time,volt pairs to file */
              fprintf ( fp,"%e,%lf\n",time_value[i],volts[i]);

          }
          fclose ( fp );                 /* close file */
      }
      else
          printf ("Unable to open file 'pairs.csv'\n");

}  /* end store_csv ( ) */
```

Service Request Example

The sample C program, gen_srq.c, shows how to initialize the interface and instrument and generate a service request. The init_IO() function initializes the instrument and interface and sets up and generates a service request. In the initialize function, the *RST command is a common command that resets the instrument to a known default configuration. Using this command ensures that the instrument is in a known state before you configure it. *RST ensures very consistent and repeatable results. Without *RST, a program may run one time, but it may give different results in following runs if the instrument is configured differently. *RST defaults the instrument to a set configuration so that the program can proceed from the same state each time. The *CLS command clears the status registers and the output queue. AUToscale finds and displays all signals that are attached to the instrument. You should program the instrument's time base, channel, and trigger for the specific measurement to be made, as you would do from the front panel, and use whatever other commands are needed to configure the instrument for the desired measurement.

File: gen_srq.c

```
/* gen_srq.c */

/*
 *  This example programs initializes the Agilent 86100 scope, runs an
 *  autoscale, then generates and responds to a Service Request from the
 *  scope. The program assumes an Agilent 86100 at address 7, an interface card
 *  at interface select code 7, and a signal source attached to channel 1.
 */

#include <stdio.h>      /* location of: printf ( ) */
#include "hpibdecl.h"

void initialize ( );
void setup_SRQ ( );
void create_SRQ ( );

void main ( void )
{
        init_IO ( );      /* initialize interface and device sessions */
        initialize ( );   /* initialize the scope and interface */
        setup_SRQ ( );    /* enable SRQs on scope and set up SRQ handler */
        create_SRQ ( );   /* generate SRQ */
        close_IO ( );     /* close interface and device sessions */

} /* end main ( ) */

/*
 *  Function name:  initialize
 *  Parameters:  none
 *  Return value:  none
 *  Description:  This routine initializes the analyzer for proper acquisition of data.
 * The instrument is reset to a known state and the interface is cleared.
 * System headers are turned off to allow faster throughput and immediate access
 * to the data values requested by queries. The analyzer performs an autoscale to acquire waveform data.
 */
void initialize (   )
{
        write_IO ("*RST");    /* reset scope - initialize to known state */
        write_IO ("*CLS");    /* clear status registers and output queue */
        write_IO (":SYSTem:HEADer OFF"); /* turn off system headers */
        write_IO (":AUToscale");/* perform autoscale */

} /* end initialize ( ) */

/*
 *  Function name:  setup_SRQ
 *  Parameters:  none
 *  Return value:  none
 *  Description:  This routine initializes the device to generate Service
 *  Requests. It sets the Service Request Enable Register Event Status Bit
 *  and the Standard Event Status Enable Register to allow SRQs on Command
 *  or Query errors.
 */

void setup_SRQ (   )
{
        /* Enable Service Request Enable Register - Event Status Bit */
```

```
        write_IO ("*SRE 32");

        /* Enable Standard Event Status Enable Register enable Command Error - bit 4 - value 32 Query
    Error - bit 1 - value 4 */
        write_IO ("*ESE 36");

} /* end setup_SRQ ( ) */

/*
*   Function name:  create_SRQ
*   Parameters:  none
*   Return value:  none
*   Description:   This routine sends two illegal commands to the scope which will generate an
* SRQ and will place two error strings in the error queue. The scope ID is requested to allow
* time for the SRQ to be generated.  The ID string will contain a leading character which
* is the response placed in the output queue by the interrupted query.
*/
void create_SRQ ( )
{
        char buf [256] = { 0 };  //read buffer for id string
        int bytes_read = 0;
        int srq_asserted;

        /* Generate query error (interrupted query)*/
        /* send legal query followed by another command other than a read query response */
        write_IO (":CHANnel2:DISPlay?");

        write_IO (":CHANnel2:DISPlay OFF");

        /* Generate command error - send illegal header */
        write_IO (":CHANnel:DISPlay OFF");

        /* get instrument ID - allow time for SRQ to set */
        write_IO ("*IDN?");
        bytes_read = read_IO (buf,256L);

        /* add NULL to end of string */
        buf [bytes_read] = '\0';

        printf ( "%s\n", buf);
        srq_asserted = check_SRQ ( );
        if ( srq_asserted )
            srq_handler ( );
} /* end create_SRQ ( ) */
```

SRQ From GPIB Device Example

File: srq.c

```
/* file:  srq.c */

/*  This file contains the code to handle Service Requests from an GPIB device */

#include <stdio.h> /* location of printf ( ), fopen ( ), and fclose ( ) */
#include "hpibdecl.h"

/*
*  Function name:  srq_handler
*  Parameters:  none
*  Return value:  none
*  Description:  This routine services the scope when an SRQ is generated.
* An error file is opened to receive error data from the scope.
*/

void srq_handler ( )
    {
        FILE *fp;
        unsigned char statusbyte = 0;
        int i =0;
        int more_errors = 0;
        char error_str[64] ={0};
        int bytes_read;
        int srq_asserted = TRUE;

        srq_asserted = check_SRQ ( );

        while (srq_asserted)
        {
        statusbyte = read_status ( );

        if ( statusbyte & SRQ_BIT )
        {
            fp = fopen ( "error_list","wb" );/* open error file */
            if (fp == NULL)
                printf ("Error file could not be opened.\n");
    /* read error queue until no more errors */
            more_errors = TRUE;
            while ( more_errors )
            {
                write_IO (":SYSTEM:ERROR? STRING");
                bytes_read = read_IO (error_str, 64L);

                error_str[bytes_read] = '\0';

                /* write error msg to std IO */
                printf ("Error string:%s\n", error_str );

            if (fp != NULL)
                /* write error msg to file*/
                fprintf (fp,"Error string:%s\n", error_str );

                if ( error_str[0] == '0' )
                {
                    /* Clear event registers and queues,except output */
                  write_IO("*CLS");

                  more_errors = FALSE;

                    if ( fp != NULL)
                        fclose ( fp );
                }
                for (i=0;i<64;i++)      /* clear string */
                    error_str[i] = '\0';

            } /* end while (more_errors) */
        }
        else
        {
            printf (" SRQ not generated by scope.\n ");/* scope did not cause SRQ */
        }
        srq_asserted = check_SRQ ( );    /* check for SRQ line status */

        }/* end while ( srq_asserted ) */
```

```
}/* end srq_handler */
```

Learn String Example

File: learnstr.c

```
/* learnstr.c */

/*
*   This example program initializes the Agilent 86100 scope, runs autoscale to
*   acquire a signal, queries for the learnstring, and stores the learnstring
*   to disk.  It then allows the user to change the setup, then restores the
*   original learnstring. It assumes that a signal is attached to the scope.
*/

#include <stdio.h>              /* location of: printf ( ), fopen ( ), fclose ( ), fwrite ( ),getchar */
#include "hpibdecl.h"

void initialize ( );
void store_learnstring ( );
void change_setup ( );
void get_learnstring ( );

void main ( void )
{
        init_IO ( );            /* initialize device and interface  */
                                /* Note: routine found in sicl_IO.c or natl_IO.c */
        initialize ( );         /* initialize the scope and interface, and set up SRQ */
        store_learnstring ( );/* request learnstring and store */
        change_setup ( );       /* request user to change setup */
        get_learnstring ( );  /* restore learnstring */
        close_IO ( );           /* close device and interface sessions */
                                /* Note: routine found in sicl_IO.c or natl_IO.c */


} /* end main */
/*
*   Function name:  initialize
*   Parameters:  none
*   Return value:  none
*   Description:  This routine initializes the analyzer for proper acquisition of data.
* The instrument is reset to a known state and the interface is cleared.
* System headers are turned off to allow faster throughput and immediate access to the data values
requested by queries.
*   Autoscale is performed to acquire a waveform. The signal is then
*   digitized, and the channel display is turned on following the acquisition.
*/

void initialize (   )
{
        write_IO ("*RST");      /* reset scope - initialize to known state */
        write_IO ("*CLS");      /* clear status registers and output queue */

        write_IO (":SYSTem:HEADer ON");/* turn on system headers */

       /* initialize Timebase parameters to center reference, 2 ms full-scale (200 us/div), and 20 us
    delay */
        write_IO (":TIMebase:REFerence CENTer;RANGe 5e-3;POSition 20e-6");

        /* initialize Channel1 1.6v full-scale (200 mv/div); offset -400mv */
        write_IO (":CHANnel1:RANGe 1.6;OFFSet -400e-3");

        /* initialize trigger info: channel1 signal on positive slope at 300mv */
        write_IO (":TRIGger:SOURce FPANel;SLOPe POSitive");
        write_IO (":TRIGger:LEVel-0.40");

        /* initialize acquisition subsystem */
        /* Real time acquisition - no averaging; record length 4096 */
        write_IO (":ACQuire:AVERage OFF;POINts 4096");

} /* end initialize ( ) */


/*
*   Function name:  store_learnstring
*   Parameters:  none
*   Return value:  none
*   Description:  This routine requests the system setup known as a learnstring.
* The learnstring is read from the scope and stored in a file called Learn2.
```

```
*/

void store_learnstring (   )
{
        FILE *fp;
        unsigned char setup[MAX_LRNSTR] ={0};
        int actualcnt = 0;

        write_IO (":SYSTem:SETup?");     /* request learnstring */
        actualcnt = read_IO (setup, MAX_LRNSTR);

        fp = fopen ( "learn2","wb");


        if ( fp != NULL )
        {
            fwrite ( setup,sizeof (unsigned char), (int) actualcnt,fp);
            printf ("Learn string stored in file Learn2\n");

            fclose ( fp );
        }
        else
            printf ("Error in file open\n");

}/* end store_learnstring */

/*
*  Function name:  change_setup
*  Parameters:  none
*  Return value:  none
*  Description:  This routine places the scope into local mode to allow the customer to change the
system setup.
*/

void change_setup ( )
{
        printf ("Please adjust setup and press ENTER to continue.\n");
        getchar();

} /* end change_setup */

/*
*  Function name:  get_learnstring
*  Parameters:  none
*  Return value:  none
*  Description:  This routine retrieves the system setup known as a
*  learnstring from a disk file called Learn2. It then restores the system setup to the scope.
*/

void get_learnstring ( )
{
        FILE *fp;
        unsigned char setup[MAX_LRNSTR];
        unsigned long count = 0;

        fp = fopen ( "learn2","rb");

        if ( fp != NULL )
        {
            count = fread ( setup,sizeof(unsigned char),MAX_LRNSTR,fp);

            fclose ( fp );
        }
        write_lrnstr (setup,count);           /* send learnstring */
        write_IO (":RUN");

}/* end get_learnstring */
```

SICL I/O Example

File: sicl_IO.c

```
/* sicl_IO.c */

#include <stdio.h>                /* location of: printf ( ) */
#include <string.h>              /* location of: strlen ( ) */
#include "hpibdecl.h"

/*  This file contains IO and initialization routines for the SICL libraries. */
/*
*  Function name:  init_IO
*  Parameters:  none
*  Return value:  none
*  Description:  This routine initializes the SICL environment. It sets up
*  error handling, opens both an interface and device session, sets timeout
*  values, clears the interface by pulsing IFC, and clears the instrument
*  by performing a Selected Device Clear.
*/

void init_IO ( )
{
      ionerror (I_ERROR_EXIT);   /* set-up interface error handling */

      /* open interface session for verifying SRQ line */
      bus = iopen ( INTERFACE );
      if ( bus == 0 )
         printf ("Bus session invalid\n");

      itimeout ( bus, 20000 );   /* set bus timeout to 20 sec */
      iclear ( bus );            /* clear the interface - pulse IFC */

      scope = iopen ( DEVICE_ADDR );/* open the scope device session */
      if ( scope == 0)
         printf ( "Scope session invalid\n");

      itimeout ( scope, 20000 ); /* set device timeout to 20 sec */
      iclear ( scope );          /* perform Selected Device Clear on scope */

} /* end init_IO */

/*
*  Function name:  write_IO
*  Parameters:  char *buffer which is a pointer to the character string to be
*  output; unsigned long length which is the length of the string to be output
*  Return value:  none
*  Description:  This routine outputs strings to the scope device session
*  using the unformatted I/O SICL commands.
*/

void write_IO ( void *buffer )
   {
      unsigned long actualcnt;
      unsigned long length;
      int send_end = 1;
      length = strlen ( buffer );
      iwrite ( scope, buffer, length, send_end, &actualcnt );

} /* end write_IO */

/*
*  Function name:  write_lrnstr
*  Parameters:  char *buffer which is a pointer to the character string to be
*  output; long length which is the length of the string to be output
*  Return value:  none
*  Description:  This routine outputs a learnstring to the scope device
*  session using the unformatted I/O SICL commands.
*/

void write_lrnstr ( void *buffer, long length )
{
      unsigned long actualcnt;
      int send_end = 1;

      iwrite ( scope, buffer, (unsigned long) length,
          send_end, &actualcnt );
```

```
} /* end write_lrnstr ( ) */



/*
*   Function name:  read_IO
*   Parameters:   char *buffer which is a pointer to the character string to be
*   input; unsigned long length which indicates the max length of the string to be input
*   Return value:   integer which indicates the actual number of bytes read
*   Description:  This routine inputs strings from the scope device session using SICL commands.
*/

int read_IO (void *buffer,unsigned long length)
{
        int reason;
        unsigned long actualcnt;

        iread (scope,buffer,length,&reason,&actualcnt);

        return( (int) actualcnt );
}

/*
*   Function name: check_SRQ
*   Parameters:  none
*   Return value:   integer indicating if bus SRQ line was asserted
*   Description:  This routine checks for the status of SRQ on the bus and returns a value to indicate
the status.
*/

int check_SRQ( )
{
        int srq_asserted;

          /* check for SRQ line status */
          ihpibbusstatus(bus, I_GPIB_BUS_SRQ, &srq_asserted);

        return ( srq_asserted );

} /* end check_SRQ ( ) */



/*
*   Function name:  read_status
*   Parameters:  none
*   Return value:   unsigned char indicating the value of status byte
*   Description:  This routine reads the scope status byte and returns the status.
*/

unsigned char read_status ( )
{
        unsigned char statusbyte;

        /* Always read the status byte from instrument */
        /* NOTE: ireadstb uses serial poll to read status byte - this should clear bit 6 to allow another
     SRQ. */

          ireadstb ( scope, &statusbyte );
        return ( statusbyte );

} /* end read_status ( ) */

/*
*   Function name:  close_IO
*   Parameters:  none
*   Return value:   none
*   Description:  This routine closes device and interface sessions for the
*   SICL environment and calls the routine _siclcleanup which de-allocates
*   resources used by the SICL environment.
*/

void close_IO ( )
{
        iclose ( scope );   /* close device session */
        iclose ( bus );     /* close interface session */
```

```
        _siclcleanup ( );     /* required for 16-bit applications */
} /* end close_SICL ( ) */
```

National I/O Example

File: natl_IO.c

```
/* natl_IO.c */

#include <stdio.h>      /* location of: printf ( ) */
#include <string.h>     /* location of: strlen ( ) */
#include "hpibdecl.h"

/*  This file contains IO and initialization routines for the NI488.2 commands. */
/*
*   Function name:  hpiberr
*   Parameters:  char* - string describing error
*   Return value:  none
*   Description:  This routine outputs error descriptions to an error file.
*/

void hpiberr( char *buffer )
{
      printf ("Error string: %s\n",buffer );

} /* end hpiberr ( ) */

/*
*   Function name:  init_IO
*   Parameters:  none
*   Return value:  none
*   Description:  This routine initializes the NI environment. It sets up error
*   handling, opens both an interface and device session, sets timeout values
*   clears the interface by pulsing IFC, and clears the instrument by performing
*   a Selected Device Clear.
*/

void init_IO ( )
{
      bus = ibfind ( INTERFACE );/* open and initialize GPIB board */
      if ( ibsta & ERR )
         hpiberr ("ibfind error");

      ibconfig ( bus, IbcAUTOPOLL, 0);/* turn off autopolling */

      ibsic ( bus );              /* clear interface - pulse IFC */
      if ( ibsta & ERR )
      {
         hpiberr ( "ibsic error" );
      }

      /* open device session */
      scope = ibdev ( board_index, prim_addr, second_addr, timeout,
                      eoi_mode, eos_mode );
      if ( ibsta & ERR )
      {
         hpiberr ( "ibdev error" );
      }

      ibclr ( scope );       /* clear the device( scope ) */

      if ( ibsta & ERR)
      {
         hpiberr ("ibclr error" );
      }

} /* end init_IO */

/*
*   Function name:  write_IO
*   Parameters:  void *buffer which is a pointer to the character string to be output
*   Return value:  none
*   Description:  This routine outputs strings to the scope device session.
*/
void write_IO ( void *buffer )
{
      long length;

      length = strlen ( buffer );

      ibwrt ( scope, buffer, (long) length );
```

```
        if ( ibsta & ERR )
        {
            hpiberr ( "ibwrt error" );
        }

} /* end write_IO() */


/*
*   Function name:  write_lrnstr
*   Parameters:  void *buffer which is a pointer to the character string to
*   be output; length which is the length of the string to be output
*   Return value:  none
*   Description:  This routine outputs a learnstring to the scope device session.
*/
void write_lrnstr ( void *buffer, long length )
{
        ibwrt ( scope, buffer, (long) length );
        if ( ibsta & ERR )
        {
            hpiberr ( "ibwrt error" );
        }

} /* end write_lrnstr ( ) */

/*
*   Function name:  read_IO
*   Parameters:  char *buffer which is a pointer to the character string to be input;
* unsigned long length which indicates the max length of the string to be input
*   Return value:  integer which indicates the actual number of bytes read
*   Description:  This routine inputs strings from the scope device session.
*/

int read_IO (void *buffer,unsigned long length)
{
    ibrd (scope, buffer, ( long ) length );

    return ( ibcntl );

} /* end read_IO ( ) */


/*
*   Function name:  check_SRQ
*   Parameters:  none
*   Return value:  integer indicating if bus SRQ line was asserted
*   Description:  This routine checks for the status of SRQ on the bus and
*   returns a value to indicate the status.
*/

int check_SRQ ( )
{
        int srq_asserted;
        short control_lines = 0;

        iblines ( bus, &control_lines);

        if ( control_lines & BusSRQ )
           srq_asserted = TRUE;
        else
           srq_asserted = FALSE;

         return ( srq_asserted );

} /* end check_SRQ ( ) */

/*
*   Function name:  read_status
*   Parameters:  none
*   Return value:  unsigned char indicating the value of status byte
*   Description:  This routine reads the scope status byte and returns the status.
*/
unsigned char read_status ( )
{
        unsigned char statusbyte;

        /* Always read the status byte from instrument */
```

```
        ibrsp ( scope,  &statusbyte );

        return ( statusbyte );

} /* end read_status ( ) */


/*
*  Function name:  close_IO
*  Parameters:  none
*  Return value:  none
*  Description:  This routine closes device session.
*/

void close_IO ( )
{
        ibonl ( scope,0 );    /* close device session */

} /* end close_IO ( ) */
```

Multi-Database Example

File: multidatabase.c

```
/*multidatabase.c*/

/*
* This example program demonstrates the use of the Multidatabase functionality of the
* Agilent 86100 DCA.  The program sets up an acquitision of 200 waveforms on two
* channels, first serially, then in parallel.  A mask test and simple
* measurements are made on each channel.  NOTE: the timeout value must
* be set to a higher value (~30s) so that there is enough time to acquire the
* data.
*/

#include <stdio.h>//standard c++ io funcitons
#include <time.h>//time funcitons

//GPIB prototypes (from IO file)
void init_IO ( );
void write_IO ( char* );
int read_IO ( char*, unsigned long );
void close_IO ( );

//prototypes
void initialize();
int acquire_serial();
int acquire_parallel();

void main()
{
    int serialTime, parallelTime; //declarations
    init_IO();      //initial the interface and open GPIB communications
    initialize();                   //set up the instrument
    serialTime = acquire_serial();//acquire the data in serial
    parallelTime = acquire_parallel();//acquire the data in parallel
    close_IO();                     //close GPIB communications

    printf("\nSerial Acquisition Time: %d ms\nParallel Acquisition Time: %d ms\n",
        serialTime, parallelTime);//display acquisition times
    printf("Time Savings: %d ms\n", serialTime-parallelTime);
                                            //display the time savings
}//main()


/*
* Function Name: initialize
* Paramters: none
* Returned value: none
* Description:  This method sets up the channels and acquisition limits of the
* DCA
*/

void initialize()
{
    write_IO("*RST");//reset the DCA
    write_IO("*CLS");//clear the status registers
    write_IO("SYSTem:MODE EYE");//switch to Eye/mask mode

    write_IO("STOP");//stop acquistion
    write_IO("CDISplay");//clear the display

    write_IO("ACQuire:RUNTil WAVeforms,200");
                    //set the acquistion limit to 200 waveforms

    write_IO("CHANnel1:FSELect 1");//choose filter #1 on channel 1
    write_IO("CHANnel1:FILTer ON");//turn on the filter

    write_IO("CHANnel3:FSELect 1");//choose filter #1 on channel 3
    write_IO("CHANnel3:FILTer ON");//turn on the filter

}//initialize()

/*
* Funciton Name: acquireSerial
* Parameters: none
* Returned value: int - the time to acquire the data
* Description: This routine turns on channel 1, performs an autoscale, acquires
```

```c
* 200 waveforms, performs a mask test, and then performs the measurements.  The
* process is then repeated for channel 2.
*/

int acquire_serial()
{
    printf("Serial Acquisition in progress\n");//status report

//decalrations
    int start=clock(),stop;
    char Msk_hits1[16],Crss_pct1[16],Ext_rat1[16],buff[32];
    char Msk_hits2[16],Crss_pct2[16],Ext_rat2[16];

    write_IO("CHANnel1:DISPlay ON");//turn on channel one
    write_IO("RUN");            //start acquistion
    write_IO("AUToscale");//Autoscale
    write_IO("*OPC?");          //query for completion
    read_IO(buff,5);            //read completion response

    write_IO("MTESt:LOAD \"STM016_OC48.msk\"");//load OC-48 mask
    write_IO("MTESt:START");//start mask test
    write_IO("MTESt:COUNt:FSAMples?");//query the number of failed samples
    Msk_hits1[read_IO(Msk_hits1, 15)]=0;//get the number of mask hits
    write_IO("MTESt:TEST OFF");//trun off the maks test

    write_IO("MEASure:CGRade:CROSsing?");//query the crossing percentage
    Crss_pct1[read_IO(Crss_pct1,15)]=0;//get the crossing percentage

    write_IO("MEASure:CGRade:ERATio? DECibel");//query the extinction ratio
    Ext_rat1[read_IO(Ext_rat1,15)]=0;//get the extinction ratio

    write_IO("CHANnel3:DISPlay ON");//turn on channel three
    write_IO("RUN");            //start acquistion
    write_IO("AUToscale");//Autoscale
    write_IO("*OPC?");          //query for completion
    read_IO(buff,5);            //read completion response

    write_IO("MTESt:TEST ON");//start mask test
    write_IO("MTESt:COUNt:FSAMples?");//query the number of failed samples
    Msk_hits2[read_IO(Msk_hits2, 15)]=0;//get the number of mask hits

    write_IO("MEASure:CGRade:CROSsing?");//query the crossing percentage
    Crss_pct2[read_IO(Crss_pct2,15)]=0;//get the crossing percentage

    write_IO("MEASure:CGRade:ERATio? DECibel");//query the extinction ratio
    Ext_rat2[read_IO(Ext_rat2,15)]=0;//get the extinction ratio

    stop = clock();

    //display the results
    printf("Channel 1:\n Mask hits:%s Crossing %%:%s Extinction Ratio:%s\n",
                                        Msk_hits1,Crss_pct1,Ext_rat1);
    printf("Channel 3:\n Mask hits:%s Crossing %%:%s Extinction Ratio:%s\n",
                                        Msk_hits2,Crss_pct2,Ext_rat2);

    return (stop-start);
}//acquireSerial()

/*
* Funciton Name: acquireParallel
* Parameters: none
* Returned value: int - the time to acquire the data
* Description: This routine is identical to acquireSerial, except that the data
* is aquired at the same time.
*/

int acquire_parallel()
{
    printf("Parallel Acquisition In progress\n");//status report

//decalrations
    int start=clock(),stop;
    char Msk_hits1[16],Crss_pct1[16],Ext_rat1[16],buff[32];
    char Msk_hits2[16],Crss_pct2[16],Ext_rat2[16];

    write_IO("CHANnel1:DISPlay ON");//turn on channel one
    write_IO("CHANnel3:DISPlay ON, APPEnd");//turn on channel three
    write_IO("RUN");                    //start acquistion
```

```
        write_IO("AUToscale");          //Autoscale
        write_IO("CALibrate:SKEW:AUTO");//auto deskew the two channels
        write_IO("*OPC?");                      //query for completion
        read_IO(buff,5);                        //read completion response

        write_IO("MTESt:LOAD \"STM016_OC48.msk\"");//load OC-48 mask
        write_IO("MTESt:SOURce CHANnel1");//set mask test channel1
        write_IO("MTESt:START");//start mask test
        write_IO("MTESt:COUNt:FSAMples?");//query the number of failed samples
        Msk_hits1[read_IO(Msk_hits1, 15)]=0;//get the number of mask hits

        write_IO("MTESt:SOURce CHANnel3");//mask test channel3
        write_IO("MTESt:TEST ON");//start mask test
        write_IO("MTESt:COUNt:FSAMples?");//query the number of failed samples
        Msk_hits2[read_IO(Msk_hits2, 15)]=0;//get the number of mask hits

        write_IO("MEASure:CGRade:SOURce CHANnel1"); //measure Channel 1
        write_IO("MEASure:CGRade:CROSsing?");//query the crossing percentage
        Crss_pct1[read_IO(Crss_pct1,15)]=0;//get the crossing percentage

        write_IO("MEASure:CGRade:ERATio? DECibel");//query the extinction ratio
        Ext_rat1[read_IO(Ext_rat1,15)]=0;//get the extinction ratio


        write_IO("MEASure:CGRade:SOURce CHANnel3"); //measure Channel 1
        write_IO("MEASure:CGRade:CROSsing?");//query the crossing percentage
        Crss_pct2[read_IO(Crss_pct2,15)]=0;//get the crossing percentage

        write_IO("MEASure:CGRade:ERATio? DECibel");//query the extinction ratio
        Ext_rat2[read_IO(Ext_rat2,15)]=0;//get the extinction ratio

        stop = clock();

        //display the results
        printf("Channel 1:\n Mask hits:%s Crossing %%:%s Extinction Ratio:%s\n",
                                                Msk_hits1,Crss_pct1,Ext_rat1);
        printf("Channel 3:\n Mask hits:%s Crossing %%:%s Extinction Ratio:%s\n",
                                                Msk_hits2,Crss_pct2,Ext_rat2);

        return (stop-start);  //return the total run time


        return 1;
}//acquireParallel()
```

GPIB Header File

File: gpibdecl.c

```
/* gpibdecl.h */

/*
*  This file includes necessary prototypes and declarations for
*  the example programs for the Agilent 86100*/
*/

/*
*  User must indicate which GPIB card (Agilent or National) is being used.
*  Also, if using a National card, indicate which version of windows
*  (WIN31 or WIN95) is being used.
*/


#define Agilent          /* Uncomment if using Agilent interface card */
/* #define NATL */

/* #define WIN31 */      /* For National card ONLY - select windows version */
#define WIN95

#ifdef Agilent
#include <sicl.h>
#else
        #ifdef WIN95
        #include <windows.h>/* include file for Windows 95 */
        #include <decl-32.h>
        #else
        #include <windecl.h>/* include file for Windows 3.1 */
        #endif
#endif

#define CME 32
#define EXE 16
#define DDE  8
#define QYE  4

#define SRQ_BIT 64
#define MAX_LRNSTR 14000
#define MAX_LENGTH 4096
#define MAX_INT 4192


#ifdef Agilent
#define DEVICE_ADDR "hpib7,7"
#define INTERFACE "hpib7"
#else
#define INTERFACE "hpib0"

#define board_index 0
#define prim_addr 7
#define second_addr 0
#define timeout 13
#define eoi_mode   1
#define eos_mode 0
#endif


#define TRUE 1
#define FALSE 0


/* GLOBALS */
#ifdef Agilent
        INST bus;
        INST scope;
#else
        int bus;
        int scope;
#endif

/* GPIB prototypes */
void init_IO ( );
```

```
void write_IO ( void* );
void write_lrnstr ( void*, long );
int read_IO ( void*, unsigned long );
int check_SRQ ( );
unsigned char read_status ( );
void close_IO ( );
void hpiberr ( );


void srq_handler ( );
```

BASIC Programming Examples

Listings of the BASIC sample programs in this section include:

General Measurement Example

File: init.bas

```
10   !file: init
20   !
30   !
40   ! This program demonstrates the order of commands suggested for operation of
50   ! the Agilent 86100 analyzer via GPIB.  This program initializes the scope, acquires
60   ! data, performs automatic measurements, and transfers and stores the data on the
70   ! PC as time/voltage pairs in a comma-separated file format useful for spreadsheet
80   ! applications.  It assumes an interface card at interface select code 7, an
90   ! Agilent 86100 scope at address 7, and the Agilent 86100 cal signal connected to Channel 1.
100    !
110    !
120    !
130    COM /Io/@Scope,@Path,Interface
140    COM /Raw_data/ INTEGER Data(4095)
150    COM /Converted_data/ REAL Time(4095),Volts(4095)
160    COM /Variables/ REAL Xinc,Xref,Xorg,Yinc,Yref,Yorg
170    COM /Variables/ INTEGER Record_length
180    !
190    !
200    CALL Initialize
210    CALL Acquire_data
220    CALL Auto_msmts
230    CALL Transfer_data
240    CALL Convert_data
250    CALL Store_csv
260    CALL Close
270    END
280    !
290    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
300    !
310    !
320    !                             BEGIN SUBPROGRAMS
330    !
340    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
350    !
360    !
370    !     Subprogram name:  Initialize
380    !     Parameters: none
390    !     Return value: none
400    !     Description:  This routine initializes the interface and the scope.   The instrument
410    !      is reset to a known state and the interface is cleared.  System headers
420    !     are turned off to allow faster throughput and immediate access to the
430    !      data values requested by the queries. The analyzer time base,
440 !    channel, and trigger subsystems are then configured. Finally, the
450 !    acquisition subsystem is initialized.
460 !
470 !
480    SUB Initialize
490    COM /Io/@Scope,@Path,Interface
500    COM /Variables/ REAL Xinc,Xref,Xorg,Yinc,Yref,Yorg
510    COM /Variables/ INTEGER Record_length
520       Interface=7
530       ASSIGN @Scope TO 707
540       RESET Interface
550       CLEAR @Scope
560       OUTPUT @Scope;"*RST"
570       OUTPUT @Scope;"*CLS"
580       OUTPUT @Scope;":SYSTem:HEADer OFF"
590       !Initialize Timebase: center reference, 2 ms full-scale (200 us/div), 20 us delay
600       OUTPUT @Scope;":TIMebase:REFerence CENTer;RANGe 2e-3;POSition 20e-6"
```

```
610             ! Initialize Channel1:  1.6V full-scale (200mv/div), -415mv offset
620         OUTPUT @Scope;":CHANnel1:RANGe 1.6;OFFSet -415e-3"
630             !Initialize Trigger: Edge trigger, channel source at -415mv
640         OUTPUT @Scope;":TRIGger:SOURce FPANel;SLOPe POSitive"
650         OUTPUT @Scope;":TRIGger:LEVel-0.415"
660             ! Initialize acquisition subsystem
665             ! Real time acquisition, Averaging off, memory depth 4096
670         OUTPUT @Scope;":ACQuire:AVERage OFF;POINts 4096"
680         Record_length=4096
690     SUBEND
700     !
710     !
720     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
730     !
740     !
750     !     Subprogram name:  Acquire_data
760     !     Parameters: none
770     !     Return value: none
780     !     Description:  This routine acquires data according to the current instrument
790      !                   setting.  It uses the root level :DIGitize command.  This command
800     !                   is recommended for acquisition of new data because it will initialize
810      !                   the data buffers, acquire new data, and ensure that acquisition
820     !                   criteria are met before acquisition of data is stopped.  The captured
830     !                   data is then available for measurements, storage, or transfer to a
840     !                   PC.  Note that the display is automatically turned off by the :DIGitize
850     !                   command and must be turned on to view the captured data.
860     !
870     !
880     SUB Acquire_data
890     COM /Io/@Scope,@Path,Interface
900     OUTPUT @Scope;":DIGitize CHANnel1"
910     OUTPUT @Scope;":CHANnel1:DISPlay ON"
920     SUBEND
930     !
940     !
950     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
960     !
970     !
980     !     Subprogram name:  Auto_msmts
990     !     Parameters: none
1000    !     Return value: none
1010    !     Description:  This routine performs automatic measurements of volts peak-to-peak
1020    !                   and frequency on the acquired data.  It also demonstrates two methods
1030    !                   of error detection when using automatic measurements.
1040    !
1050    !
1060    SUB Auto_msmts
1070    COM /Io/@Scope,@Path,Interface
1080    REAL Period,Vpp
1090    DIM Vpp_str$[64]
1100    DIM Period_str$[64]
1110    Bytes_read=0
1120    !
1130    !     Error checking on automatic measurements can be done using one of two methods.
1140    !     The first method requires that you turn on results in the Measurement subsystem
1150    !     using the command ":MEASure:SEND ON".  When this is on, the scope will return the
1160    !     measurement and a result indicator.  The result flag is zero if the measurement
1170    !     was successfully completed, otherwise a non-zero value is returned which indicates
1180    !     why the measurement failed.  See the Programmer's Manual for descriptions of result
1190    !     indicators.  The second method simply requires that you check the return value of
1200    !     the measurement.  Any measurement not made successfully will return with the value
1210    !     +9.999e37.  This could indicate that either the measurement was unable to be
1220    !     performed or that insufficient waveform data was available to make the measurement.
1230    !
1240    !     METHOD ONE
1250    !
1260        OUTPUT @Scope;":MEASure:SEND ON"            !turn on results
1270         OUTPUT @Scope;":MEASure:VPP? CHANnel1"       !Query volts peak-to-peak
1280        ENTER @Scope;Vpp_str$
1290        Bytes_read=LEN(Vpp_str$)                     !Find length of string
1300        CLEAR SCREEN
1310        IF Vpp_str$[Bytes_read;1]="0" THEN           !Check result value
1320            PRINT
1330            PRINT "VPP is ";VAL(Vpp_str$[1,Bytes_read-1])
1340            PRINT
1350        ELSE
1360            PRINT
1370            PRINT "Automated vpp measurement error with result ";Vpp_str$[Bytes_read;1]
```

```
1380            PRINT
1390         END IF
1400    !
1410    !
1420         OUTPUT @Scope;":MEASure:PERiod? CHANnel1"  !Query frequency
1430         ENTER @Scope;Period_str$
1440         Bytes_read=LEN(Period_str$)                      !Find string length
1450         IF Period_str$[Bytes_read;1]="0" THEN         !Determine result value
1460            PRINT
1470            PRINT "Period is ";VAL(Period_str$[1,Bytes_read-1])
1480            PRINT
1490         ELSE
1500            PRINT
1510            PRINT "Automated period measurement error with result ";Period_str$[Bytes_read;1]
1520            PRINT
1530         END IF
1540    !
1550    !
1560    !     METHOD TWO
1570    !
1580         OUTPUT @Scope;":MEASure:SEND OFF"         !turn off results
1590         OUTPUT @Scope;":MEASure:VPP? CHANnel1"       !Query volts peak-to-peak
1600         ENTER @Scope;Vpp
1610         IF Vpp<9.99E+37 THEN
1620            PRINT
1630            PRINT "VPP is ";Vpp
1640            PRINT
1650         ELSE
1660            PRINT
1670            PRINT "Automated vpp measurement error ";Vpp
1680            PRINT
1690         END IF
1700         OUTPUT @Scope;":MEASure:PERiod? CHANnel1"
1710         ENTER @Scope;Period
1720         IF Freq<9.99E+37 THEN
1730            PRINT
1740            PRINT "Period is ";Period
1750            PRINT
1760         ELSE
1770            PRINT
1780            PRINT "Automated period measurement error";Period
1790            PRINT
1800         END IF
1810    SUBEND
1820    !
1830    !
1840    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1850    !
1860    !
1870    !     Subprogram name:  Transfer_data
1880    !     Parameters: none
1890    !     Return value: none
1900    !     Description:  This routine transfers the waveform data and conversion factors to
1910    !               to PC.
1920    !
1930    !
1940    SUB Transfer_data
1950    COM /Io/@Scope,@Path,Interface
1960    COM /Raw_data/ INTEGER Data(4095)
1970    COM /Converted_data/ REAL Time(4095),Volts(4095)
1980    COM /Variables/ REAL Xinc,Xref,Xorg,Yinc,Yref,Yorg
1990    COM /Variables/ INTEGER Record_length
2000    !                                   define waveform data source and format
2010    OUTPUT @Scope;":WAVeform:SOURce CHANnel1"
2020    OUTPUT @Scope;":WAVeform:FORMat WORD"
2030    !                                   request values needed to convert raw data to real
2040    OUTPUT @Scope;":WAVeform:XINCrement?"
2050    ENTER @Scope;Xinc
2060    OUTPUT @Scope;":WAVeform:XORigin?"
2070    ENTER @Scope;Xorg
2080    OUTPUT @Scope;":WAVeform:XREFerence?"
2090    ENTER @Scope;Xref
2100    OUTPUT @Scope;":WAVeform:YINCrement?"
2110    ENTER @Scope;Yinc
2120    OUTPUT @Scope;":WAVeform:YORigin?"
2130    ENTER @Scope;Yorg
2140    OUTPUT @Scope;":WAVeform:YREFerence?"
2150    ENTER @Scope;Yref
```

```
2160    !
2170    !                                           request data
2180    OUTPUT @Scope;":WAVeform:DATA?"
2190    ENTER @Scope USING "#,1A";First_chr$        !ignore leading #
2200    ENTER @Scope USING "#,1D";Header_length     !input number of bytes in header value
2210    ENTER @Scope USING "#,"&VAL$(Header_length)&"D";Record_length      !Record length in bytes
2220    Record_length=Record_length/2               !Record length in words
2230    ENTER @Scope USING "#,W";Data(*)
2240    ENTER @Scope USING "#,A";Term$              !Enter terminating character
2250    !
2260    SUBEND
2270    !
2280    !
2290    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2300    !
2310    !
2320    !       Subprogram name:  Convert_data
2330    !       Parameters: none
2340    !       Return value: none
2350    !       Description:  This routine converts the waveform data to time/voltage information
2360    !                     using the values Xinc, Xref, Xorg, Yinc, Yref, and Yorg used to describe
2370    !                     the raw waveform data.
2380    !
2390    !
2400    SUB Convert_data
2410    COM /Io/@Scope,@Path,Interface
2420    COM /Raw_data/ INTEGER Data(4095)
2430    COM /Converted_data/ REAL Time(4095),Volts(4095)
2440    COM /Variables/ REAL Xinc,Xref,Xorg,Yinc,Yref,Yorg
2450    COM /Variables/ INTEGER Record_length
2460    !
2470    FOR I=0 TO Record_length-1
2480        Time(I)=(((I)-Xref)*Xinc)+Xorg
2490        Volts(I)=((Data(I)-Yref)*Yinc)+Yorg
2500    NEXT I
2510    SUBEND
2520    !
2530    !
2540    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2550    !
2560    !
2570    !       Subprogram name: Store_csv
2580    !       Parameters: none
2590    !       Return value: none
2600    !       Description:  This routine stores the time and voltage information about the waveform
2610    ! as time/voltage pairs in a comma-separated variable file format.
2620    !
2630    !
2640    SUB Store_csv
2650    COM /Io/@Scope,@Path,Interface
2660    COM /Converted_data/ REAL Time(4095),Volts(4095)
2670    COM /Variables/ REAL Xinc,Xref,Xorg,Yinc,Yref,Yorg
2680    COM /Variables/ INTEGER Record_length
2690        !Create a file to store pairs in
2700    ON ERROR GOTO Cont
2710    PURGE "Pairs.csv"
2720 Cont: OFF ERROR
2730    CREATE "Pairs.csv",Max_length
2740    ASSIGN @Path TO "Pairs.csv";FORMAT ON
2750                                !Output data to file
2760    FOR I=0 TO Record_length-1
2770       OUTPUT @Path;Time(I),Volts(I)
2780    NEXT I
2790    SUBEND
2800    !
2810    !
2820    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
2830    !
2840    !
2850    !       Subprogram name: Close
2860    !       Parameters: none
2870    !       Return value: none
2880    !       Description: This routine closes the IO paths.
2890    !
2900    !
2910    SUB Close
2920    COM /Io/@Scope,@Path,Interface
2930    !
```

```
2940    RESET Interface
2950    ASSIGN @Path TO *
2960    SUBEND
```

Service Request Example

File: srq.bas

```
10    !File: srq.bas
20    !
30    !    This program demonstrates how to set up and check Service Requests from
40    !    the scope.  It assumes an interface select code of 7 with a scope at
50    !    address 7.  It also assumes a signal is connected to the scope.
60    !
70    !
80    COM /Io/@Scope,Interface
90    COM /Variables/Temp
100   CALL Initialize
110   CALL Setup_srq
120       ON INTR Interface CALL Srq_handler      !Set up routine to handle interrupt
130       ENABLE INTR Interface;2                 !Enable SRQ Interrupt for Interface
140   CALL Create_srq
150   CALL Close
160   END
170   !
180   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
190   !
200   !                          BEGIN SUBPROGRAMS
210   !
220   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
230   !
240   !
250   !     Subprogram name:  Initialize
260   !     Parameters: none
270   !     Return value: none
280   !     Description:   This routine initializes the interface and the scope.
290   !               The instrument is reset to a known state and the interface is
300   !                  cleared.  System headers are turned off to allow faster throughput
310   !                  and immediate access to the data values requested by the queries.
320   !
330   !
340   SUB Initialize
350   COM /Io/@Scope,Interface
360       ASSIGN @Scope TO 707
370       Interface=7
380       RESET Interface
390       CLEAR @Scope
400       OUTPUT @Scope;"*RST"
410       OUTPUT @Scope;"*CLS"
420       OUTPUT @Scope;":SYSTem:HEADer OFF"
430       OUTPUT @Scope;":AUToscale"
440   SUBEND
450   !
460   !
470   !
480   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
490   !
500   !     Subprogram name: Setup_srq
510   !     Parameters: none
520   !     Return value: none
530   !     Description:  This routine sets up the scope to generate Service Requests.
540   !                   It sets the Service Request Enable Register Event Status Bit
550   !                   and the Standard Event Status Enable REgister to allow SRQs on
560   !               Command or Query errors.
570   !
580   !
590   SUB Setup_srq
600   COM /Io/@Scope,Interface
610       OUTPUT @Scope;"*SRE 32"    !Enable Service Request Enable Registers – Event Status bit
620   !
630   !     Enable Standard Event Status Enable Register:
640   !        enable  bit 4 – Command Error –  value 32
650   !                bit 1 – Query Error – value 4
660       OUTPUT @Scope;"*ESE 36"
670   SUBEND
680   !
690   !
700   !
710   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
720   !
730   !
```

```
740   !       Subprogram name:  Create_srq
750   !       Parameters: none
760   !       Return value: none
770   !       Description:  This routine will send an illegal command to the scope to
780   !                     show how to detect and handle an SRQ.  A query is sent to
790   !                     the scope which is then followed by another command causing
800   !                     a query interrupt error.  An illegal command header is then
810   !                     sent to demonstrate how to handle multiple errors in the error queue.
820   !
830   !
840   !
850   SUB Create_srq
860   COM /Io/@Scope,Interface
870       DIM Buf$[256]
880       OUTPUT @Scope;":CHANnel2:DISPlay?"
890       OUTPUT @Scope;":CHANnel2:DISPlay OFF"      !send query interrupt
900       OUTPUT @Scope;":CHANnel:DISPlay OFF"       !send illegal header
910                                   ! Do some stuff to allow time for SRQ to be recognized
920                                   !
930       OUTPUT @Scope;"*IDN?"    !Request IDN to verify communication
940        ENTER @Scope;Buf$        !NOTE: There is a leading zero to this query response
950       PRINT                    !which represents the response to the interrupted query above
960       PRINT Buf$
970       PRINT
980   SUBEND
990   !
1000  !
1010  !
1020  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1030  !
1040  !
1050  !       Subprogram name:  Srq_handler
1060  !       Parameters: none
1070  !       Return value: none
1080  !       Description:  This routine verifies the status of the SRQ line.  It then checks
1090  !                     the status byte of the scope to determine if the scope caused the
1100  !                     SRQ.  Note that using a SPOLL to read the status byte of the scope
1110  !                     clears the SRQ and allows another to be generated.  The error queue
1120  !                     is read until all errors have been cleared.  All event registers and
1130  !                     queues, except the output queue, are cleared before control is returned
1140  !                     to the main program.
1150  !
1160  !
1170  !
1180  SUB Srq_handler
1190      COM /Io/@Scope,Interface
1200      DIM Error_str$[64]
1210      INTEGER Srq_asserted,More_errors
1220    Status_byte=SPOLL(@Scope)
1230    IF BIT(Status_byte,6) THEN
1240        More_errors=1
1250        WHILE More_errors
1260            OUTPUT @Scope;":SYSTem:ERROR? STRING"
1270            ENTER @Scope;Error_str$
1280            PRINT
1290            PRINT Error_str$
1300            IF Error_str$[1,1]="0" THEN
1310                OUTPUT @Scope;"*CLS"
1320                More_errors=0
1330            END IF
1340        END WHILE
1350      ELSE
1360          PRINT
1370          PRINT "Scope did not cause SRQ"
1380          PRINT
1390      END IF
1400      ENABLE INTR Interface;2     !re-enable SRQ
1410  SUBEND
1420  !
1430  !
1440  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1450  !
1460  !       Subprogram name:  Close
1470  !       Parameters: none
1480  !       Return value: none
1490  !       Description:  This routine resets the interface.
1500  !
1510  !
```

```
1520  !
1530  SUB Close
1540  COM /Io/@Scope,Interface
1550
1560  RESET Interface
1570  SUBEND
1580  !
1590  !
1600  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Learn String Example

File: lrn_str.bas

```
10    !FILE: lrn_str.bas
20    !
30    !THIS PROGRAM WILL INITIALIZE THE SCOPE, AUTOSCALE, AND DIGITIZE THE WAVEFORM
40    !INFORMATION.  IT WILL THEN QUERY THE INSTRUMENT FOR THE LEARNSTRING AND WILL
50    !SAVE THE INFORMATION TO A FILE.  THE PROGRAM WILL THEN PROMPT YOU TO CHANGE
60    !THE SETUP THEN RESTORE THE ORIGINAL LEARNSTRING CONFIGURATION. IT ASSUMES
70    !AN Agilent 86100 at ADDRESS 7, GPIB INTERFACE at 7, AND THE CAL SIGNAL ATTACHED TO
80    !CHANNEL 1.
90    !
100   !
110   COM /Io/@Scope,@Path,Interface
120   COM /Variables/Max_length
130   CALL Initialize
140   CALL Store_lrnstr
150   CALL Change_setup
160   CALL Get_lrnstr
170   CALL Close
180   END
1200   !
210   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
220   !
230   !                    BEGIN SUBROUTINES
240   !
250   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
260   !    Subprogram name:  Initialize
270   !    Parameters: none
280   !    Return value: none
290   !    Description:  This routine initializes the path descriptions and resets the
300   !                  interface and the scope.  It performs an autoscale on the signal,
310   !                  acquires the data on channel 1, and turns on the display.
320   !                  NOTE:  This routine also turns on system headers.  This allows the
330   !                  string ":SYSTEM:SETUP " to be returned with the learnstring so the
340   !                  return string is in the proper format.
350   !
360   SUB Initialize
370       COM /Io/@Scope,@Path,Interface
380       COM /Variables/Max_length
390       Max_length=14000
400       ASSIGN @Scope TO 707
410       Interface=7
420       RESET Interface
430       CLEAR @Scope
440       OUTPUT @Scope;"*RST"
450       OUTPUT @Scope;"*CLS"
460       OUTPUT @Scope;":SYSTem:HEADer ON"
470       OUTPUT @Scope;":AUToscale"
480   SUBEND
500   !
510   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
520   !
540   !     Subprogram name: Store_lrnstr
550   !     Parameters: none
560   !     Return value: none
570   !     Description:  This routine creates a file in which to store the learnstring
580   !                   configuration (Filename:Lrn_strg).  It requests the learnstring
590   !                   and inputs the configuration to the PC.  Finally, it stores the
600   !                   configuration to the file.
610   !
620   SUB Store_lrnstr
630       COM /Io/@Scope,@Path,Interface
640       COM /Variables/Max_length
650       ON ERROR GOTO Cont
660       PURGE "Lrn_strg"
670 Cont:  OFF ERROR
680       CREATE BDAT "Lrn_strg",1,14000
690       DIM Setup$[14000]
700       ASSIGN @Path TO "Lrn_strg"
710       OUTPUT @Scope;":SYSTem:SETup?"
720       ENTER @Scope USING "-K";Setup$
730       OUTPUT @Path,1;Setup$
740       CLEAR SCREEN
750       PRINT "Learn string stored in file: Lrn_strg"
760   SUBEND
```

```
770    !
790    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
800    !
810    !      Subprogram name: Change_setup
820    !      Parameters: none
830    !      Return value: none
840    !      Description:  This subprogram requests that the user change the
850    !                    scope setup, then press a key to continue.
860    !
870    !
880    SUB Change_setup
890        COM /Io/@Scope,@Path,Interface
900
910        PRINT
920        PRINT "Please adjust setup and press Continue to resume."
930        PAUSE
940    SUBEND
950    !
970    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
980    !
990    !      Subprogram name: Get_lrnstr
1000   !      Parameters: none
1010   !      Return value: none
1020   !      Description:  This subprogram loads a learnstring from the
1030   !                    file "Lrn_strg" to the scope.
1050   !
1060   SUB Get_lrnstr
1070       COM /Io/@Scope,@Path,Interface
1080       COM /Variables/Max_length
1090       DIM Setup$[14000]
1100       ENTER @Path,1;Setup$
1110       OUTPUT @Scope USING "#,-K";Setup$
1120       OUTPUT @Scope;":RUN"
1130   SUBEND
1150   !
1160   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1180   !
1190   !      Subprogram name: Close
1200   !      Parameters: none
1210   !      Return value: none
1220   !      Description:  This routine resets the interface, and closes all I/O paths.
1240   !
1250   !
1260   SUB Close
1270   COM /Io/@Scope,@Path,Interface
1280
1290   RESET Interface
1300   ASSIGN @Path TO *
1310   SUBEND
1320   !
1330   !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

3 Common Commands

Common commands are defined by the IEEE 488.2 standard. They control generic device functions that are common to many different types of instruments. Common commands can be received and processed by the analyzer, whether they are sent over the GPIB as separate program messages or within other program messages.

KEYSIGHT
TECHNOLOGIES

Introduction

Receiving Common Commands

Common commands can be received and processed by the analyzer, whether they are sent over the GPIB as separate program messages or within other program messages. If a subsystem is currently selected and a common command is received by the analyzer, the analyzer remains in the selected subsystem. For example, if the program message
`"ACQUIRE:AVERAGE ON;*CLS;COUNT 1024"`

is received by the analyzer, the analyzer enables averaging, clears the status information, then sets the number of averages without leaving the selected subsystem.

Status Registers

The following two status registers used by common commands have an enable (mask) register. By setting bits in the enable register, the status information can be selected for use. Refer to "Status Reporting" on page 26 for a complete discussion of status.

Table 14   Status Registers

Status Register	Enable Register
Event Status Register	Event Status Enable Register
Status Byte Register	Service Request Enable Register

Command Synchronization

Three commands are available for the synchronization between remote command scripts and the instrument: *OPC (command and query) and *WAI. The *OPC command sets a bit in the Standard Event Status Register when all pending device operations have finished. It is useful to verify the completion of commands that could take a variable amount of time or commands executed in parallel with other commands, such as PRINt, and the limit test commands (ACQuire:RUNtil, MTEST:RUNtil, and LTEST). It does not stop the execution of the remote script. The *OPC query allows synchronization between the computer and the instrument by using the message available (MAV) bit in the Status Byte, or by reading the output queue. Unlike the *OPC command, the *OPC query does not affect the OPC event bit in the Standard Event Status Register. The execution of the remote script is halted and therefore the *OPC query should be used judiciously. For example, the command ":MTEST:RUNtil FSAMPLES,100'; *OPC?" will lock the remote interface until 100 failed samples are detected, which could take a very long time. Under these circumstances, the user must send a device clear or power down to re-start the instrument. The *WAI command is similar to the *OPC query as it will also block the execution of the remote script until all pending operations are finished. It is particularly useful if the host computer is connected to two or more instruments. This command will not block the GPIB bus, allowing the computer to continue issuing commands to the instrument not executing the *WAI command.

Commands

*CLS (Clear Status)

Command   `*CLS`

Clears all status and error registers. Refer to "Error Messages" on page 41 for a complete discussion of status.

Example   `10 OUTPUT 707;"*CLS"`

*ESE (Event Status Enable)

Command   `*ESE <mask>`

Sets the Standard Event Status Enable Register bits. <mask> is an integer, 0 to 255, representing a mask value for the bits to be enabled in the Standard Event Status Register as shown in Table 15 on page 92.

Example   This example enables the User Request (URQ) bit of the Standard Event Status Enable Register. When this bit is enabled and a front-panel key is pressed, the Event Summary bit (ESB) in the Status Byte Register is also set.
`10 OUTPUT 707;"*ESE 64"`

Query   `*ESE?`

Returns the current contents of the Standard Event Status Enable Register.

Returned Format   `<mask><NL>`

<mask> is an integer, +0 to +255 (the plus sign is also returned), representing a mask value for the bits enabled in the Standard Event Status Register as shown in Table 15 on page 92.

Example   This example places the current contents of the Standard Event Status Enable Register in the numeric variable, Event.
`10 OUTPUT 707;"*ESE?"`
`20 ENTER 707;Event`

The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register. A "1" in the Standard Event Status Enable Register enables the corresponding bit in the Standard Event Status Register. A "0" in the enable register disables the corresponding bit.

Table 15   Standard Event Status Enable Register Bits

Bit	Weight	Enables	Definition
7	128	PON - Power On	Indicates power is turned on.
6	64	URQ - User Request	Not used. Permanently set to zero.
5	32	CME - Command Error	Indicates whether the parser detected an error.
4	16	EXE - Execution Error	Indicates whether a parameter was out-of-range, or was inconsistent with the current settings.
3	8	DDE - Device Dependent Error	Indicates whether the device was unable to complete an operation for device-dependent reasons.
2	4	QYE - Query Error	Indicates if the protocol for queries has been violated.
1	2	RQC - Request Control	Indicates whether the device is requesting control.
0	1	OPC - Operation Complete	Indicates whether the device has completed all pending operations.

See Also   Refer to "Status Reporting" on page 26 for a complete discussion of status.

*ESR? (Event Status Register)

Query   `*ESR?`

Returns the contents of the Standard Event Status Register. Reading this register clears the Standard Event Status Register, as does *CLS.

Returned Format   `<status><NL>`

`<status>` is an integer, 0 to 255, representing the total bit weights of all bits that are high at the time you read the register.

Example
```
10 OUTPUT 707;"*ESR?"
20 ENTER 707;Event
```

Table 16 lists each bit in the Event Status Register and the corresponding bit weights.

Table 16   Standard Event Status Register Bits

Bit	Bit Weight	Bit Name	Condition
7	128	PON	1 = OFF to ON transition has occurred.
6	64		Not Used. Permanently set to zero.
5	32	CME	0 = no command errors. 1 = a command error has been detected.
4	16	EXE	0 = no execution error. 1 = an execution error has been detected.
3	8	DDE	0 = no device-dependent errors. 1 = a device-dependent error has been detected.
2	4	QYE	0 = no query errors. 1 = a query error has been detected.
1	2	RQC	0 = request control - NOT used - always 0.

Table 16   Standard Event Status Register Bits (continued)

0	1	OPC	0 = operation is not complete. 1 = operation is complete.
	0 = False = Low		1 = True = High

*IDN? (Identification Number)

Query   `*IDN?`

Returns the company name, model number, serial number, and software version by returning the following string. Notice that AGILENT is returned instead of KEYSIGHT.
`AGILENT TECHNOLOGIES,86100C,<USXXXXXXXX>,<Rev #>`

Specifies the serial number, <USXXXXXXXX>, of the analyzer. The first two letters and digits of the serial prefix are the country of manufacture for the analyzer. The last five digits are the serial suffix, which is assigned sequentially, and is different for each analyzer.

<Rev #> specifies the software version of the analyzer, and is the revision number.

Returned Format   `AGILENT TECHNOLOGIES,86100C,USXXXXXXXX,A.XX.XX<NL>`

Example   This example places the analyzer's identification information in the string variable, Identify$.
```
10 DIM Identify$[50]!Dimension variable
20 OUTPUT 707;"*IDN?"
30 ENTER 707;Identify$
```

*LRN? (Learn)

Query   `*LRN?`

Returns a string that contains the analyzer's current setup. The analyzer's setup can be stored and sent back to the analyzer at a later time. This setup string should be sent to the analyzer just as it is. It works because of its embedded :SYStem:SETup header. The *LRN query always returns :SYSTem:SETup as a prefix to the setup block. The SYSTem:HEADer command has no effect on this response.

Returned Format   `:SYSTem:SETup <setup><NL>`

This is a definite length arbitrary block response specifying the current analyzer setup. The block size is subject to change with different firmware revisions.

Example   This example sets the scope's address and asks for the learn string, then determines the string length according to the IEEE 488.2 block specification. It then reads the string and the last EOF character.
```
10 ! Set up the scope's address and
20 ! ask for the learn string...
30 ASSIGN @Scope TO 707
40 OUTPUT @Scope:"*LRN?"
50 !
60 ! Search for the # sign.
70 !
80 Find_pound_sign: !
90 ENTER @Scope USING "#,A";Thischar$
100 IF Thischar$<>"#" THEN Find_pound_sign
110 !
120 ! Determine the string length according
130 ! to the IEEE 488.2 # block spec.
140 ! Read the string then the last EOF char.
150 !
160 ENTER @Scope USING "#,D";Digit_count
170 ENTER @Scope USING "#,"&VAL$(Digit_count)&"D";Stringlength
180 ALLOCATE Learn_string$[Stringlength+1]
190 ENTER @Scope USING "-K";Learn_string$
200 OUTPUT 707;":syst:err?"
```

```
210 ENTER 707;Errornum
220 PRINT "Error Status=";Errornum
```

See Also   SYSTem:SETup command and query. When HEADers and LONGform are ON, the SYSTem:SETup command performs the same function as the *LRN query. Otherwise, *LRN and SETup are not interchangeable.

*OPC (Operation Complete)

Command   `*OPC`

Use either the command or the query to notify the calling program when an operation is complete thus allowing the program to perform other tasks while waiting until notified. Refer also to **"*WAI (Wait-to-Continue)"** on page 101. The *OPC command and *OPC? query work with any of the following commands. Use with other commands is unreliable or fails.

- AUToscale   **105** *(In Jitter mode only.)*
- DIGitize   **107**
- LTESt   **124**
- PRECision   **330**
- PRECision:RFRequency   **331**
- PRECision:TREFerence   **331**
- PRINt   **112**
- PWAVeform:SAVE   **169**
- RUNTil   **126**
- RUNTil   **225**
- SINGle   **113**

The *OPC command sets the Standard Event Status Register's operation complete bit (OPC) when the operation is complete. The calling program must either poll periodically to see if the bit is set or setup an SRQ to be notified when the bit has been set. **Refer to "*ESR? (Event Status Register)" on page 92** for more information.

The *OPC? query holds the GPIB bus until the operations are complete at which time it returns a "1" in the output queue and calling code is then free to continue with other tasks. It causes the Status Byte Register's message available (MAV) bit to be set. **Refer to "*STB? (Status Byte)" on page 100**.

If instrument conditions have been set that can not be met and the *OPC command or query is sent out, the instrument halts remote execution and you must send a device clear or power down to restart the instrument. For more information, **refer to "Status Reporting" on page 26**.

*OPC Example   `10 OUTPUT 707;":PRINT;*OPC"`

*OPC? Example   `10 OUTPUT 707;":SINGle;*OPC?"`

*OPT? (Option)

Query   `*OPT?`

Returns a string with a list of installed hardware and software options. The query returns a 1 as the first character if option 86100C-001 or 86100D-ETR (enhanced trigger) is installed. If no options are installed, the string will have a 0 as the first character. The length of the returned string may increase as options become available in the future. Once implemented, an option name will be appended to the end of the returned string, delimited by a comma.

Restrictions	In software revisions A.05.00 and below, the query returns a list of any hardware options but does not include any software options.

Example	`10 OUTPUT 707;"*OPT?"`

*RCL (Recall)

Command	`*RCL <register>`
	Restores the state of the analyzer to a setup previously stored in the specified save/recall register. An analyzer setup must have been stored previously in the specified register. Registers 0 through 9 are general-purpose registers and can be used by the *RCL command. <register> is an integer, 0 through 9, specifying the save/recall register that contains the analyzer setup you want to recall.

Example	`10 OUTPUT 707;"*RCL 3"`

See Also	SAVe. An error message appears on the analyzer display if nothing has been previously saved in the specified register.

*RST (Reset)

Command	`*RST`
	Places the instrument in a known state. Table 17 lists the reset conditions as they relate to the analyzer commands. This is the same as using the front-panel default setup button.

Example	`10 OUTPUT 707;"*RST"`

Table 17   Default Setup (Sheet 1 of 5)

Acquisition	
Run/Stop	100 ms
	Grid on
	30
	Enabled
	8 hours
	Default legend
	Off
	Off (until the first marker is placed on the screen)
	User selectable if more than one source is available.
	28 ns
	0V
Points/Waveform (Record length)	Automatic - 1350 points
Averaging	Off
# of Averages	16
Trigger	
Source	Front Panel
Bandwidth	2.5 GHz
Hysteresis	Normal
Slope	Positive

Table 17   Default Setup (Sheet 2 of 5)

Gated Trigger	Off	
Level	0 V	
Time Base		
Units	Time	
Scale	1 ns/div	
Position	24 ns	
Reference	Left	
Display		
Persistence	Variable (oscilloscope mode) Gray Scale (Infinite) (Eye/Mask mode)	
Persistence Time	100 ms	
Graticule	Grid on	
Intensity	30	
Backlight Saver	Enabled	
Turn off backlight after	8 hours	
Colors	Default legend	
Labels	Off	
Markers		
Mode		
Readout	Off (until the first marker is placed on the screen)	
X1, Y1 source	User selectable if more than one source is available	
X1 position	28 ns	
Y1 position	0V	
X2, Y2 source	User selectable if more than one source is available	
X2 position	24 ns	
Y2 position	0V	
Measure	Oscilloscope mode	Eye/Mask mode
QuickMeas, Meas.1	V p-p	Extinction ratio
QuickMeas, Meas. 2	Period	Jitter
QuickMeas, Meas. 3	Frequency	Average power
QuickMeas, Meas. 4	Rise time	Crossing %
Start mask test	–	Off

Table 17   Default Setup (Sheet 3 of 5)

Define Measure	
Thresholds - percent	10%, 50%, 90%
Thresholds - volts	0.0, 1.6, 5.0
Top-Base Definition	Standard
Statistics	Off
Top-Base volts	0.0, 5.0
Measurements	Off
Start Edge	Rising, 1 level, middle
Stop Edge	Falling, 1 level, middle
Eye Window 1	40%
Eye Window 2	60%
Duty cycle distortion format	Time
Extinction ratio format	Decibel
Eye width	Time
Jitter	RMS
Average power	Watts
Waveform	
Memory display	Off
Waveform source	First available channel or memory 1
Memory type	Waveform
Math	
Function	Function 1
Function state	Off
Operator	Magnify
Operand 1	First available channel or memory 1
Operand 2	First available channel or memory 1
Horizontal scaling	Track source
Vertical scaling	Track source
Channel	
Display	On (lowest number installed channel; others are off)
Scale	50 µW/div or 10 mV/div
Offset	0.0 V or 0 W
Units	Volts (or watts)

Table 17 Default Setup (Sheet 4 of 5)

Filter	Dependent on module
Wavelength	Wavelength 1
Bandwidth	Dependent on module
Histogram	
Mode	Off
Axis	Horizontal
Window source	First available channel
Size	Horizontal - 4.0 divisions Vertical - 5.0 divisions
X1 position	25 ns
Y1 position	1 division up from bottom, value depends on module
X2 position	33 ns
Y2 position	1 division down from top, value depends on module
Utilities	
Cal Output	5.0 mv
Calibration Details	Off
Self Test	Scope Self Tests
Service Extensions	Off
Remote Interface	Unchanged
Dialog Preferences	Opaque Dialogs
Allow Multiple Active Dialogs	Off
Sound	enabled, volume 48
Limit Test	
Test	Off
Measurement	None
Fail when	Outside
Upper limit	10
Lower limit	–10
Run until	Forever
Run until failures	1 failure
Run until waveforms	1,000,000 waveforms
Store summary	Off
Store screen	Off

Table 17   Default Setup (Sheet 5 of 5)

Store waveforms	Off
Mask Test	
Test	Off
Scale source	Displayed channel
X1 position	2 divisions from left, 26 ns
1 level	2 divisions down
0 level	2 divisions up
Mask margins	Off
Run until	Forever
Failed waveforms	1 failure
Failed samples	1 sample
Waveforms	1,000,000
Samples	1,000,000
Store waveforms	Off
Store summary	Off
Store screen	Off

*SAV (Save)

Command   `*SAV <register>`

Stores the current state of the analyzer in a save register. <register> is an integer, 0 through 9, specifying which register to save the current analyzer setup. See also *RCL (Recall).

Example   `10 OUTPUT 707;"*SAV 3"`

*SRE (Service Request Enable)

Command   `*SRE <mask>`

Sets the Service Request Enable Register bits. By setting the *SRE, when the event happens, you have enabled the analyzer's interrupt capability. The scope will then do an SRQ (service request), which is an interrupt. <mask> is an integer, 0 to 255, representing a mask value for the bits to be enabled in the Service Request Enable Register as shown in Table 18 on page 100.

Example   This example enables a service request to be generated when a message is available in the output queue. When a message is available, the MAV bit is high.
`10 OUTPUT 707;"*SRE 16"`

Query   `*SRE?`

Returned Format   `<mask><NL>`

Example   This example places the current contents of the Service Request Enable Register in the numeric variable, Value.
`10 OUTPUT 707;"*SRE?"`

The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A "1" in the Service Request Enable Register enables the corresponding bit in the Status Byte Register. A "0" disables the bit.

Table 18   Service Request Enable Register Bits

Bit	Weight	Enables
7	128	OPER - Operation Status Register
6	64	Not Used
5	32	ESB - Event Status Bit
4	16	MAV - Message Available
3	8	Not Used
2	4	MSG - Message
1	2	USR - User Event Register
0	1	TRG - Trigger

*STB? (Status Byte)

Query   `*STB?`

Seturns the current contents of the Status Byte, including the Master Summary Status (MSS) bit. See Table 19 on page 100 for Status Byte Register bit definitions.

Returned Format   `<value><NL>`

`<value>` is an integer, from 0 to 255.

Example   This example reads the contents of the Status Byte into the numeric variable, Value.

```
10 OUTPUT 707;"*STB?"
20 ENTER 707;Value
```

In response to a serial poll (SPOLL), Request Service (RQS) is reported on bit 6 of the status byte. Otherwise, the Master Summary Status bit (MSS) is reported on bit 6. MSS is the inclusive OR of the bitwise combination, excluding bit 6, of the Status Byte Register and the Service Request Enable Register. The MSS message indicates that the scope is requesting service (SRQ).

Table 19   Status Byte Register Bits

Bit	Bit Weight	Bit Name	Condition
7	128	OPER	0 = no enabled operation status conditions have occurred 1 = an enabled operation status condition has occurred
6	64	RQS/MSS	0 = analyzer has no reason for service 1 = analyzer is requesting service
5	32	ESB	0 = no event status conditions have occurred 1 = an enabled event status condition occurred
4	16	MAV	0 = no output messages are ready 1 = an output message is ready
3	8	–	0 = not used

Table 19   Status Byte Register Bits

2	4	MSG	0 = no message has been displayed 1 = message has been displayed
1	2	USR	0 = no enabled user event conditions have occurred 1 = an enabled user event condition has occurred
0	1	TRG	0 = no trigger has occurred 1 = a trigger occurred
	0 = False = Low		1 = True = High

*TRG (Trigger)

Command   `*TRG`

The *TRG command has the same effect as the Group Execute Trigger message (GET) or RUN command. It acquires data for the active waveform display, if the trigger conditions are met, according to the current settings.

Example   `10 OUTPUT 707;"*TRG"`

*TST? (Test)

Query   `*TST?`

Causes the analyzer to perform a self-test, and places a response in the output queue indicating whether or not the self-test completed without any detected errors. Use the :SYSTem:ERRor command to check for errors. A zero indicates that the test passed and a non-zero indicates the self-test failed. You must disconnect all front-panel inputs before sending the *TST? query. If a test fails, refer to the troubleshooting section of the service guide. The Self-Test takes approximately 3 minutes to complete. When using timeouts in your program, 200 seconds duration is recommended.

Returned Format   `<result><NL>`

`<result>` is 0 for pass; non-zero for fail.

Example   `10 OUTPUT 707;"*TST?"`

*WAI (Wait-to-Continue)

Command   `*WAI`

Prevents the analyzer from executing any further commands or queries until all currently executing commands are completed. See *OPC for alternate methods for synchronization.

Example   `10 OUTPUT 707;"SINGle;*WAI"`

4 Root Level Commands

KEYSIGHT
TECHNOLOGIES

Root level commands control many of the basic operations of the analyzer that can be selected by pressing the labeled keys on the front panel. These commands are always recognized by the parser if they are prefixed with a colon, regardless of the current tree position. After executing a root level command, the parser is positioned at the root of the command tree. For any of the Standard Event Status Register bits to generate a summary bit, the bits must be enabled. These bits are enabled by using the *ESE common command to set the corresponding bit in the Standard Event Status Enable Register. URQ in the Event Status Register always returns 0. To generate a service request (SRQ) interrupt to an external computer, at least one bit in the Status Byte Register must be enabled. These bits are enabled by using the *SRE common command to set the corresponding bit in the Service Request Enable Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register. In the SRE query, bit 6 always returns 0. Various root level commands documented in this chapter query and set various registers within the register set.

AEEN

Command   `:AEEN <mask>`

Sets a mask into the Acquisition Limits Event Enable register. A "1" in a bit position enables the corresponding bit in the Acquisition Limits Event Register to set bit 9 in the Operation Status Register. The <mask> argument is the decimal weight of the enabled bits. Only bits 0 through 4 of the Acquisition Limits Event Enable Register are used at this time. Table 20 shows the enabled bits for some useful example mask values. Bits that are not marked as enabled by the mask are blocked from affecting the operation status register.

Query   `:AEEN?`

The query returns the current decimal value in the Acquisition Limits Event Enable register.

Returned Format   `[:AEEN] <mask><NL>`

Table 20   Enabled Bits for Some Useful Example Mask Values

Mask Value	Bit 4 CH4	Bit 3 CH3	Bit 2 CH2	Bit 1 CH1	Bit 0 COMP
0					
1					.
2				.	
3				.	.
4			.		
5			.		.
6			.	.	
7			.	.	.
8		.			
16	.				

ALER?

Query   `:ALER?`

Returns the current value of the Acquisition Limits Event Register as a decimal number and also clears this register. Bit 0 (COMP) of the Acquisition Limits Event Register is set when the acquisition completes. The acquisition completion criteria are set by the :ACQuire:RUNTil command.

Acquistion Limit Tests on Individual Channels   When in independent acquisition mode and a channel finishes the corresponding bit of the acquisition limit event register (ALER) is set. For example, when channel 1 limit is reached bit 1 of the ALER is set; when channel 2 limit is reached bit 2 of the ALER is set. Bit 0 of the ALER is not set until all channels that acquisition limit tests are being performed on have finished. If the acquisition limit of a channel is set to off then the corresponding bit of the ALER for that channel is not set during the acquisition limit test. ALER? return the decimal weight of the enabled bits of the ALER. For example, if channels 1and 2 have reached their acquisition limit and no other channels have acquisition limits specified, then the value returned by the ALER? will be 7 (111 in binary). Bits 0, 1, & 2 of the ALER will then be set.

Returned Format   `[:ALER] <value><NL>`

AUToscale

Command   `:AUToscale [<data rate>]`

This command causes the instrument to evaluate the current input signal and find the optimum conditions for displaying the signal. It adjusts the vertical gain and offset for the channel, and sets the time base on the lowest numbered input channel that has a signal. If signals cannot be found on any vertical input, the analyzer is returned to its former state.

Autoscale sets the following:

·   Channel Display, Scale, and Offset

·   Trigger and Level

·   Time Base Scale and Position

Autoscale turns off the following:

·   Measurements on sources that are turned off

·   Functions

·   Windows

·   Memories

No other controls are affected by Autoscale.

For faster and more reliable execution of the autoscale function, enter the signal's data rate using the optional <data rate> argument. The instrument uses this argument as an aid in setting the horizontal scaling for a signal. The value is only valid for NRZ eye diagrams or clock signals. The <data rate> argument sets the data rate in the same manner as the TRIGger:BRATe and TIMebase:BRATe commands. The limits for all three commands are identical. Normally, the valid range is 1 Mb/s to 160 Gb/s, however, in pattern lock, the range is 50 Mb/s to 160 Gb/s. When using the 86107A precision timebase, the data rate must be a multiple of the reference clock frequency. Refer to "PRECision:RFRequency" on page 331.

Restrictions   Software revision A.04.10 and above for <data rate> argument.

Example   This example sets the data rate to 155.520 Mb/s and automatically scales the analyzer for the input signal.
`10 OUTPUT 707;":AUTOSCALE 155.520E6"`

Query   `:AUToscale?`

Returns a string explaining the results of the last autoscale. The string is empty if the last autoscale completed successfully. The returned string stays the same until the next autoscale is executed.

The following are examples of strings returned by the AUToscale? query.
```
No channels turned on
Left module requires calibration for autoscale
Right module requires calibration for autoscale
Channel n signal is too small
Channel n signal is too high
Channel n signal exceeds the measurable range at the top
Channel n offset exceeds the measurable range at the bottom
No trigger or trigger too slow
Trigger is in Free Run
Unable to set horizontal scale/delay for channel n
```

Returned Format   `[:AUToscale] <data rate>`

BLANk

Command   `:BLANk {CHANnel<N> | FUNCtion<N> | WMEMory<N> | JDMemory | RESPonse<N> | HISTogram | CGMemory}`

Turns off an active channel, function, waveform memory, jitter data memory, TDR response, histogram, or color grade memory. The VIEW command turns them on. <N> is an integer, 1 through 4.

Restrictions   Software revision A.04.00 and above (86100C instruments) or 86100D instruments for jitter data memory argument.

Example   `10 OUTPUT 707;":BLANK CHANNEL1"`

CDISplay

Command   `:CDISplay [CHANnel<N>]`

Clears the display and resets all associated measurements. If the analyzer is stopped, all currently displayed data is erased. If the analyzer is running, all of the data in active channels and functions is erased; however, new data is displayed on the next acquisition. Waveform memories are not erased. If a channel is specified as a parameter, only the displayed data from that channel is cleared. <N> is an integer, 1 through 4.

Restrictions   In TDR mode (software revision A.06.00 and above), the optional channel argument is not allowed.

Example   `10 OUTPUT 707;":CDISPLAY"`

COMMents

Command   `:COMMents {LMODule | RMODule},"<comments_text>"`

Sets the comments field for the module. This field is used to describe options included in the module, or for user comments about the module. A maximum of 35 characters is allowed. The <comments_text> argument represents the ASCII string enclosed in quotation marks. The maximum length of the string is 35 characters.

Example   `10 OUTPUT 707;":COMMENTS LMODULE"`

Query   `:COMMents? {LMODule | RMODule}`

The query returns a string with the comments field associated with the module.

Returned Format   `[:COMMents] <string>`

CREE

Command   `:CREE <mask>`

Sets a mask into the Clock Recovery Event Enable Register. A "1" in a bit position enables the corresponding bit in the Clock Recovery Event Register to set bit 7 in the Operation Status Register. <mask> is the decimal weight of the enabled bits. Table 21 on page 107 shows the enabled bits for some useful example mask values. Bits that are not marked as enabled for a mask are blocked from affecting the operation status register.

Query   `:CREE?`

Returned Format   `[:CREE] <mask><NL>`

Table 21   Enabled Bits for Some Useful Example Mask Values

Mask Value	Bit 6 FIN	Bit 5 SPR2	Bit 4 NSPR2	Bit 3 SPR1	Bit 2 NSPR1	Bit 1 LOCK	Bit 0 UNLK
0							
1							·
2						·	
4					·		
8				·			
16			·				
32		·					
64	·						

CRER?

Query   `:CRER?`

Returns the current value of the Clock Recovery Event Register as a decimal number and also clears the register. Refer to "SPResent?" on page 162 for more detailed information on receiver one and receiver two. Refer to "Clock Recovery Event Register (CRER)" on page 34 for a definition of each bit in the register.

Returned Format   `[:CRER] <value><NL>`

DIGitize

Command   `:DIGitize [CHANnel<N> | FUNCtion<N> | RESPonse<N>]`

Invokes a special mode of data acquisition that is more efficient than using the RUN command when using averaging in the Oscilloscope mode. With the faster computations of the Agilent 86100B/C, the DIGitize command is no longer significantly faster than the RUN and RUNTil commands. In Jitter mode, the DIGitize command does not use any arguments, and the desired channel or function must be set up before this command is sent. See *OPC (Operation Complete) command on page 94 for synchronization of PRINT operations. <N> is an integer, 1 through 4.

The DIGitize command initializes the selected channels or functions, then it acquires them according to the current analyzer settings. When the signal is completely acquired (for example, when the specified number of averages have been taken), the analyzer is stopped.

In any instrument mode *except* Jitter mode, if you use the DIGitize command with channel, function, or response parameters, only the specified channels, functions, or responses are acquired. In Jitter mode, do not append any arguments to this command. To speed up acquisition, the waveforms are not displayed and their display state indicates "off." Subsequent to the digitize operation, the display of the acquired waveforms may be turned on for viewing, if desired. Other sources are turned off and their data is invalidated.

NOTE   Even though digitized waveforms are not displayed, the full range of measurement and math operators may be performed on them.

If you use the DIGitize command with no parameters, the digitize operation is performed on the channels or functions that were acquired with a previous digitize, run, or single operation. In this case, the display state of the acquired waveforms is not changed. Because the command executes more quickly without parameters, this form of the command is useful for repetitive measurement sequences. You can also use this mode if you want to view the digitize results because the display state of the digitized waveforms is not affected.

Data acquired with the DIGitize command is placed in the normal channel, function, or response.

NOTE   The DIGitize command is not intended for use with limit tests. Use the RUN and RUNTil commands instead. The stop condition for the RUN command is specified by commands ACQuire:RUNTil on page 126, MTEST:RUNTil on page 225, or LTEST on page 205.

NOTE   Before executing the DIGitize command for a differential or common mode response, the type of response must be specified by turning on the response. This is done using the :TDR{2|4}:RESPonse<N> command. Refer to "RESPonse" on page 321.

See Chapter 2, "Programming Examples" for examples of how to use DIGitize and its related commands.

Example   This example acquires data on channel 1 and function 2.
```
10 OUTPUT 707;":DIGITIZE CHANNEL1,FUNCTION2"
```

The ACQuire subsystem commands set up conditions such as TYPE and COUNT for the next DIGitize command. The WAVeform subsystem commands determine how the data is transferred out of the analyzer, and how to interpret the data.

JEE

Command   :JEE <mask>

Sets a mask into the Jitter Event Enable register. A "1" in a bit position enables the corresponding bit in the Jitter Event Register. This action sets bit 12 (JIT) in the Operation Status Register, which potentially can cause an SRQ to be generated. <mask> is the decimal value of the enabled bits. Only bits 0, 1, and 2 of the Jitter Event Enable Register are used at this time. The following table shows the enabled bits for each useful mask value. Bits that are not marked as enabled for a mask are blocked from affecting the operation status register.

Table 22   Enabled Bits for Mask Values

Mask Value	Bit 2 AREQD	Bit 1 JLOSS	Bit 0 EFAIL
0			
1			·
2		·	
3		·	·
4	·		
5	·		·
6	·	·	
7	·	·	·

Restrictions   Jitter mode. Software revision A.04.00 and above (86100C instruments) or 86100D instruments with Option 100 or 200.

Query   :JEE?

The query returns the current decimal value in the Jitter Event Enable Register.

Returned Format   [:JEE] <mask><NL>

JER?

Query   :JER?

Returns the current value of the Jitter Event Register as a decimal number and also clears the register. Bit 0 of the register is set when characterizing edges in Jitter Mode fails. Bit 1 of the register is set when pattern synchronization is lost in Jitter Mode. Bit 2 of the register is set when a parameter change in Jitter Mode has made autoscale necessary. Bit 12 of the Operation Status Register (JIT) indicates that one of the enabled conditions in the Jitter Event Register has occurred.

Restrictions   Jitter mode. Software revision A.04.00 and above (86100C instruments) or 86100D instruments with Option 100 or 200.

Returned Format   [:JER] <value><NL>

LER?

Query   :LER?

Reads the Local (LCL) Event Register. A "1" is returned if a remote-to-local transition has taken place due to the front-panel Local key being pressed. A "0" is returned if a remote-to-local transition has not taken place. After the LCL Event Register is read, it is cleared. Once this bit is set, it can only be cleared by reading the Status Byte, reading the register with the LER? query, or sending a *CLS common command.

Returned Format   [:LER] {1 | 0}<NL>

Example   10 OUTPUT 707;":LER?"

LTEE

Command   `:LTEE <mask>`

Sets a mask into the Limit Test Event Enable register. A "1" in a bit position enables the corresponding bit in the Limit Event Register to set bit 8 in the Operation Status Register. <mask> is the decimal weight of the enabled bits. Only bits 0 and 1 of the Limit Test Event Register, are used at this time. The following table shows the enabled bits for each useful mask value. Bits that are not marked as enabled for a mask are blocked from affecting the operation status register.

Table 23   Enabled Bits for Mask Values

Mask Value	Bit 1 FAIL	Bit 0 COMP
0		
1		·
2	·	
3	·	·

Query   `:LTEE?`

Returned Format   `[:LTEE] <mask><NL>`

LTER?

Query   `:LTER?`

Returns the current value of the Limit Test Event Register as a decimal number and also clears this register. Bit 0 (COMP) of the Limit Test Event Register is set when the Limit Test completes. The Limit Test completion criteria are set by the LTESt:RUN command. Bit 1 (FAIL) of the Limit Test Event Register is set when the Limit Test fails. Failure criteria for the Limit Test are defined by the LTESt:FAIL command.

Returned Format   `[:LTER] <value><NL>`

MODel?

Query   `:MODel? {FRAMe | LMODule | RMODule}`

Returns the Agilent model number for the 86100C/D or module. The 86108A Precision Waveform Analyzer module only has one model number and either the LMODule and RMODule arguments can be used to return it. The query returns a string which is six-character alphanumeric model number in quotation marks. Output is determined by header and longform status as in Table 24.

Returned Format   `[:MODel] <string>`

Table 24   Model? Returned Format

HEADER		LONGFORM		Example Responses
ON	OFF	ON	OFF	
	·		·	86100C
	·	·		86100C
·			·	:MOD 86100C
·		·		:MODEL 86100C

Example	`10 OUTPUT 707;":Model? FRAME"`

MTEE

Command	`:MTEE <mask>`

Sets a mask into the Mask Event Enable register. A "1" in a bit position enables the corresponding bit in the Mask Test Event Register to set bit 10 in the Operation Status Register. <mask> is the decimal weight of the enabled bits. Only bits 0 and 1 of the Mask Test Event Register are used at this time. The following table shows the enabled bits for each useful mask value. Bits that are not marked as enabled for a mask are blocked from affecting the operation status register.

Table 25   Enabled Bits for Mask Values

Mask Value	Bit 1 FAIL	Bit 0 COMP
0		
1		•
2	•	
3	•	•

Query	`:MTEE?`
Returned Format	`[:MTEE] <mask><NL>`

MTER?

Query	`:MTER?`

Returns the current value of the Mask Test Event Register as a decimal number and also clears this register. Bit 0 (COMP) of the Mask Test Event Register is set when the Mask Test completes. Bit 1 (FAIL) of the Mask Test Event Register is set when the Mask Test fails. This will occur whenever any sample is recorded within any region defined in the mask.

Returned Format	`[:MTER] <value><NL>`

OPEE

Command	`:OPEE <mask>`

Sets a mask in the Operation Status Enable register. Each bit that is set to a "1" enables that bit to set bit 7 in the Status Byte Register, and potentially causes an SRQ to be generated. Bit 5, Wait for Trig, is used. Other bits are reserved. <mask> The decimal weight of the enabled bits.

Query	`:OPEE?`

The query returns the current value contained in the Operation Status Enable register as a decimal number.

Returned Format	`[:OPEE] <value><NL>`

OPER?

Query	`:OPER?`

Returns the value contained in the Operation Status Register as a decimal number and also clears this register. This register is the summary of the CLCK bit (bit 7), LTEST bit (bit 8), ACQ bit (bit 9) and MTEST bit (bit 10). The CLCK bit is set by the Clock Recovery Event Register and indicates that a clock event has occurred. The LTEST bit is set by the Limit Test Event Register and indicates that a limit test has failed or completed. The ACQ bit is set by the Acquisition Event Register and indicates that an acquisition limit test has completed. The MTEST bit is set by the Mask Test Event Register and indicates that a mask limit test has failed or completed.

Returned Format   `[:OPER] <value><NL>`

PTEE

Command   `:PTEE <mask>`

Sets a mask into the Precision Timebase Event Enable register. A "1" in a bit position enables the corresponding bit in the Precision Timebase Event Register to set bit 11 in the Operation Status Register. <mask> is the decimal weight of the enabled bits. Only bit 0 of the Precision Timebase Event Register are used at this time. The useful mask values are shown in the following table. The following table shows the enabled bits for each useful mask value. Bits that are not marked as enabled for a mask are blocked from affecting the operation status register.

Restrictions   Software revision A.03.01 and above

Table 26   Enabled Bits for Mask Values

Mask Value	Bit 0 LOSS
0	
1	.

Query   `:PTEE?`

Returned Format   `[:PTEE] <mask><NL>`

PTER?

Query   `PTER?`

Returns the current value of the Precision Timebase Event Register as a decimal number and also clears this register. Bit 0 (LOSS) of the Precision Timebase Event Register is set when loss of the time reference occurs. Time reference is lost when a change in the amplitude or frequency of the reference clock signal is detected. The Precision Timebase Event Register is read and cleared with the PTER? query. When the LOSS bit is set, it in turn sets the PTIME bit (bit 11) of the Operation Status Register. Results from the Precision Timebase Register can be masked by using the PTEE command to set the Precision Timebase Event Enable Register to the value 0. You enable the LOSS bit by setting the mask value to 1.

Restrictions   Software revision A.03.01 and above

Returned Format   `[:MTER] <value><NL>`

PRINt

Command   `:PRINt`

Outputs a copy of the screen to a printer or other device destination, such as a file, specified in the HARDcopy subsystem. You can specify the selection of the output and the printer using the HARDcopy subsystem commands. See *OPC (Operation Complete) command on for synchronization of PRINT operations.

Example   `10 OUTPUT 707;":PRINT"`

RECall:SETup

Command   `:RECall:SETup <setup_memory_num>`

Recalls a setup that was saved in one of the analyzer's setup memories. You can save setups using either the STORe:SETup command or the front panel. <setup_memory_num> is the setup memory number, an integer, 0 through 9.

Example   `10 OUTPUT 707;":RECall:SETup 2"`

RUN

Command   `:RUN [CHANnel<N>]`

Starts the instrument running where the instrument acquires waveform data according to its current settings. Acquisition runs repetitively until the analyzer receives a correspondent STOP command. <N> is an integer, 1 through 4. The execution of the RUN command is subordinate to the status of ongoing limit tests. (see commands ACQuire:RUNTil on , MTEST:RUNTil on , and LTESt:RUNTil on). The RUN command will not restart a full data acquisiton if the stop condition for a limit test has been met.

Restrictions   In TDR mode (software revision A.06.00 and above), the optional channel argument is not allowed.

Example   `10 OUTPUT 707;":RUN"`

SERial

Command   `:SERial {FRAMe | LMODule | RMODule},<string>`

Sets the serial number for the 86100C/D or module. Because the serial number is entered by Agilent Technologies, setting the serial number is not normally required unless the instrument is serialized for a different application. The <string> argument is a ten-character alphanumeric serial number enclosed with quotation marks. The analyzer's serial number is part of the string returned for the *IDN? query, described in , "Common Commands". The 86108A Precision Waveform Analyzer module only has one serial number and either the LMODule and RMODule arguments can be used to specify it.

Example   `10 OUTPUT 707;":SERIAL FRAME,""1234A56789"""`

Query   `:SERial? {FRAMe | LMODule | RMODule}`

Returned Format   `[:SERial] <string><NL>`

Example   `10 OUTPUT 707;":SERIAL? FRAME"`

SINGle

Command   `:SINGle`

Initiates a single acquisition when the next trigger event occurs. This command should be followed by *WAI, *OPC, or *OPC? in order to synchronize data acquisition with remote control.

Example	`10 OUTPUT 707;":SINGLE"`

STOP

Command	`:STOP [CHANnel<N>]`
	Stops data acquisition for the active display. If no channel is specified, all active channels are affected. To restart the acquisition, use the RUN or SINGle command. <N> is an integer, 1 through 4.
Restrictions	In TDR mode (software revision A.06.00 and above), the optional channel argument is not allowed.
Example	`10 OUTPUT 707;":STOP"`

STORe:SETup

Command	`:STORe:SETup <setup_memory_num>`
	Saves the current instrument setup in one of the setup memories. <setup_memory_num> is the setup memory number, an integer, 0 through 9.

STORe:WAVeform

Command	`:STORe:WAVeform {CHANnel<N>	FUNCtion<N>	WMEMory<N>	RESPonse<N>},<destination>`
	Copies a channel, function, stored waveform, or TDR response to a waveform memory or to color grade memory. The parameter preceding the comma specifies the source and can be any channel, function, response, color grade memory, or waveform memory. The parameter following the comma is the destination, and can be any waveform memory. <N> is an integer, 1 through 4. Only channels or functions can be sources for color grade memory. <destination> is {WMEMory<N>	CGMemory}.
Restrictions	This command operates on waveform and color grade gray scale data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.
Example	`10 OUTPUT 707;":STORE:WAVEFORM CHANNEL1,WMEMORY3"`

TER?

Query	`:TER?`
	Reads the Trigger Event Register. A "1" is returned if a trigger has occurred. A "0" is returned if a trigger has not occurred. Once this bit is set, you can clear it only by reading the register with the TER? query, or by sending a *CLS common command. After the Trigger Event Register is read, it is cleared.
Returned Format	`[:TER] {1	0}<NL>`
Example	`10 OUTPUT 707;":TER?"`

UEE

Command	`:UEE <mask>`
	Sets a mask into the User Event Enable register. A "1" in a bit position enables the corresponding bit in the User Event Register to set bit 1 in the Status Byte Register and, thereby, potentially cause an SRQ to be generated. Only bit 0 of the User Event Register is used at this time; all other bits are reserved. <mask> is the decimal weight of the enabled bits.
Query	`:UEE?`

Returned Format	`[:UEE] <mask><NL>`

UER?

Query	`:UER?`

Returns the current value of the User Event Register as a decimal number and also clears this register. Bit 0 (LCL - Remote/Local change) is used. All other bits are reserved.

Returned Format	`[:UER] <value><NL>`

VIEW

Command	`:VIEW {CHANnel<N>	FUNCtion<N>	WMEMory<N>	JDMemory	RESPonse<N>	HISTogram	CGMemory}`

Turns on a channel, function, waveform memory, jitter data memory, TDR response, histogram, or color grade memory. <N> is an integer, 1 through 4.

NOTE	This command operates on waveform and color grade gray scale data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode with an argument other than JDMemory. It generates a "Control is set to default" error for the HISTogram argument and "Illegal parameter value" error for other arguments.

Restrictions	Software revision A.04.00 and above (86100C instruments) or 86100D instruments for jitter data memory argument.
Example	`10 OUTPUT 707;":VIEW CHANNEL1"`
See Also	The BLANk command turns off a channel, function, waveform memory, TDR response, histogram, or color grade memory.

5 System Commands

SYSTem subsystem commands control the way in which query responses are formatted, send and receive setup strings, and enable reading and writing to the advisory line of the analyzer. You can also set and read the date and time in the analyzer using the SYSTem subsystem commands.

DATE

Command   `:SYSTem:DATE <day>,<month>,<year>`

Sets the date in the analyzer, and is not affected by the *RST common command. The argument <day> specifies the day in the format <1. . . .31>. The argument <month> specifies the month in the format <1, 2, ...12> | <JAN, FEB, MAR . . . .>. The argument <year> specifies the year in the format <yyyy> | <yy>. The values range from 1992 to 2035.

Example   The following example sets the date to July 1, 1997.
```
10 OUTPUT 707;":SYSTEM:DATE 7,1,97"
```

Query   `:SYSTem:DATE?`

The query returns the current date in the analyzer.

Returned Format   `[:SYSTem:DATE] <day> <month> <year>><NL>`

Example   The following example queries the date.
```
10 DIM Date$ [50]
20 OUTPUT 707;":SYSTEM:DATE?"
30 ENTER 707; Date$
```

DSP

Command   `:SYSTem:DSP <string>`

Writes a quoted string, excluding quotation marks, to the advisory line of the instrument display. If you want to clear a message on the advisory line, send a null (empty) string. The argument <string> is an alphanumeric character array up to 92 bytes long.

KEYSIGHT
TECHNOLOGIES

5   System Commands

Example	The following example writes the message, "Test 1" to the advisory line of the analyzer. `10 OUTPUT 707;":SYSTEM:DSP ""Test 1"""`
Query	`:SYSTem:DSP?` Returns the last string written to the advisory line. This may be a string written with a SYSTem:DSP command, or an internally generated advisory. The string is actually read from the message queue. The message queue is cleared when it is read. Therefore, the displayed message can only be read once over the bus.
Returned Format	`[:SYSTem:DSP] <string><NL>`

Example	The following example places the last string written to the advisory line of the analyzer in the string variable, Advisory$.
	`10 DIM Advisory$[89]!Dimension variable`
	`20 OUTPUT 707;":SYSTEM:DSP?"`
	`30 ENTER 707;Advisory$`

ERRor?

Query	`:SYSTem:ERRor? [{NUMBer	STRing}]` Returns the next error number in the error queue. Positive valued error numbers are instrument specific. Negative valued error numbers indicate a standard SCPI error. When either NUMBer or no parameter is specified in the query, only the numeric error code is output. When STRing is specified, the error number is output followed by a comma and a non-quoted string describing the error. Refer to Table 12 on page 42 for a list of error numbers, messages, and descriptions.
Returned Format	`[:SYSTem:ERRor] <error_number>[,<string>]<NL>` The <error_number> is anumeric error code. The <string> describes the error.

Example	The following example reads the oldest error number and message in the error queue into the string variable, Condition$.
	`10 DIM Condition$[64]!Dimension variable`
	`20 OUTPUT 707;":SYSTEM:ERROR? STRING"`
	`30 ENTER 707;Condition$`
	The error queue is 30 errors deep and operates on a first-in, first-out (FIFO) basis. Successively sending the SYSTem:ERRor query returns the error numbers in the order that they occurred until the queue is empty. When the queue is empty, this query returns headers of 0, "No error." Any further queries return zeros until another error occurs. Note that front-panel generated errors are also inserted in the error queue and the Event Status Register.

NOTE	Send the *CLS common command to clear the error queue and Event Status Register before you send any other commands or queries.

See Also	"Error Messages" on page 41 for more information on error messages and their possible causes.

FCONfig

Command	`:SYSTem:FCONfig {LEGacy	HYBRid	STANDard}` Changes the configuration of the 86100D. In LEGacy configuration, the user interface controlling the instrument is the version that the instrument booted into in software revisions below A.12.00. The programming commands documented in this book apply to LEGacy configuration.

118   Programmer's Guide

In HYBRid configuration, the instrument user interface is FlexDCA or FlexDCA N1010A on a PC controlling the legacy user interface. This configuration was available on software revisions A.10.01 through A.12.00. Use the programming commands documented in FlexDCA's online help. With the instrument in HYBRid configuration, click Help > Contents to locate the programming commands.

In STANDard configuration the FlexDCA user interface directly controls the instrument. By default, the instrument boots up in this configuraton, which supports one-slot mini modules. Use the programming commands documented in FlexDCA's online help. With the instrument in STANDard configuration, click Help > Contents to locate the programming commands.

NOTE   The features available vary between the LEGacy, HYBRid, and STANDard configurations. Consult the HYBRid or STANDard help system to learn about the differences.

Restrictions	Software revision A.12.00 and above.
Example	`10 OUTPUT 707;":SYSTEM:FONFIG HYBRID"`
Query	`:SYSTem:FCONFIG?`
Returned Format	`[:SYSTem:FCONFIG] {LEGacy	HYBRid	STANDard}`

HEADer

Command	`:SYSTem:HEADer {{ON	1}	{OFF	0}}`

Specifies whether the instrument will output a header for query responses. When SYSTem:HEADer is set to ON, the query responses include the command header. Turn headers off when returning values to numeric variables. Headers are always off for all common command queries because headers are not defined in the IEEE 488.2 standard.

Example	The following example sets up the analyzer to output command headers with query responses.

```
10 OUTPUT 707;":SYSTEM:HEADER ON"
```

Query	`:SYSTem:HEADer?`
Returned Format	`[:SYSTem:HEADer] {1	0}<NL>`

Example	This example examines the header to determine the size of the learn string. Memory is then allocated to hold the learn string before reading it. To output the learn string, the header is sent, then the learn string and the EOF.

```
10 DIM Header$[64]
20 OUTPUT 707;"syst:head on"
30 OUTPUT 707;":syst:set?"
40 More_chars:   !
50 ENTER 707 USING "#,A";This_char$
60 Header$=Header$&This_char$
70 IF This_char$<>"#" THEN More_chars
80 !
90 ENTER 707 USING "#,D";Num_of_digits
100 ENTER 707 USING "#,"&VAL$(Num_of_digits)&"D";Set_size
110 Header$=Header$&"#"&VAL$(Num_of_digits)&VAL$(Set_size)
120!
130 ALLOCATE INTEGER Setup(1:Set_size)
140 ENTER 707 USING "#,B";Setup(*)
150 ENTER 707 USING "#,A";Eof$
160 !
170 OUTPUT 707 USING "#,-K";Header$
180 OUTPUT 707 USING "#,B";Setup(*)
190 OUTPUT 707 USING "#,A";Eof$
200
```

LONGform

Command	`:SYSTem:LONGform {ON	1	OFF	0}`

Specifies the format for query responses. If the LONGform is set to OFF, command headers and alpha arguments are sent from the instrument in the short form (abbreviated spelling). If LONGform is set to ON, the whole word is output. This command has no effect on input headers and arguments sent to the instrument. Headers and arguments may be sent to the instrument in either the long form or short form, regardless of the current state of the LONGform command.

Example	The following example sets the format for query response from the instrument to the short form (abbreviated spelling).

`10 OUTPUT 707;":SYSTEM:LONGFORM OFF"`

Query	`:SYSTem:LONGform?`

The query returns the current state of the SYSTem:LONGform command.

Returned Format	`[:SYSTem:LONGform] {0	1}<NL>`

Example	`120 OUTPUT 707;":SYSTEM:LONGFORM?"`

MODE

Command	`:SYSTem:MODE {EYE	OSCilloscope	TDR	JITTer}`

Sets the system mode. Specifying Eye/Mask mode, turns off all active channels except the lowest numbered channel. Changing to Eye/Mask mode turns off averaging for all modes unless Pattern Lock (:TRIGger:PLOCk) is turned on. If a TDR/TDT module is present, changing to TDR/TDT mode using this command turns on averaging for both TDR/TDT and Oscilloscope modes. Because some DCA features are unavailable in Jitter Mode, refer to "Commands Unavailable in Jitter Mode" on page 39.

Restrictions	Software revision A.04.00 and above (86100C instruments) or 86100D instruments for Jitter mode argument. Jitter mode is only available on 86100C/D mainframes with Option 100 or 200.

Example	`10 OUTPUT 707;":SYSTEM:MODE EYE"`

Query	`:SYSTem:MODE?`

Returned Format	`[:SYSTem:MODE] {EYE	OSC	TDR	JITT}`

Example	`20 OUTPUT 707;":SYSTEM:MODE?"`

SETup

Command	`:SYSTem:SETup <binary_block_data>`

Sets up the instrument as defined by the data in the setup string from the controller. <binary_block_data> is a string, consisting of bytes of setup data. The number of bytes is a dynamic number that is read and allocated by the analyzer's software.

Example	The following example sets up the instrument as defined by the setup string stored in the variable, Set$. # is an BASIC image specifier that suppresses the automatic output of the EOI sequence following the last output item. K is an BASIC image specifier that outputs a number or string in standard form with no leading or trailing blanks.

`10 OUTPUT 707 USING "#,-K";":SYSTEM:SETUP ";Set$`

Query	`:SYSTem:SETup?`

The query outputs the instrument's current setup to the controller in binary block data format as defined in the IEEE 488.2 standard. When headers and LONGform are on, the SYSTem:SETup query operates the same as the *LRN query in the common commands. Otherwise, *LRN and SETup are not interchangeable.

Returned Format   `[:SYSTem:SETup] #NX...X<setup data string><NL>`

The first character in the setup data string is a number added for disk operations.

Example   The following example stores the current instrument setup in the string variable, Set$. –K is an BASIC image specifier which places the block data in a string, including carriage returns and line feeds, until EOI is true, or when the dimensioned length of the string is reached.

```
10 DIM Set$[15000]!Dimension variable
20 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
30 OUTPUT 707;":SYSTEM:SETUP?"
40 ENTER 707 USING "-K";Set$
50 END
```

TIME

Command   `:SYSTem:TIME <hour>,<minute>,<second>`

Sets the time in the instrument, and is not affected by the *RST common command. <hour> is 0. . . .23. <minute> is 0. . . .59. <second> is 0. . . .59.

Example   `10 OUTPUT 707;":SYSTEM:TIME 10,30,45"`

Query   `:SYSTem:TIME?`

Returned Format   `[:SYSTem:TIME] <hour>,<minute>,<second>`

6 Acquire Commands

The ACQuire subsystem commands set up conditions for acquiring waveform data, including the DIGitize root level command. The commands in this subsystem select the number of averages and the number of data points. This subsystem also includes commands to set limits on how much data is acquired, and specify actions to execute when acquisition limits are met.

AVERage

Command   `:ACQuire:AVERage {{ON | 1} | {OFF | 0}}`

Enables or disables averaging. When ON, the analyzer acquires multiple data values for each time bucket, and averages them. When OFF, averaging is disabled. To set the number of averages, use the :ACQuire:COUNt command described later in this chapter.

> **NOTE**   Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Query   `:ACQuire:AVERage?`

Returned Format   `[:ACQuire:AVERage] {1 | 0}<NL>`

Example   `10 OUTPUT 707;":ACQUIRE:AVERAGE ON"`

BEST

Command	`:ACQuire:BEST {THRuput | FLATness}`

When averaging is enabled with ACQuire:AVERage, the FLATness option improves the step flatness by using a signal processing algorithm within the instrument. You should use this option when performing TDR measurements or when step flatness is important. The THRuput option improves the instrument's throughput and should be used whenever best flatness is not required.

NOTE	Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Query	`:ACQuire:BEST?`
Returned Format	`[:ACQuire:BEST] {THRuput | FLATness}<NL>`
Example	`10 OUTPUT 707;":ACQUIRE:BEST FLATNESS"`

COUNt

Command	`:ACQuire:COUNt <value>`

Sets the number of averages for the waveforms. In the AVERage mode, the ACQuire:COUNt command specifies the number of data values to be averaged for each time bucket before the acquisition is considered complete for that time bucket. <value> is an integer, 1 to 4096, specifying the number of data values to be averaged.

Query	`:ACQuire:COUNt?`
Returned Format	`[:ACQuire:COUNt] <value><NL>`
Example	`10 OUTPUT 707;":ACQUIRE:COUNT 16"`

EYELine

Command	`:ACQuire:EYELine {{ON | 1} | {OFF | 0}}`

Enables or disables eyeline mode. It is only available when pattern lock is turned on in Oscilloscope or Eye/Mask modes. When eyeline is turned on, the relative trigger bit is incremented after each acquisition. When combined with averaging, averaged eyes can be acquired. Pattern lock and eyeline are only available on an 86100C-001 or 86100D-ETR instruments.

Restrictions	Software revision A.04.00 and above (86100C instruments) or 86100D instruments.
Query	`:ACQuire:EYELine?`
Returned Format	`[:ACQuire:EYELine] {1 | 0}<NL>`
Example	`10 OUTPUT 707; ":ACQUIRE:EYELINE ON"`

LTESt

Command	`:ACQuire:LTESt [ALL | INDividual]`

Sets the mode for acquisition limit tests. The default is ALL. When it is set to INDividual, the :ACQuire:RUNtil command can be used with the optional channel parameter to specify conditions for each channel individually. When it is set to ALL, acquisition limit tests are performed on all channels simultaneously.

Restrictions	In TDR mode (software revision A.06.00 and above), the optional INDividual argument is not allowed.
Query	`:ACQuire:LTESt?`
Returned Format	`[:ACQuire:LTESt] {ALL	IND} <NL>`
Example	`10 OUTPUT 707;":ACQUIRE:LTEST ALL"`

POINts

Command	`:ACQuire:POINts {AUTO	<points_value>}`
	Sets the requested memory depth for an acquisition. Always query the points value with the WAVeform:POINts query or WAVeform:PREamble to determine the actual number of acquired points. You can set the points value to AUTO, which allows the analyzer to select the number of points based upon the sample rate and time base scale. <points_value> is an integer representing the memory depth. The points value range is 16 to 16,384 points. See also :WAVeform:DATA.
Restrictions	This command operates on waveform data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.
Query	`:ACQuire:POINts?`
Returned Format	`[:ACQuire:POINts] <points_value><NL>`
Example	`10 OUTPUT 707;":ACQUIRE:POINTS 500"`

REYE

Command	`:ACQuire:REYE {ON	OFF}`
	In Eye/Mask mode, turns on and off Rapid Eye data acquisition, which significantly reduces the time required to acquire the waveform samples. Rapid Eye employs several techniques that improve measurement efficiency. Use the command "REYE:INTerval" on page 126 to select the number of UI (Unit Intervals) to acquire the data from. The database is expanded as required, while keeping the displayed eye (usually 1.6 UIs) unchanged. All acquired samples, even those residing in an incomplete UI area, are properly applied. For example, after an autoscale, 1.6 UIs are displayd and 40% of the samples occur before or after the UI area. With Rapid Eye on, these sample points will now be recorded and included in their proper position within the UI. Use the command "BRATe" on page 329 to enter the bit rate. When multiple eye diagrams are displayed, use the command "REYE:ASKew" on page 125 to enable an autoscale to align the eyes using software delay.
	When using this Rapid Eye with waveform acquisition limits, set the record length using "POINts" on page 125 to AUTo.
Restrictions	Software revision A.10.60 and above. On 86100D instruments, Rapid Eye is enabled. 86100C instruments require software Option 500, "Productivity Package."
Query	`:ACQuire:REYE?`
Returned Format	`[:ACQuire:REYE] {ON	OFF}<NL>`
Example	`10 OUTPUT 707;":ACQUIRE:REYE ON"`

REYE:ASKew

Command	`:ACQuire:REYE:ASKew {ON	OFF}`

When using this Rapid EyeIf with multiple eye diagrams displayed, enables an autoscale to align the eyes when rapid eye is on. The eye diagrams are aligned using software delay.

Restrictions	Software revision A.10.60 and above. On 86100D instruments, Rapid Eye is enabled. 86100C instruments require software Option 500, "Productivity Package."
Query	`:ACQuire:REYE:ASKew?`
Returned Format	`[:ACQuire:REYE:ASKew] {ON	OFF}<NL>`
Example	`10 OUTPUT 707;":ACQUIRE:REYE:ASKew ON"`

REYE:INTerval

Command	`:ACQuire:REYE:INTerval <points_value>}`
	Enters an integer that specifies the number of UI (Unit Intervals) to acquire the data from for a Rapid Eye. The database is expanded as required for the number of UIs, while keeping the displayed eye (usually 1.6 UIs) unchanged. Use the command **"REYE"** on page 125 to turn on Rapid Eye measurements..
Restrictions	Software revision A.10.60 and above. On 86100D instruments, Rapid Eye is enabled. 86100C instruments require software Option 500, "Productivity Package."
Query	`:ACQuire:REYE:INTerval?`
Returned Format	`[:ACQuire:POINts:INTerval] <points_value><NL>`
Example	`10 OUTPUT 707;":ACQUIRE:REYE:INTerval 2"`

RUNTil

Command	`:ACQuire:RUNTil {OFF	WAVeforms,<number_of_waveforms>	SAMPles,` `<number_of_samples>	PATTerns,<number_of_pattern_repetitions>}[,CHANnel<N>]`
	Selects the acquisition run until mode. The RUNTil command does not initiate run mode, for which you need to send the root-level command **"RUN"** on page 113. For this reason, you will need to send RUNTil to set up the acquisition conditions followed by RUN to start continuous sweeps.
	The acquisition may be set to run until *n* waveforms, *n* patterns, or *n* samples have been acquired, or to run forever (OFF). If more than one run until criteria is set, then the instrument will act upon the completion of whichever run until criteria is achieved first. The PATTerns argument is valid only when the Eyeline feature is on or when the instrument is in Jitter Mode. The optional channel parameter can be set to specify RUNTil conditions on each channel individually when the :ACQuire:LTESt command is set to INDividual. If the acquisition limit test mode is set to INDividual and the :ACQuire:RUNTil OFF command is sent with no channel specified, all channels will be set to OFF. To turn off acquisition limit tests for an individual channel, you must specify the channel.
	<number_of_waveforms> is an integer, 1 through 2^{31}–1. <number_of_samples> is an integer, 1 through 2^{31}–1. <number_of_pattern_repetitions> is an integer, 1 through 2^{31}–1. <N> is an integer, 1 through 4.
Restrictions	Software revision A.04.00 and above (86100C instruments) or 86100D instruments for the PATTerns argument.
Query	`:ACQuire:RUNTil? [CHANnel<N>]`
	Returns the currently selected run until state. If the channel parameter is specified, the run until state of the specified channel is returned.

Returned Format   `[:ACQuire:RUNTil] {OFF | WAVeform, <n waveforms> |`
`PATT,<number_of_pattern_repetitions> | SAMPles, <n samples>}<NL>`

Examples   `10 OUTPUT 707;":ACQuire:RUNTIL SAMPLES,200"`
`20 OUTPUT 707;":RUN"`

The following example specifies that Channel 1 acquisition runs until 300 waveforms have been obtained.

`write_IO (":ACQuire:LTESt IND");`
`write_IO (":ACQuire:RUNTil WAVeforms, 300, CHANnel1");`
`write_IO (":RUN");`

SSCReen

Command   `:ACQuire:SSCReen {OFF | DISK [,<filename>]}`

Saves a copy of the screen when the acquisition limit is reached (number of averages and the number of data points). To capture a screen image at any time, use the command **"SIMage"** on page 170. To capture a screen image when a limit test fails, use the command **"SSCReen"** on page 206. To capture a screen image when a mask test fails, use the command **"SSCReen"** on page 228.

Use the SSCReen command to specify the name, type, and location to save a screen capture. Then, use the command **"RUNTil"** on page 126 to specify and arm the conditions for capturing a screen capture. Each time that the specified acquisition limit is reached, a screen capture will be saved. The argument DISK and optional filename specifies that a file be saved to a disk. OFF turns off the save action. The <filename> argument is an ASCII string enclosed in quotation marks. With each screen capture, the file is overwritten. If you want to save the results of consecutive limit tests, do not include an optional filename. The default filename, *AcqLimitScreenX.bmp*, will be used where X is an incremental number assigned by the instrument.

The save screen options established by the commands ACQuire:SSCReen DISK, ACQuire:SSCReen:AREA, and ACQuire:SSCReen:IMAG are stored in the instrument's memory and will be employed in consecutive save screen operations, until changed by the user. This includes the <filename> parameter for the ACQuire:SSCReen DISK command.

The *filename* field includes the network path and the directory in which the file will be saved, as well as the file format that will be used. The following is a list of valid file locations:

- Files can only be created within the folder "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.
- Files can not be saved on the root folder of the D: drive (C: on 86100A/B).
- Files can not be saved on USB removable drives. To save files on a USB drive, use front-panel controls. *(Applies only to firmware version A.09.00 and below)*
- Using the command **"CDIRectory"** on page 165 to change the present working directory has no effect on the location of saved files.

If a filename is specified without a path, the default path will be D:\User Files\screen images. (C drive on 86100A/B instruments.) The default file type is a bitmap (.bmp). The following graphics formats are available by specifying a file extension: PCX files (.pcx), EPS files (.eps), Postscript files (.ps), JPEG files (.jpg), TIFF files (.tif), and GIF files (.gif).

NOTE   For .gif and .tif file formats, this instrument uses LZW compression/decompression licensed under U.S. patent No 4,558,302 and foreign counterparts. End user should not modify, copy, or distribute LZW compression/decompression capability. For .jpg file format, this instrument uses the .jpg software written by the Independent JPEG Group.

Table 27   Example Filenames

File Name	File Saved in Directory...
"test.jpg"	D:\User Files\screen images\ This is the default folder. Filenames without a path are saved to this folder.
"subfolder\test.jpg"	D:\User Files\screen images\subfolder The subfolder must already exist before saving the file.
"D:\User Files\subfolder\test.jpg"	D:\User Files\subfolder The subfolder must already exist before saving the file.
"D:\User Files\test.jpg"	D:\User Files
"D:\test.jpg"	This is not a valid file location. The file is not saved.
"E:test4.eps"	File saved in the instrument's drive E:, that could be mapped to any disk in the network.
"\\computer-ID\d$\test3.bmp"	File saved in drive D: of computer "computer-ID", provided all permissions are set properly.

Query   `:ACQuire:SSCReen?`

Returned Format   `[:ACQuire:SSCReen] {OFF | DISK [,<filename>]}<NL>`

Example   `10 OUTPUT 707;":ACQUIRE:SSCREEN DISK, "test.jpg""`
`20 OUTPUT 707;":ACQUIRE:RUNTIL WAV,5"`

SSCReen:AREA

Command   `:ACQuire:SSCReen:AREA {GRATicule | SCReen}`

Selects which data from the screen is to be saved to disk when the run until condition is met. When you select GRATicule, only the graticule area of the screen is saved (this is the same as choosing Waveforms Only in the Specify Report Action for acquisition limit test dialog box). When you select SCReen, the entire screen is saved.

Query   `:ACQuire:SSCReen:AREA?`

Returned Format   `[:ACQuire:SSCReen:AREA] {GRATicule | SCReen}<NL>`

Examples   `10 OUTPUT 707;":ACQUIRE:SSCREEN:AREA GRATICULE"`

SSCReen:IMAGe

Command   `:ACQuire:SSCReen:IMAGe {NORMal | INVert | MONochrome}`

Saves the screen image to disk normally, inverted, or in monochrome. INVert is the same as choosing Invert Background Waveform Color in the Specify Report Action for acquisition limit test dialog box.

Query   `:ACQuire:SSCReen:IMAGe?`

Returned Format   `[:ACQuire:SSCReen:IMAGe] {NORMal | INVert | MONochrome}<NL>`

Example   `10 OUTPUT 707;":ACQuire:SSCReen:IMAGE NORMAL"`

SWAVeform

Command   `:ACQuire:SWAVeform <source>, <destination> [,<filename>[, <format>]]`

Saves waveforms from a channel, function, TDR response, or waveform memory when the number of waveforms or samples as specified in the limit test is acquired. Each waveform source can be individually specified, allowing multiple channels, responses, or functions to be saved to disk or waveform memories. Setting a particular source to OFF removes any waveform save action from that source.

NOTE	This command operates on waveform and color grade gray scale data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

<source>	`{CHANnel<N>	FUNCtion<N>	WMEMory<N>	RESPonse<N>}`
<destination>	`{OFF	WMEMory<N>	DISK}`
<filename>	An ASCII string enclosed in quotes. If no filename is specified, a default filename will be assigned. The default filenames will be *AcqLimitChN_X*, *AcqLimitFnN_X*, *AcqLimitMemN_X* or *AcqLimitRspN_X*, where X is an incremental number assigned by the instrument. If a specified filename contains no path, the default path will be D:\User Files\waveforms. (C drive on 86100A/B instruments.)

NOTE	If the selected waveforms of consecutive limit tests are to be stored in individual files, omit the <filename> parameter. The waveforms will be stored in the default format (INTERNAL) using the default naming scheme.

<format>	`{TEXT [,YVALues	VERBose]	INTernal}` Where INTernal is the default format, and VERBose is the default format for TEXT.
Query	`:ACQuire:SWAVeform? <source>` The query returns the current state of the :ACQuire:SWAVeform command.
Returned Format	`[:ACQuire:SWAVeform]<source>, <destination> [,<filename>[,<format>]]<NL>`
Example	`10 OUTPUT 707;":ACQUIRE:SWAVEFORM CHAN1,OFF"`

SWAVeform:RESet

Command	`:ACQuire:SWAVeform:RESet` Sets the save destination for all waveforms to OFF. Setting a source to OFF removes any waveform save action from that source. This is a convenient way to turn off all saved waveforms if it is unknown which are being saved.
Example	`10 OUTPUT 707;":ACQuire:SWAVeform:RESet"`

7 Calibration Commands

This section briefly explains the calibration of the instrument. It is intended to give you and the calibration lab personnel an understanding of the calibration procedure and how the calibration subsystem is intended to be used. Also, this section acquaints you with the terms used in this chapter, help screens, and data sheets. A calibration procedure is included at the end of this chapter.

Mainframe Calibration

Mainframe calibration establishes calibration factors for the analyzer. These factors are stored in the analyzer's hard disk. You initiate the calibration from the Calibration menu or by sending the :CALibrate:FRAMe:STARt command. You should calibrate the analyzer mainframe periodically (at least annually), or if the ambient temperature since the last calibration has changed more than ±5°C. The temperature change since the last calibration is shown on the calibration status screen which is found under the Mainframe and Skew tab on the All Calibrations dialog box. It is the line labeled:

Cal ΔT _____ ·C.

KEYSIGHT
TECHNOLOGIES

Refer to the Service Guide for more details about the mainframe calibration.

NOTE   Let the instrument warm up for at least 1 hour before you calibrate it.

Module Calibration

Module calibrations enhance measurement precision by establishing calibration factors which compensate for imperfections in the measurement system, such as variations due to the ambient temperature. It is recommended you routinely perform this calibration for best measurement accuracy. Module calibration factors are valid only for the mainframe and slot in which the module was calibrated. You can install the module in the slots provided for Channels 1 and 2 or for Channels 3 and 4. Module calibrations do not require any external equipment setup. Always remove or disable all inputs to the module. However, inputs do not have to be removed from 83496A modules. The duration of the calibration is typically between 60 and 90 seconds.

A module calibration is recommended when:

· The instrument power has been cycled

· A module has been removed and then reinserted since the last calibration

· A change in the temperature of the module exceeds 5·C compared to the temperature of the last module calibration ($\Delta T > 5 \cdot C$)

· The time since the last calibration has exceeded 10 hours

You initiate a module calibration from the Modules tab on the All Calibrations dialog box or by sending the :CALibrate:MODule:VERTical command as shown in the following example.

```
DIM Prompt$[64]
OUTPUT 707;":CALIBRATE:MODULE:VERTICAL LMODULE"
OUTPUT 707;":CALIBRATE:SDONE?"
ENTER 707;Prompt$ <Disconnect everything from left module>
OUTPUT 707;":CALIBRATE:CONTINUE"
OUTPUT 707;":CALIBRATE:SDONE?"
ENTER 707;Prompt$ <Done>
```

NOTE   Let the Module Warm Up First. In order for the calibration to be accurate, the temperature of the module must reach equilibrium prior to performing the calibration.

NOTE   Reinserting the module into the mainframe can affect the electrical connections, which in turn can affect the calibration accuracy.

NOTE ΔT Value. A positive value for ΔT indicates how many degrees warmer the current module temperature is compared to the temperature of the module at the time of the last module calibration.

NOTE   Once the module calibration procedure is started, all access to the instrument's front panel is blocked, including the use of the Local button. Pressing Local during a module calibration will not place the instrument in local mode. The calibration must either be cancelled or finished before you can regain control to the instrument's front panel.

CAUTION   The input circuits can be damaged by electrostatic discharge (ESD). Avoid applying static discharges to the front-panel input connectors. Momentarily short the center and outer conductors of coaxial cables *prior* to connecting them to the front-panel inputs. *Before* touching the front-panel input connectors be sure to first touch the frame of the instrument. Be sure the instrument is properly earth-grounded to prevent buildup of static charge. Wear a wrist-strap or heel-strap.

Probe Calibration

The probe calibration is initiated from the Probe tab on the Calibrate/All Calibrations dialog or by sending either the :CALibrate:PROBe command or the :CHANnel<N>:PROBe:CALibrate command. The probe calibration allows the instrument to identify the offset and the gain, or loss, of specific probes that are connected to an electrical channel of the instrument. Those factors are then applied to the calibration of that channel. The instrument calibrates the vertical scale and offset based on the voltage measured at the tip of the probe or the cable input.

NOTE   For passive or non-identified probes, the instrument adjusts the vertical scale factors only if a probe calibration is performed.

Typically probes have standard attenuation factors, such as divide by 10, divide by 20, or divide by 100. If the probe being calibrated has a non-standard attenuation, the instrument will adjust the vertical scale factors of the input channel to match this attenuation.

CAUTION   The input circuits can be damaged by electrostatic discharge (ESD). Avoid applying static discharges to the front-panel input connectors. Momentarily short the center and outer conductors of coaxial cables *prior* to connecting them to the front-panel inputs. *Before* touching the front-panel input connectors be sure to first touch the frame of the instrument. Be sure the instrument is properly earth-grounded to prevent buildup of static charge. Wear a wrist-strap or heel-strap.

CANCel

Command   `:CALibrate:CANCel`

During a calibration, this command is equivalent to clicking Cancel on a displayed calibration message. Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.

Example   `10 OUTPUT 707;":CALIBRATE:CANCEL"`

CONTinue

Command   `:CALibrate:CONTinue`

During a calibration, this command is equivalent to clicking Continue on a displayed calibration message. Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.

Example   `10 OUTPUT 707;":CALIBRATE:CONTINUE"`

ERATio:DLEVel?

Query   `:CALibrate:ERATio:DLEVel? CHANnel<N>`

Returns the last dark level value for the specified channel, regardless of the current calibration status. If an extinction ratio calibration has been performed the returned value is the calibration result. If no calibration has been performed the default value of 0.0 is returned. <N> is an integer, from 1 to 4.

If the channel has multiple input connectors (for example, option 004 channels on 86115D modules), the reported dark level only applies to the currently selected connector. To query the dark level of a different connector, first select that connector with the command "CONNector" on page 142.

Returned Format   [:CALibrate:ERATio:DLEVel] <value><NL>

ERATio:STARt

Command   :CALibrate:ERATio:STARt CHANnel<N>

Starts an extinction ratio calibration. Before performing an extinction ratio calibration, display an eye diagram and adjust the vertical scale and offset so that the eye diagram uses the full display. Also, the dark level (the signal level when there is no input to the measurement) must be on the screen to be correctly measured. Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power. <N> is an integer, from 1 to 4.

If the channel being calibrated has multiple input connectors (for example, option 004 channels on 86115D modules), the Extinction Ratio calibration is only performed for the currently selected connector. Use the command "CONNector" on page 142 to select the desired connector prior to initiating the Extinction Ratio calibration.

ERATio:STATus?

Query   :CALibrate:ERATio:STATus? CHANnel<N>

Returns CALIBRATED or DEFAULTED indicating whether the ratio being used is the result of an extinction ratio calibration or is the factory default value. Once an extinction ratio calibration is performed, the query always returns DEFAULTED until the instrument power is cycled or the module is removed and then re-inserted into the instrument. <N> is an integer, from 1 to 4.

If the channel has multiple input connectors (for example, option 004 channels on 86115D modules), the calibration status reported only applies to the currently selected connector. To query the status of a different connector, first select that connector with the command "CONNector" on page 142.

Returned Format   [:CALibrate:ERATio:STATus] {CALIBRATED | DEFAULTED}<NL>

FRAMe:LABel

Command   :CALibrate:FRAMe:LABel <label>

Creates user notes, such as name/initials of the calibrator or special notes about the calibration. It accepts a string of up to 80 characters. The information is optional. <label> is a string, enclosed with quotes, with a maximum of 80. characters.

Query   :CALibrate:FRAMe:LABel?

The query returns the currently defined label for the frame.

Returned Format   [:CALibrate:FRAMe:LABel] <quoted string><NL>

FRAMe:STARt

Command   `:CALibrate:FRAMe:STARt`

Starts the annual calibration on the instrument mainframe. Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.

FRAMe:TIME?

Query   `:CALibrate:FRAMe:TIME?`

Returns the date, time and temperature of the last full-frame calibration.

Returned Format   `[:CALibrate:FRAMe:TIME] <time> <NL>`

<time> is in the format: DD MMM YY HH:MM <delta_temp>. <delta_temp> is the difference between the current temperature and the temperature when the last calibration was done. For example, <delta_temp> might be: –5C, 10C, or –12C.

MODule:LRESistance

Command   `:CALibrate:MODule:LRESistance <resistance_value>`

Sets the load resistance value used during module calibration of a TDR module. The accuracy of the calibration is improved by specifying the exact resistance value of the load that is connected to the TDR module during the calibration process. <resistance_value> is the resistance of the load from 47 to 53 ohm. The default value is the target value of 50 ohm.

Example   This example sets the load resistance value to 49.9 ohms.
`10 OUTPUT 707;":CALIBRATE:MODULE:LRESISTANCE 49.9"`

Query   `:CALibrate:MODule:LRESistance?`

Returned Format   `[:CALibrate:MODule:LRESistance] <resistance_value><NL>`

MODule:OCONversion?

Query   `:CALibrate:MODule:OCONversion? {LMODule | RMODule | CHANnel<N>},{WAVelength1 | WAVelength2 | USER}`

Returns the optical conversion (responsivity) of the specified channel at the specified wavelength. Wavelength 1 and Wavelength 2 are for factory-calibrated wavelengths. USER is the result of a user optical calibration. If LMOD or RMOD is specified for a dual optical module, the optical conversion of channel 1 (for LMOD) or channel 3 (for RMOD) will be returned. <N> is an integer, from 1 to 4. For 86108A Precision Waveform Analyzer modules, all forms of the query return the string UNCALIBRATED.

If the channel has multiple input connectors (for example, option 004 channels on 86115D modules), the optical conversion reported only applies to the currently selected connector. To query the optical conversion of a different connector, first select that connector with the command "CONNector" on page 142.

Returned Format   `[:CALibrate:MODule:OCONversion] {<value> | UNCALIBRATED}<NL>`

MODule:OPOWer

Command   `:CALibrate:MODule:OPOWer <optical_power_value>`

Sets the optical power level for an optical channel module calibration. Use only with modules that have an optical channel.

Example   `10 OUTPUT 707;":CALIBRATE:MODULE:OPOWER 500E-6"`

MODule:OPTical

Command   `:CALibrate:MODule:OPTical {CHANnel<N>}`

Initiates an O/E calibration on the selected channel. The selected channel must be an optical channel. <N> is an integer, from 1 to 4. If the channel being calibrated has multiple input connectors (for example, option 004 channels on 86115D modules), the O/E calibration is only performed for the currently selected connector. Use the command **"CONNector"** on page 142 to select the desired connector prior to initiating the O/E calibration.

Example
```
10 DIM Prompt $[64]
20 OUTPUT 707;":CALIBRATE:MODULE:OPTICAL CHAN1"
30 OUTPUT 707;":CALIBRATE:SDONE?"
40 ENTER 707;Prompt$ <Disconnect optical source form channel 1>
50 OUTPUT 707;":CALIBRATE:CONTINUE"
60 OUTPUT 707;":CALIBRATE:SDONE?"
70 ENTER 707;Prompt$ <Enter wavelength and power of optical source>
80 OUTPUT 707;":CALIBRATE:MODULE:OWAVELENGTH 1340E-9"
90 OUTPUT 707;":CALIBRATE:MODULE:OPOWER 500E-6"
100 OUTPUT 707;":CALIBRATE:CONTINUE"
110 OUTPUT 707;":CALIBRATE:SDONE?"
120 ENTER 707;Prompt$ <Connect optical source to channel 1>
130 OUTPUT 707;":CALIBRATE:CONTINUE"
140 OUTPUT 707;":CALIBRATE:SDONE?"
150 ENTER 707;Prompt$ <Done>
160 END
```

MODule:OWAVelength

Command   `:CALibrate:MODule:OWAVelength <wavelength>`

This command sets the optical wavelength for an optical channel calibration. This command should only be used for modules with an optical channel.

Example   `10 OUTPUT 707;":CALIBRATE:MODULE:OWAVELENGTH 1340E-9"`

MODule:STATus?

Query   `:CALibrate:MODule:STATus? {LMODule | RMODule}`

Returns the status of the module calibration (electrical and optical channels) and optical calibration (optical channels) as either CALIBRATED or UNCALIBRATED. It will return UNKNOWN if the module does not have calibration capability. Queries to modules with two electrical channels (including TDR modules) returns the status of module calibration only. Queries to modules with two optical channels will return the status of the module calibration, followed by the status of optical calibration of the first channel, followed by the status of the optical calibration of the second channel. For 86108A Precision Waveform Analyzer modules the argument is required, but both LMODule and RMODule arguments result in the identical query. The status of module calibration is returned followed by the status of the clock data recovery, followed by the status of the precision timebase.

If the channel has multiple input connectors (for example, option 004 channels on 86115D modules), the optical calibration status reported only applies to the currently selected connector. To query the optical calibration status of a different connector, first select that connector with the command **"CONNector"** on page 142.

Returned Format   *Dual-Electrical Channel Modules (including TDR modules)*
`[:CALibrate:MODule:STATus] <module calibration status><NL>`

Dual-Optical Channel Modules
```
[:CALibrate:MODule:STATus] <module calibration status>,<optical calibration
status channel 1>,<optical calibration status channel 2><NL>
```

86108A/B Module
```
[:CALibrate:MODule:STATus] <electrical channels calibration status>,<clock data
recovery status>,<precision timebase status><NL>
```

MODule:TIME?

Query   `:CALibrate:MODule:TIME? {LMODule | RMODule | CHANnel <N>}`

Returns the date and time at the last module calibration (channel) and the difference between the current channel temperature and the temperature of the channel when it was last calibrated. If there is not a module in the selected slot, the message "Empty Slot" is returned. <N> is an integer, from 1 to 4. Returned <time> is in the format: DD MMM YY HH:MM. Returned <delta_temp> is the difference between the current temperature and the temperature when the last calibration was done. For example, <delta_temp> might be –0.2 ·C or 2.4 ·C.

For 86108A Precision Waveform Analyzer modules the argument is required, but both LMODule, RMODule, and CHANnel arguments result in the identical query. Returns values for the module calibration followed by the clock data recovery calibration followed by the precision timebase calibration.

Returned Format   `[:CALibrate:MODule:TIME] <time> <delta_temp><NL>`

86108A/B Module
```
[:CALibrate:MODule:TIME] <time> <delta_temp>,<time> <delta_temp>,<time>
<delta_temp><NL>
```

The first <time><delta_temp> field returns values for the electrical channels calibration. The second <time><delta_temp> field returns values for the clock data recovery calibration. The third <time><delta_temp> field returns values for the precision timebase calibration.

MODule:VERTical

Command   `:CALibrate:MODule:VERTical {LMODule | RMODule | CHANnel<N> | SLOT<N> }`

Initiates a module calibration on a selected module, channel, or slot. For the CHANnel and SLOT arguments, the specified value should be either 1 (left module position) or 3 (right module position). Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.

For the 86108A Precision Waveform Analyzer module, calibrate the electrical channels by sending the LMOD, CHAN1, CHAN2, SLOT1, or SLOT2 argument. To calibrate the precision timebase along with the clock data recovery use the RMOD, CHAN3, CHAN4, SLOT3, or SLOT4 argument. Always calibrate the electrical channels before calibrating the precision timebase and clock data recovery.

Example (Sequence for modules other than 86108A/B)
```
10 OUTPUT 707;":CALIBRATE:MODULE:VERTICAL LMOD" ! or RMOD
20 OUTPUT 707;":CALIBRATE:MODULE:CONTINUE"
```

Example (Sequence for 86108A/B modules)
```
10 OUTPUT 707;":CALIBRATE:MODULE:VERTICAL LMOD"
10 OUTPUT 707;":CALIBRATE:MODULE:VERTICAL RMOD"
30 OUTPUT 707;":CALIBRATE:MODULE:CONTINUE"
```

OUTPut

Command   `:CALibrate:OUTPut <dc_value>`

Sets the dc level of the calibrator signal output through the front-panel CAL connector. <dc_value> is the dc level value in volts, adjustable from –2.0V to +2.0 Vdc.

Example   `10 OUTPUT 707;":CALIBRATE:OUTPUT 2.0"`

Query   `:CALibrate:OUTPut?`

Returns the current dc level of the calibrator output.

Returned Format   `[:CALibrate:OUTPut] <dc_value><NL>`

PROBe

Command   `:CALibrate:PROBe CHANnel<N>`

Starts the probe calibration for the selected channel. It has the same action as the command :CHANnel<N>:PROBe:CALibrate. For more information about probe calibration, refer to "Probe Calibration" on page 133. <N> is an integer, 1 through 4.

Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.

Example   `10 OUTPUT 707;":CALibrate:PROBe CHANnel1"`

RECommend?

Query   `:CALibrate:RECommend? {CHANnel<N>}`

Returns the current calibration recommendations of the instrument. There are seven comma-separated integers. A "1" indicates that a calibration is recommended, a 0 indicates that the calibration is either not required or not possible. These values match the calibration recommendations found in the All Calibrations dialog box. Open the Calibrate menu on the instrument display screen, then choose All Calibrations to open the All Calibrations dialog box. <N> is an integer, 1 through 4.For 86108A Precision Waveform Analyzer modules the CHANnel argument is required but ignored.

If the channel has multiple input connectors (for example, option 004 channels on 86115D modules), the extinction ratio and optical calibration status reported only applies to the currently selected connector. To query the extinction ratio and optical calibration status of a different connector, first select that connector with the command "CONNector" on page 142.

Restrictions   Required firmware revision 3.0 and above.

Example   `10 OUTPUT 707;":CALibrate:RECommend CHANnel1"`

Returned Format   `[:CALibrate:RECommend] <values><NL>`

Returned <values>
for Modules other
than 86108A/B
```
<Module/Vertical>,
<Mainframe/Horizontal>,
<ChannelN Extinction Ratio>,
<ChannelN Probe>,
<ChannelN Optical Wavelength1>,
<ChannelN Optical Wavelength2>,
<ChannelN Optical User-defined>
```

Returned <values>
for 86108A/B
Modules
```
<Electrical Channels/Vertical>,
<Clock Data Recovery>,
<Precision Timebase>,
<Mainframe/Horizontal>,
<Channel1 Probe>,
<Channel2 Probe>
```

SAMPlers

Command	`:CALibrate:SAMPlers {DISable	ENABle}`
	Enables or disables the sampler calibration in the module.
Example	`10 OUTPUT 707;":CALIBRATE:SAMPLERS ENABLE"`
Query	`:CALibrate:SAMPlers?`
Returned Format	`[:CALibrate:SAMPlers]{DISable	ENABle}<NL>`
Example	`20 OUTPUT 707;":CALIBRATE:SAMPLERS?"`

SDONe?

Query	`:CALibrate:SDONe?`
	Returns a string when the current calibration step is complete. The contents of the string returned indicates to the user the next step. Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.
Returned Format	`[:CALibrate:SDONe] <string><NL>`

SKEW

Command	`:CALibrate:SKEW {CHANnel<N>},<skew_value>`
	Sets the channel-to-channel skew factor for a channel. The numerical argument is a real number in seconds which is added to the current time base position to shift the position of the channel's data in time. Use this command to compensate for differences in the electrical lengths of input paths due to cabling and probes. <N> is an integer, from 1 to 4. <skew_value> is a real number, 0s to 100 μs. When pattern lock is active, <skew_value> is limited to 100 ns.

> **NOTE** In Jitter Mode, skew adjustments are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Example	`10 OUTPUT 707;":CALIBRATE:SKEW CHANNEL1,0.1s "`
Query	`:CALibrate:SKEW? {CHANnel<N>}`
	The query returns the current skew value.
Returned Format	`[:CALibrate:SKEW] <skew_value><NL>`

SKEW:AUTO

Command	`CALibrate:SKEW:AUTO`
	Sets the horizontal skew of multiple, active channels with the same bit rate, so that the waveform crossings align with each other. In addition, auto skew optimizes the instrument trigger level. Prior to auto skew, at least one channel must display a complete eye diagram in order to make the initial bit rate measurement.
Restrictions	NRZ Eye mode only.

NOTE	In Jitter Mode, skew adjustments are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

NOTE	Auto skew uses the current color grade measurement completion criterion (refer to "CGRade:COMPlete" on page 249). If auto skew fails to make the bit rate measurement or determine the time of the crossing points needed to compute the skew, it may be necessary to increase the color grade completion criterion. Increasing the value will increase the time for auto skew to complete.

STATus?

Query   `:CALibrate:STATus?`

Returns the calibration status of the instrument. These are nine comma-separated integers, with 1 or 0. A "1" indicates calibrated; a "0" indicates uncalibrated. The values that always return "0" are used to make the returned format compatible with the Agilent 83480A and 54750A.

NOTE	Use CALibrate:RECommend? to query for recommended calibrations.

Returned Format
```
[:CALibrate:STATus] <status><NL>
```

\<status\> for Modules other than 86108A/B
```
<Mainframe Calibration Status>,
<Channel1 Module Calibration>, 0,
<Channel2 Module Calibration>, 0,
<Channel3 Module Calibration>, 0,
<Channel4 Module Calibration>, 0
```

\<status\> for 86108A/B Modules
```
<Mainframe Calibration Status>,
<Channel1 Module Calibration>,
<Channel2 Module Calibration>,
<Clock Data Recovery Calibration>,
<Precision Timebase Calibration>
```

8 Channel Commands

The CHANnel subsystem commands control all vertical (Y axis) functions. You may toggle the channel displays on and off with the root level commands VIEW and BLANk, or with DISPlay.

BANDwidth

Command   `:CHANnel<N>:BANDwidth {HIGH | MID | LOW}`

Controls the channel bandwidth setting. When HIGH, the bandwidth is set to the upper bandwidth limit. When LOW, a lower bandwidth setting is selected in order to minimize broadband noise. For modules with three bandwidths, MID will select the center bandwidth. See the module section of the online Help for cutoff frequency specifications. <N> represents the channel number and is an integer 1 to 4.

Example   `10 OUTPUT 707;":CHANNEL1:BANDwidth HIGH"`

Query   `:CHANnel<N>:BANDwidth?`

Returned Format   `[:CHANnel<N>:BANDwidth] {HIGH | MID | LOW}<NL>`

KEYSIGHT
TECHNOLOGIES

Example   `20 OUTPUT 707;":CHANNEL1:BANDwidth?"`

CONNector

Command   `:CHANnel<N>:CONNector {A | B}`

Selects the optical input connector for modules that have more than one connector per channel. For example, the 86115D with option 004 has two connectors per channel. When a channel's alternate connector is selected, all the vertical settings of the current connector (for example scale, offset, wavelength, filter) will be applied to the newly selected connector. This command is not available for 86115D Option 002 modules, as these modules have only one connector available for each channel.

Restrictions   86115D Option 004 modules only. Software revision A.09.00 and above.

Example   `10 OUTPUT 707;":CHANNEL1:CONNector A"`

Query   `:CHANnel<N>:CONNector?`

Returned Format   `[:CHANnel<N>:CONNector] {A | B}<NL>`

Example   `20 OUTPUT 707;":CHANNEL1:CONNector?"`

DISPlay

Command   `:CHANnel<N>:DISPlay {{ON | 1} | {OFF | 0}}[,APPend]`

Turns the display of the specified channel on or off. <N> represents the channel number and is an integer 1 to 4. Use the APPend argument in Eye/Mask mode to turn on additional channels without turning off any other database signals that are currently on. Without the APPend parameter, all other database signals in the Eye/Mask mode would be turned off when turning a channel on.

Example   `10 OUTPUT 707;"CHANNEL1:DISPLAY ON"`

Query   `:CHANnel<N>:DISPlay?`

Returned Format   `[:CHANnel<N>:DISPlay] {1 | 0}<NL>`

Example   `20 OUTPUT 707;":CHANNEL1:DISPLAY?"`

DSKew

Command   `:CHANnel<N>:DSKew <skew value>`

On the 86118A-H01 module, sets the skew on the selected differential input channel. After setting the skew, to ensure amplitude accuracy, use the command "DSKew:LCALibrate" on page 143 to run a linearity calibration. <N> represents the channel number and is an integer 1 to 4.

Restrictions   86118A-H01 modules only.

Example   `10 OUTPUT 707;":CHANNEL1:DSKEW 5E-12"`

Query   `:CHANnel<N>:DSKew?`

Returned Format   `[:CHANnel<N>:DSKew] <skew value><NL>`

DSKew:AUTO

Command   `:CHANnel<N>:DSKew:AUTO`

Start automatic skew between two active input channels on an 86118A-H01 module. A differential signal must be connected to the two channels. Use the DSKew:AUTO:STEP command to set the step size used by the automatic skew adjustment. After performing an automatic skew, to ensure amplitude accuracy, use the command **"DSKew:LCALibrate"** on page 143 to run a linearity calibration. If the de-skew fails, error message –155, "Auto differential skew not performed" is displayed. <N> represents the channel number and is an integer 1 to 4.

Restrictions   86118A-H01 modules only.

Example   `10 OUTPUT 707;":CHANNEL1:DSKEW:AUTO"`

DSKew:AUTO:STEP

Command   `:CHANnel<N>:DSKew:AUTO:STEP <skew value>`
On the 86118A-H01 module, sets the step size used by the automatic skew adjustment. <N> represents the channel number and is an integer 1 to 4.

Restrictions   86118A-H01 modules only.

Example   `10 OUTPUT 707;":CHANNEL1:DSKEW:AUTO:STEP 2E-12"`

Query   `:CHANnel<N>:DSKew:AUTO:STEP?`

Returned Format   `[:CHANnel<N>:DSKew:AUTO:STEP] <step size><NL>`

DSKew:LCALibrate

Command   `:CHANnel<N>:DSKew:LCALibrate`
On the 86118A-H01 module, starts a linearity calibration. Perform a linearity calibration after a manual or automatic skew adjustment ensures amplitude accuracy. One a linearity calibration is started, use the :CAL:CONTinue, :CAL:CANCel and CAL:SDONe commands in **Chapter 7**, "Calibration Commands" to complete the linearity calibration. <N> represents the channel number and is an integer 1 to 4.

Restrictions   86118A-H01 modules only.

Example   `10 OUTPUT 707;":CHANNEL1:DSKEW:LCALIBRATE"`

FDEScription?

Query   `:CHANnel<N>:FDEScription?`
Returns the number of filters and a brief description of each filter for channels with one or more internal low-pass filters. The filter description is the same as the softkey label for the control used to select the active filter. <N> represents the channel number and is an integer 1 to 4.

Returned Format   `[:CHANnel<N>:FDEScription]<N><filter1_description>,<filter2_description>, ...`
`<filterN_description><NL>`
<filter_description> is XXX b/s or XXX b/s:N (depending on the module option), where XXX is bit rate of filter and N is filter order.

FILTer

Command   `:CHANnel<N>:FILTer {ON | 1 | OFF | 0}`

Controls an internal low-pass filter, if one is present, in the channel hardware. <N> represents the channel number and is an integer 1 to 4. When you turn the filter on, you can select which channel bandwidth setting you want to use. When you turn the filter off, the instrument sets the channel bandwidth to its default setting.

Example   10 OUTPUT 707;":CHANNEL1:FILTER ON"

Query   :CHANnel<N>:FILTer?

Returned Format   [:CHANnel<N>:FILTer] {1 | 0}<NL>

Example   20 OUTPUT 707;":CHANNEL1:FILTER?"

FSELect

Command   :CHANnel<N>:FSELect FILTer<filter_number>

Selects which filter is controlled by on/off for channels with more than one filter selection. <N> represents the channel number and is an integer 1 to 4. To query for a description of the filters, see the CHANnel:FDEScription query. <filter_number> is the filter number is an integer. In the Channel dialog box, filter number 1 is the first filter listed in the Filter box. See also CHANnel:FDEScription?

Example   10 OUTPUT 707;":CHANNEL1:FSELECT FILTER1"

Query   :CHANnel<N>:FSELect?

Returned Format   [:CHANnel<N>:FSELect]{FILT<filter_number>}<NL>

Example   20 OUTPUT 707;":CHANNEL1:FSELECT?"

OFFSet

Command   :CHANnel<N>:OFFSet <offset_value>

Sets the voltage that is represented at the center of the display for the selected channel. Offset parameters are probe and vertical scale dependent. For TDR and TDT applications, when the TDR stimulus is set to differential or common mode, the instrument will change offset to magnify offset. This command is used to set the magnify offset as well as the offset. <N> represents the channel number and is an integer 1 to 4.

NOTE   In Jitter Mode, channel scale and offset controls are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

<offset_value> is offset value at center screen. In TDR mode or with optical channels (any situation with positive values only), the offset value is positioned two divisions up from the bottom of the graticule as shown in the following picture. The value is usually expressed in volts, but could be in other measurement units, such as amperes, if you have specified other units using the CHANnel:UNITs command.

Alternate offset value position
(two divisions up)
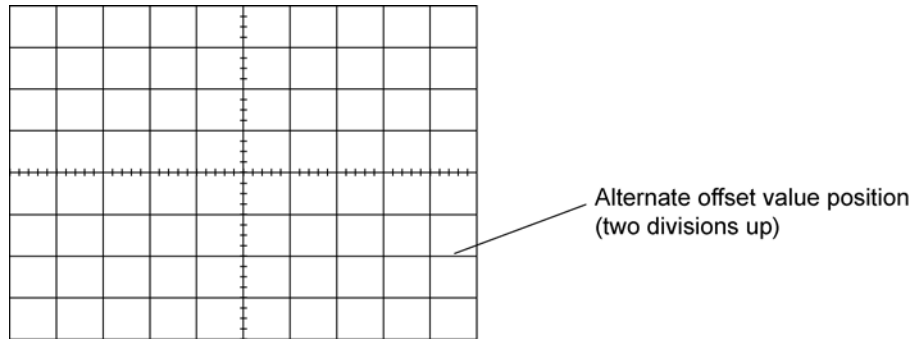
Example   This example sets the offset for channel 1 to 0.125 in the current measurement units.
```
10 OUTPUT 707;":CHANNEL1:OFFSET 125E-3"
```

Query   :CHANnel<N>:OFFSet?

The query returns the current offset value for the specified channel.

Returned Format   [CHANnel<N>:OFFSet] <offset value><NL>

Example   This example places the offset value of the specified channel in the string variable, Offset$.
```
10 OUTPUT 707;"SYSTEM:HEADER OFF"
20 OUTPUT 707;"CHANNEL1:OFFSET?"
30 ENTER 707;Offset
```

PROBe

Command   :CHANnel<N>:PROBe <attenuation factor>[,{RATio | DECibel}]

Sets the channel attenuation factor and units. It provides the equivalent function of the Attenuation Factor setting under the Setup menu's Channel command. The default attenuation factor is 1:1 and the default units are ratio. When the TDR stimulus is set to differential or common mode, the instrument will change offset to magnify offset. This command is used to set the magnify offset as well as the offset. <N> represents the channel number and is an integer 1 to 4.

Query   :CHANnel<N>:PROBe?

Returned Format   [:CHANnel<N>:PROBe] <attenuation factor>, {RATio | DECibel}<NL>

PROBe:CALibrate

Command   :CHANnel<N>:PROBe:CALibrate

Starts the probe's calibration for the selected channel. It has the same action as the command :CALibrate:PROBe CHANnel<N>. For more information about probe calibration, refer to "Probe Calibration" on page 133. <N> represents the channel number and is an integer 1 to 4. Whenever a calibration message is displayed on the instrument, send the :CALibrate:CONTinue, :CALibrate:CANCel, or :CALibrate:SDONE commands. Sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.

Example   `10 OUTPUT 707;":CHANNEL1:PROBE:CALIBRATE"`

PROBe:SELect

Command   :CHANnel<N>:PROBe:SELect <probe_id>[,<meas_mode>]

Selects an AutoProbe interface probe used in conjunction with the Agilent N1022A or N5477A probe adapter. The probes that are currently supported by this command are the Agilent single-ended/differential 1131A, 1132A, 1134A, 1168A, and 1169A probes, single-ended 1152A, 1156A, 1157A, and 1158A probes, and the N2800-series InfiniiMax III probes. <N> represents the channel number and is an integer 1 to 4. If you elect to use an AutoProbe style probe that is not in the supported probe list, select one of the probes from the supported list that is closest in type to your unspecified probe. This command is not available for TDR/TDT measurements. An error condition will occur if an AutoProbe is not connected to a channel.

<probe_id> is used to select the AutoProbe type: {P2800A | P2801A | P2802A | P2803A | P1131A | P1132A | P1134A | P1152A | P1156A | P1157A | P1158A | P1168A | P1169A}.

The optional <meas_mode> parameter is used to set the measurement mode. The default measurement mode is Single ENDed. Use the DIFFerential parameter for the differential probes to measure differential signals: {SENDed | DIFFerential}.

Restrictions   86100D or 86100C (software revision A.10.80 and above) for the P2800A, P2801A, P2802A, and P2803A arguments.

Example   The following example selects the 1134A in differential mode on channel 2.
```
10 OUTPUT 707;":CHANNEL2:PROBE:SELECT P1134A,DIFFERENTIAL"
```

Query   :CHANnel<N>:PROBe:SELect?

This query returns the AutoProbe type that is attached to the specified channel. If the type of probe that is attached is a passive probe or not an AutoProbe, an error will be returned.

Returned Format   [:CHANnel<N>:PROBe:SELect] <probe_id>, {SEND | DIFF}<NL>

Example   The following example places the current probe type in the string variable, Probe$.
```
10 DIM Probe$[50] !Probe variable
20 OUTPUT 707;":CHANNEL2:PROBE:SELECT?"
30 ENTER 707;Probe$
```

RANGe

Command   :CHANnel<N>:RANGe <range_value>

Defines the full-scale vertical axis of the selected channel. It sets up acquisition and display hardware to display the waveform at a given range scale. The values represent the full-scale deflection factor of the vertical axis in volts. These values change as the probe attenuation factor is changed. For TDR and TDT applications, when the TDR stimulus is set to differential or common mode, or when OHM, REFLect, or GAIN units are selected, the instrument will change scale to magnify scale. This command is used to set the magnify range as well as the range. <N> represents the channel number and is an integer 1 to 4.

> **NOTE**   In Jitter Mode, channel scale and offset controls are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

<range_value>   Full-scale voltage of the specified channel number.

Example   This example sets the full-scale range for channel 1 to 500 mV.
```
10 OUTPUT 707;":CHANNEL1:RANGE 500E-3"
```

Query   :CHANnel<N>:RANGe?

The query returns the current full-scale vertical axis setting for the selected channel.

Returned Format   [:CHANnel<N>:RANGe]<range value><NL>

Example   This example places the current range value in the number variable, Setting.
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":CHANNEL1:RANGE?"
30 ENTER 707;Setting
```

SCALe

Command   `:CHANnel<N>:SCALe <scale_value>`

Sets the vertical scale, or units per division, of the selected channel. This command is the same as the front-panel channel scale. For TDR and TDT applications, when the TDR stimulus is set to differential or common mode, the instrument will change scale to magnify scale. This command is used to set the magnify scale as well as the scale. <N> represents the channel number and is an integer 1 to 4.

NOTE   In Jitter Mode, channel scale and offset controls are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

<scale_value>   Vertical scale of the channel in units per division.

Example   This example sets the scale value for channel 1 to 500 mV.
```
10 OUTPUT 707;":CHANNEL1:SCALE 500E-3"
```

Query   `:CHANnel<N>:SCALe?`

The query returns the current scale setting for the specified channel.

Returned Format   `[:CHANnel<N>:SCALe] <scale value><NL>`

Example   This example places the current scale value in the number variable, Setting.
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":CHANNEL1:SCALE?"
30 ENTER 707;Setting
```

TDRSkew

Command   `:CHANnel<N>:TDRSkew <percent> [%]`

Sets the TDR skew for the given channel. The TDR skew control moves the TDR step relative to the trigger position. The control may be set from –100 to 100 percent of the allowable range. This command is only applicable to TDR channels. This command is enabled only if a stimulus is currently active and if the module has differential capability. <N> represents the channel number and is an integer 1 to 4 followed by an optional A or B identifying which of two possible channels in the slot is being referenced.

<percent>   A number between –100 and 100, used to set the step position.

Example   The following example sets the TDR skew for channel 1 to 20%.
```
10 OUTPUT 707;":CHANNEL1:TDRSKEW 20"
```

Query   `:CHANnel<N>:TDRSkew?`

The query returns the current TDR skew setting for the specified channel.It returns the TDR skew value in percent of allowable range from –100 to 100 percent. This command is only applicable to TDR channels. The returned format is a real number.

Returned Format   `[:CHANnel<N>:TDRSkew] <value><NL>`

UNITs

Command   `:CHANnel<N>:UNITs {VOLT | OHM |AMPere | REFLect | WATT | UNKNown}`

Sets the transducer units in Oscilloscope and Eye/Mask modes. In TDR/TDT mode this command sets the channel units (VOLT, OHM, REFLect). <N> represents the channel number and is an integer 1 to 4.

Query   `:CHANnel<N>:UNITs?`

Returned Format   `[:CHANnel<N>:UNITs] {VOLT | OHM | REFLect | AMPere | WATT | UNKNown}<NL>`

UNITs:ATTenuation

Command   `:CHANnel<N>:UNITs:ATTenuation <attenuation>`

Sets the transducer conversion factor. It provides the equivalent function of the Transducer Conversion Factors Gain setting under the Setup menu's Channel command. This command is disabled for TDR channels and destinations channels for TDR/TDT measurements. <N> represents the channel number and is an integer 1 to 4.

Query   `:CHANnel<N>:UNITs:ATTenuation?`

Returned Format   `[:CHANnel<N>:UNITs:ATTenuation] <attenuation><NL>`

UNITs:OFFSet

Command   `:CHANnel<N>:UNITs:OFFSet <offset>`

Sets the transducer offset. It provides the equivalent function of the Transducer Conversion Factors Offset setting under the Setup menu's Channel command. This command is disabled for TDR channels and destinations channels for TDR/TDT measurements. <N> represents the channel number and is an integer 1 to 4.

Query   `:CHANnel<N>:UNITs:OFFSet?`

Returned Format   `[:CHANnel<N>:UNITs:OFFSet] <offset><NL>`

WAVelength

Command   `:CHANnel<N>:WAVelength {WAVelength1 | WAVelength2 | WAVelength3 | USER}`

Sets the wavelength selection for optical channels. Modules can support one, two, or three factory-defined wavelengths. The module will have one factory calibration for each factory-defined wavelength. Invoke these calibrations using WAV1, WAV2, or WAV3. One user-defined wavelength may also be defined via the Channel Calibrate menu. The USER selection is only valid if this user-defined calibration has been performed. The calibration will request the wavelength that the USER choice corresponds to. This command will also recognize W1310 as an equivalent for WAVelength1 and W1550 for WAVelength2, for compatibility with the Agilent 83480A/54750A.

<N> represents the channel number and is an integer 1 to 4.

When an unsupported wavelength is specified, the instrument ignores the command. For example, for modules with two factory-defined wavelengths, WAV3 will not change the current wavelength selection.

Restrictions   For WAV3 argument, software revision A.04.10 and above required.

Query   `:CHANnel<N>:WAVelength?`

The query returns the currently selected wavelength for the channel.

Returned Format   `[:CHANnel<N>:WAVelength] {WAV1 | WAV2 | WAV3 | USER} <cal wavelength><NL>`

The returned <cal wavelength> string can be one of four values: 8.50E-007, 1.310E-006, 1.550E-006, or a user-defined value.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":CHANnel1:WAVELENGTH?"
30 ENTER 707;Setting
```

9 Clock Recovery Commands

The Clock RECovery (CREC) subsystem commands control the clock recovery modules. This includes setting data rates, as well as querying locked status and signal present conditions. Refer to Table 28 on page 152 for a listing of which subsystem commands work with each module. Refer to Table 29 on page 153 for a listing of available data rates for each module.

83491/2/3/4 Modules

Agilent 83491A modules have electrical inputs, 83492A have multimode optical inputs, and 83493A and 83494A modules have single-mode optical inputs. Each of these modules recovers clock signals at specific rates as listed in Table 29 on page 153. Use the RATE command to select the module's data rate so that it matches the input signal. All of these modules automatically lock on input signals, provided that they are set to the correct data rate. Use the LOCKed? query to determine if the module is locked on the signal. The loop bandwidth for each module is fixed. For the external output, the loop bandwidth is 4 to 5 MHz. On 83491/2/3A modules, the internal triggering loop bandwith is 50 to 70 kHz; on 83494A modules, it is 90 kHz. For 83492/3/4A modules, use the SPResent to check if an optical signal is detected by the module.

KEYSIGHT
TECHNOLOGIES

83495A Module

Agilent 83495A modules provide both optical and electrical clock recovery for all rates from 9.953 Gb/s to 11.32 Gb/s. Use the INPut command to select the optical or electrical input. Use the RATE command to select the module's data rate. On Option 200 modules, you can select a continuous rate range between 9.953 Gb/s to 11.32 Gb/s. The module will lock on any valid signal within this range. As with 83492/3/4A modules, this module automatically locks on the input signal, provided that the module is set to the correct data rate. Use the LOCKed? query to determine if the module is locked on the signal. Unlike 83492/3/4A modules, the SPResent command can not be used to check if an optical signal is detected. Use the LBANdwidth command to select from two loop bandwidth settings: 300 kHz and 4 MHz.

83496A/B Modules

Agilent 83496A/B modules provide both optical and electrical clock recovery selected by the INPut command. They have continuous, unbanded tuning from 50 Mb/s to 7.10 Gb/s (14.2 Gb/s, Option 200). Specify the data rate with the CRATe command rather than the RATE command as with other modules. Although the module accepts the RATE command for compatibilty with existing programs, it is recommended that you use the CRATe command. Unlike 83492/3/4A modules, the SPResent command can not be used to check if an optical signal is detected.

Because these modules do not provide automatic locking, you must issue the LOCK command to establish lock and to reestablish lock whenever a setup parameter changes (for example input port or trigger on data), the data rate changes, or the signal parameters change (for example, edge density). Use the LOCKed? query to determine if the module is locked on the signal. If the module loses lock, the trigger becomes asynchronous with the data and the instrument will not display a correctly triggered waveform. Use the TDENsity query to return the edge density of the data signal.

Table 28   Command Compatibility with Module

Command	83491A	83492A	83493A	83494A	83495A	83496A/B	83496A/B Option 300	86108A/B
ARELock						●	●	●
ARELock:CANCel						●	●	●
ARELock:STATe?						●	●	●
CLBandwidth							●	●
CRATe						●	●	●
CFRequency?								●
INPut					●	●	●	●
LBANdwidth					●	●	●*	●
LBWMode							●	●
LOCKed?	●	●	●	●	●	●	●	●
LSELect								●
LSELect:AUTomatic								●
ODRatio						●	●	●

Table 28 Command Compatibility with Module

Command	83491A	83492A	83493A	83494A	83495A	83496A/B	83496A/B Option 300	86108A/B
ODRatio:AUTO						●	●	●
PEAKing?								●
RATE	●	●	●	●		● [†]	● [b]	● [b]
RDIVider							●	●
RELock						●	●	●
SOURce:AUTOdetect								●
SPResent?	●	●	●	●				
TDENsity?						●	●	●
T2TFrequency?								●

* CONTinuous query only.

† For backwards compatibility. In new programs, use CRATe instead.

Standard 83496A/B modules have two loop bandwidth settings that are selected using the LBANdwidth command. The low bandwidth setting is 30 kHz (< 1 Gb/s data rate) or 270 kHz (≥1 Gb/s data rate). The high bandwidth setting is 1500 kHz. On Option 300 modules, you can specify any loop bandwidth between the range of 30 kHz to 10 MHz using the CLBandwidth command. Or, on Option 300 modules, use the LBWMode command to configure the module to automatically select the loop bandwidth based on data rate and data-rate divide ratio (RDIVider command).

Use the ODRatio and ODRatio:AUTO commands to specify the divide ratio that is applied to the module's front-panel Recovered Clock Output.

86108A/B

When sending a clock recovery command to the 86108A/B, only channel one can be specified for the subsystem, for example CRECOVERY1:LOCKED?. Channel 3 is not a valid selection as it is with the clock recovery modules.

Table 29 Module Data Rates

Rate (Mb/s)	83491	83492	83493	83494	83494 Option 103	83494 Option 106	83494 Option 107	83495	83496A/B	83496A/B Option 200
Trigger on data	●	●	●	●	●	●	●	●	●	●
155.52	●	●	●	●	●	●	●		●	●
622.08	●	●	●	●	●	●	●		●	●
1062.50	●	●							●	●
1250.00	●	●	●						●	●

Table 29   Module Data Rates

Rate (Mb/s)	83491	83492	83493	83494	83494 Option 103	83494 Option 106	83494 Option 107	83495	83496A/B	83496A/B Option 200
2125.00	●	●							●	●
2488.32	●	●	●	●	●	●	●		●	●
2500.00	●	●	●						●	●
2666.06						●	●		●	●
9953.28				●				●		●
10312.50					●			●		●
10664.23						●		●		●
10709.225							●	●		●
9.953 Gb/s– 11.32 Gb/s								●		
9.953 Gb/s– 14.2 Gb/s										●
Continuous									●	●

ARELock

Command	`:CRECovery{1	3}:ARELock {ON	1	OFF	0}`
	Enables or disables automatic data-rate locking.
Restrictions	83486A/B and 86108A/B modules. Software revision A.08.00 and above.
Example	`10 OUTPUT 707; ":CRECOVERY1:ARELOCK ON"`
Query	`:CRECovery{1	3}:ARELock?`
Returned Format	[:CRECovery{1	3}:ARELock] {ON	1	OFF	0}<NL>

ARELock:CANCel

Command	`:CRECovery{1	3}:ARELock:CANCel`
	During automatic data-rate locking, this command is equivalent to clicking Cancel on a displayed Clock Recovery Lock Lost message box. Whenever a this message is displayed on the instrument, sending any other command, including *OPC, disrupts the instrument forcing you to cycle instrument power.
Restrictions	83486A/B and 86108A/B modules. Software revision A.08.00 and above.
Example	`10 OUTPUT 707; ":CRECOVERY1:ARELOCK:CANCEL"`

ARELock:STATe?

Query	`:CRECovery{1	3}:ARELock:STATe?`

Queries the state of automatic data-rate locking. Returns NLOCking when the 86108A/B is not attempting to auto relock, LLM when locking the left module (for Channel 1), and LRM when locking the right module (for Channel 2).

Restrictions	83486A/B and 86108A/B modules. Software revision A.08.00 and above.
Example	`10 OUTPUT 707; ":CRECOVERY1:ARELOCK:STATE?"`
Returned Format	[:CRECovery{1 | 3}:ARELock:STATe] {NLOC | LLM | LRM}<NL>

CLBandwidth

Command	`:CRECovery{1	3}:CLBandwidth <bandwidth>`

This 83496A/B Option 300 and 86108A/B command sets or queries the module's loop bandwidth. You must issue the LBWMode FIXed command before using the CLBandwidth command. A settings conflict error is reported if the module's loop bandwidth mode is set to be rate dependent (RDEPendent). Refer to "LBWMode" on page 157. The loop bandwidth can be any bandwidth within 30 kHz to 20 MHz specified to 3 significant digits. The default setting is 60 kHz.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions	83496A/B Option 300 and 86108A/B modules. Software revision A.04.20 and above.
Example	`10 OUTPUT 707; ":CRECOVERY1:CLBANDWIDTH 1.7E6"`
Query	`:CRECovery{1	3}:CLBandwidth?`
Returned Format	[:CRECovery{1 | 3}:CLBandwidth] <bandwidth><NL>

CRATe

Command	`:CRECovery{1	3}:CRATe <data_rate>`

This 83496A/B and 86108A/B command sets or queries the module's data rate setting. Although the command "RATE" on page 160 can be used, use the preferred CRATe command in all new programs. The data rate for standard 83496A/B modules ranges from 50 Mb/s to 7.10 Gb/s. The data rate for Option 200 modules ranges from 50 Mb/s to 14.20 Gb/s. The data rate can be specified to 6 significant digits. The default setting is 2.488 Gb/s. On 86108A/B modules, only channel one can be specified in the command (:CRECovery1:CRATe). Specifying channel three (:CRECovery3:CRATe) results in the returned string "Not Present".

Restrictions	83486A/B and 86108A/B modules. Software revision A.04.20 and above.
Example	`10 OUTPUT 707; ":CRECOVERY1:CRATE 4.25E9"`
Query	`:CRECovery{1	3}:CRATe?`
Returned Format	[:CRECovery{1 | 3}:CRATe <data_rate><NL>

CFRequency?

Query	`:CRECovery1:CFRequency? {factor}`

86108A/B query that returns the frequency of the recovered data clock. The optional argument, factor, respresent the number of time periods to wait while performing the measurement. Each time period is approximately 20 milliseconds. The default value is 1.0.

Restrictions	86108A/B modules. Software revision A.08.00 and above.

Example   `10 OUTPUT 707; ":CRECOVERY1:CFREQUENCY?"`

Returned Format   `[:CRECovery1:CFRequency] <clock_frequency><NL>`

INPut

Command   `:CRECovery{1 | 3}:INPut{ELECtrical | OPTical | DIFFerential | EINVerted | AUXiliary}`

Selects the clock recovery input on 83495A, 83496A/B, and 86108A/B modules. On 83495A modules, OPTical is the default setting. On 83496A/B modules, ELECtrical is the default setting. The arguments DIFFerential and EINVerted (electrical inverted), are available on 83496A/B and 86108A/B modules only. The DIFFerential argument is the default argument for 86108A/B modules.

On 86108A/B-400 modules, the AUXiliary argument provides control for the front-panel AUX input. The AUX input can provide improved performance in situations of marginal input signals, measurements on data rates above the 14.2 Gb/s channel input limits, and phase noise measurements. To learn more about using the AUX input, refer to the instrument's help system.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions   83495A, 83496A/B, and 86108A/B modules. Software revision A.03.10 and above for 83495A module. Software revision A.04.20 and above for support of 83496A modules. For the AUX argument, an 86108A/B-400 module and instrument Firmware Revision 10.0 and above are required.

Example   `10 OUTPUT 707;":CRECOVERY1:INPUT ELECTRICAL"`

Query   `:CRECovery{1 | 3}:INPut?`

Returned Format   `[:CRECovery{1 | 3}:INPut] {ELECtrical | OPTical | DIFFerential | EINVerted | AUXiliary}<NL>`

LBANdwidth

Command   `:CRECovery{1 | 3}:LBANdwidth {BW270KHZ | BW300KHZ | BW1500KHZ | BW4MHZ | CONTinuous}`

Sets the loop bandwidth on 83495A, 83496A/B, and 86108A/B modules to a value as listed in Table 30 on page 157. The default setting is 300 kHz for 83495A modules, 270 kHz for 83496A/B modules, and 1.5 MHz for 86108A/B modules. The CONTinuous argument (83496A/B Option 300 and 86108A/B only) can be returned in queries but can not be sent in a command string. CONTinuous is returned whenever the loop bandwidth of an 83496A/B Option 300 or an 86108A/B module is set to a value other than the LBANdwidth standard values. When the CONTinuous argument is returned, use the CLBandwidth command to query the actual value. Refer to "CLBandwidth" on page 155.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Do not use this command with 83496A/B Option 300 modules. Instead, use the command "CLBandwidth" on page 155.

Restrictions   83495A, 83496A/B, and 86108A/B modules. 83495A modules , software revision A.03.10 and above. 83496A/B modules (except Option 300), software revision A.04.20 and above.

Example   `10 OUTPUT 707;":CRECOVERY1:LBANDWIDTH BW4MHZ"`

Query   `:CRECovery{1 | 3}:LBANDWIDTH?`

Returned Format   `[:CRECovery{1 | 3}:LBANdwidth] {BW270KHZ | BW300KHZ | BW1500KHZ | BW4MHZ | CONTinuous}<NL>`

Table 30   Valid Loop Bandwidth Arguments Versus Modules

Arguments	83495A	83496A/B	83496A/B (Not Opt. 300)
BW270KHZ		● [*,†]	● [a]
BW300KHZ	● [a]		●
BW1500KHZ		● [a,‡]	●
BW4MHZ	●		●
CONTinuous [**]			

* Default data rate.

† Default and only selection for data rates below 1 Gb/s.

‡ Default ≥1 Gb/s. Unavailable for data rates below 1 Gb/s.

**The CONTinuous argument is returned in queries and can not be used to set the bandwidth.

LBWMode

Command	`:CRECovery{1	3}:LBWMode {FIXed	RDEPendent}`

This 83496A/B Option 300 and 86108A/B command sets or queries the module's loop bandwidth entry mode. When FIXed is specified, the loop bandwidth value can be entered using the CLBandwidth command. When RDEPendent (rate dependent) is specified, the loop bandwidth is indirectly set by the data rate and the data-rate divide ratio (RDIVider command). The loop bandwidth can not be entered when the module is in the RDEPendent mode.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions	83496A/B Option 300 and 86108A/B modules. Software revision A.04.20 and above.
Example	`10 OUTPUT 707;":CRECOVERY1:LBWMODE FIXED"`
Query	`:CRECovery{1	3}:LBWMode?`
Returned Format	[:CRECovery{1	3}:LBWMode] {FIXed	RDEPendent}<NL>

LOCKed?

Query	`:CRECovery{1	3}:LOCKed?`

This 83491/2/3/4/5/6A/6B and 86108A/B query returns the locked status of the clock recovery module. Locked status returns 1, unlocked status returns 0. When a clock rate is selected on 83491/2/3/4/5A modules, unlocked status indicates that clock recovery cannot be established and trigger output to the mainframe is disabled. In bypass mode (trigger on data), status is always 0 and trigger output to the mainframe is not disabled. For 83495A modules, status is still locked or unlocked depending on clock recovery state. For 83496A/B modules, the trigger output to the mainframe is *not* disabled when an unlocked condition exists. On 86108A/B modules, only channel one can be specified in the command (:CRECovery1:LOCKed?). Specifying channel three (:CRECovery3:LOCKed?) results in the returned string "Not Present".

Returned Format	`[:CRECovery{1	3}:LOCK] {1	0}<NL>`

Example   `10 OUTPUT 707;":CRECOVERY1:LOCKED?"`

LSELect

Command   `:CRECovery{1}:LSELect {LOOP<N>}`

This 86108A/B command selects the Type-2 loop transition frequency (peaking), where N is an integer that specifies the setting:

- N = 1 selects 12 kHz (available for all loop bandwidths)
- N = 2 selects 280 kHz (available for loop bandwidths > 600 kHz)
- N = 3 selects 640 kHz (available for loop bandwidths > 1.6 MHz)
- N = 4 selects 1.3 MHz (available for loop bandwidths > 4.5 MHz)

In normal operation, the Type-2 transition frequency is automatically coupled to the CDR loop bandwidth and provides the desired loop characteristic for most measurements. Use "LSELect:AUTomatic" on page 159 to turn off automatic coupling. Use "T2TFrequency?" on page 163 to query the current Type-2 loop transition frequency. Use "PEAKing?" on page 159 to query the loop gain in dB.

Clock recovery extracts a clock from the incoming signal and provides the DCA with a trigger that is synchronous with the data. The clock recovery loop bandwidth primarily determines how well the recovered clock tracks low-frequency jitter on the input signal. Some signals have very large low-frequency jitter from either extremely dirty clocks or intentional modulated clocks such as found in SSC (spread spectrum clocking). In this case the 86108A/B clock recovery system provides additional control of the loop dynamics by allowing the user to select the Type-2 transition frequency of the loop. The Type-2 transition frequency indicates the frequency below which the second integrator in the loop starts to provide extra gain. Increasing this frequency provides additional loop gain and improves the tracking of the loop. The following figure shows the jitter multiplier as a function of jitter frequency for a loop-bandwidth setting of 5 MHz and various settings of transition frequency. This multiplier is the magnitude of the observed jitter transfer function (OJTF). This additional tracking also increases the peaking in the closed-loop jitter transfer function (JTF).
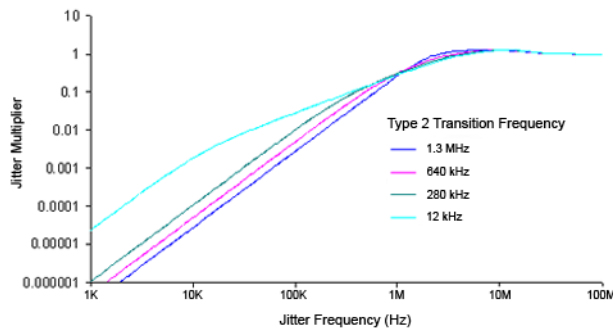
Figure 8   OJTF for 5 MHz LBW vs. Type-2 Transition Frequency

Restrictions   86108A/B modules. Software revision A.08.00 and above.

Example   `10 OUTPUT 707; ":CRECOVERY1:LSELect LOOP2"`

Query   `:CRECovery1:LSELect?`

Returned Format   `[:CRECovery1:LSELect] LOOP<N><NL>`

LSELect:AUTomatic

Command	`:CRECovery{1}:LSELect:AUTomatic {ON	1 OFF	0}`

This 86108A/B command turns on and off the coupling of the Type-2 transition frequency to the CDR loop bandwidth. **Refer to "LSELect" on page 158** for a description of this feature.

Restrictions	86108A/B modules. Software revision A.08.00 and above.
Example	`10 OUTPUT 707; ":CRECOVERY1:LSEL:AUT ON"`
Query	`:CRECovery{1}:LSELect:AUTomatic?`
Returned Format	[:CRECovery{1}:LSELect:AUTomatic] {ON	1 OFF	0}<NL>

ODRatio

Command	`:CRECovery{1	3}:ODRatio <divide_ratio>`

This 83496A/B and 86108A/B command sets or queries the output clock divide ratio. This determines the data rate at the front-panel recovered clock output. The ratio can be set to a value of 1, 2, 4, 8, 16, or 32 for divided output and –2, –4, or –8 for a multiplied clock rates of 2x, 4x, and 8x respectively. Sending this command while the output divider is set to auto (Refer to **"ODRatio:AUTO"** on page 159), results in a settings conflict error.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions	83496A/B and 86108A/B modules. Software revision A.04.20 and above. Output devide ration arguments 32, –2, –4, and –8 require software revision A.10.50.
Example	`10 OUTPUT 707; ":CRECOVERY1:ODRATIO 2"`
Query	`:CRECovery{1	3}:ODRatio?`
Returned Format	[:CRECovery{1	3}:ODRatio] <divide_ratio><NL>

ODRatio:AUTO

Command	`:CRECovery{1	3}:ODRatio:AUTO {ON	1 OFF	0}`

This 83496A/B and 86108A/B command enables or disables the module's capability to automatically set the divide ratio for the front-panel recovered clock output. With auto on, the instrument automatically selects an output divide ratio setting to 1:1 for frequencies equal to or less than 7.1 GHz or 1:2 for frequencies greater than 7.1 GHz.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions	83496A/B and 86108A/B modules. Software revision A.04.20 and above.
Example	`10 OUTPUT 707; ":CRECOVERY1:ODRATIO:AUTO ON"`
Query	`:CRECovery{1	3}:ODRatio:AUTO?`
Returned Format	[:CRECovery{1	3}:ODRatio:AUTO] {ON	1 OFF	0}<NL>

PEAKing?

Query	`:CRECovery1:PEAKing?`

Queries the loop gain in dB for the current Type-2 transition frequency. Refer to **"LSELect"** on page 158 for a description of this feature.

Restrictions   86108A/B modules. Software revision A.08.00 and above.

Example   `10 OUTPUT 707; ":CRECOVERY1:PEAKING?"`

Returned Format   `[:CRECovery1:PEAKing] <loop_gain><NL>`

RATE

Command   `:CRECovery{1 | 3}:RATE {TOData | R155 | R622 | R1062 | R1250 | R2125 | R2488 | R2500 | R2666 | R9953 | R10312 | R10664 | R10709 | RANGE10G}`

This 83491/2/3/4/5/6A/B and 86108A/B command sets the clock recovery module's data rate. The available rates for each module, with associated command arguments, are listed in Table 31 on page 161. RATE parameters are nominal and reflect front-panel labels and not actual data rates. The TOData argument selects triggering on the data. Once TOData is used, you must specify a different rate to turn off triggering on data. On 86108A/B modules, only channel one can be specified in the command (:CRECovery1:RATE). Specifying channel three (:CRECovery3:RATE) results in the returned string "Not Present". Although this command will work with 83496A/B and 86108A/B modules, new programs should use the command **"CRATe"** on page 155.

Restrictions   83491/2/3/4/5/6A/B and 86108A/B modules. The CONTinous query response is only returned by 83496A/B and 86108A/B modules and requires software revision 4.20 and above.

Example   `10 OUTPUT 707;":CRECOVERY1:RATE R2488"`

Query   `:CRECovery{1 | 3}:RATE?`

Returned Format   The CONTinuous query response appears in queries only and can not be sent in a command string. CONTinuous is returned whenever the data rate of an 83496A/B or 86108A/B module is not one of the standard values set using the CRECovery:RATE command. If the CONTinuous argument is returned, use the CRECovery:CRATE command to query the actual value. Refer to **"CRATe" on page 155**.

`[:CRECovery{1 | 3}:RATE] {TOData | R155 | R622 | R1062 | R1250 | R2125 | R2488 | R2500 | R2666 | R9953 | R10312 | R10664 | R10709 | RANGE10G | CONTinuous}<NL>`

Example   `20 OUTPUT 707;":CRECOVERY1:RATE?"`

Table 31 Valid Data Rate Arguments Versus Modules

Rate Parameter	Rate (Mb/s)	Module Model Number									
		83491/2	83493	83494	83494 Option 103	83494 Option 106	83494 Option 107	83495 Option 100 & 200 Option 101 & 200	83496A/B	83496A/B Option 200	86108A/B
TOData [*]	–	●	●	●	●	●	●	●	●	●	●
R155	155.52	●	●	●	●	●	●		●	●	●
R622	622.08	●	●	●	●	●	●		●	●	●
R1062	1062.50	●							●	●	●
R1250	1250.00	●	●						●	●	●
R2125	2125.00	●							●	●	●
R2488	2488.32	●	●	●	●	●	●		●[†]	●[b]	●[b]
R2500	2500.00	●	●						●	●	●
R2666	2666.06					●	●		●	●	●
R9953	9953.28			●				●		●	●
R10312	10312.50				●			●		●	●
R10664	10664.23					●		●		●	●
R10709	10709.225						●	●		●	●
RANGE10G	9.953 Gb/s–11.32 Gb/s							●			
CONTinuous [‡]	–								●	●	

[*] Trigger on data.

[†] Default data rate.

[‡] The CONTinuous argument is returned in queries and can not be used to set the bandwidth.

RDIVider

Command :CRECovery{1 | 3}:RDIVider <divide_ratio>

This 83496A Option 300 and 86108A/B command sets or queries the data-rate divide ratio. This value is used to compute loop bandwidth when in the rate-dependent loop bandwidth mode. Refer to the RDIVider argument of the command "LBWMode" on page 157. The default value is 5000.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions 83496A Option 300 and 86108A/B modules. Software revision A.04.20 and above.

Example 10 OUTPUT 707; ":CRECOVERY1:RDIVIDER 4"

Query	`:CRECovery{1	3}:RDIVider?`
Returned Format	[:CRECovery{1	3}:RDIVIDER] <divide_ratio><NL>

RELock

Command   `:CRECovery{1 | 3}:RELock`

This command locks an 83496A or an 86108A/B module to the data rate. Issue this command to lock the module whenever changes occur in the data rate or input data source. Under two conditions, the module may lock on a data rate other than the specified rate. In the first condition, lock can occur if the entered data rate is an integer multiple of the actual data rate of the signal. The second condition occurs because the acquisition range is broad (greater than ±5000 PPM). This makes it possible for the module to lock on a signal that is higher or lower than the selected value. For example, if you select a 2.48832 Gb/s data rate but the signal is actually 2.5 Gb/s, the module may still lock on the signal. If an 83496A module is locked, sending the RELock command does not set the Clock Recovery Event Register's UNLK bit (bit 0) or LOCK bit (bit 1). Refer to "Clock Recovery Event Register (CRER)" on page 34. To determine if the RELock command has completed, use the CRECovery:LOCKed? query.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions   83496A and 86108A/B modules. Software revision A.04.20 and above.

Example   `10 OUTPUT 707; ":CRECOVERY1:RELock"`

SPResent?

Query   `:CRECovery{1 | 3}:SPResent? {RECeiver1 | RECeiver2}`

This 83492/3/4A query returns the status of whether the specified receiver detects an optical signal (Signal PResent). RECeiver2 is used for long wavelengths and RECeiver1 is used for short wavelengths. For electrical clock recovery modules (83491A), the signal present flags will always return false. This query does not apply to 83495A or 83496A modules. Refer to Table 32 on page 162. For related information on the CRER register, refer "Clock Recovery Event Register (CRER)" on page 34.

Returned Format   `[:CRECovery{1 | 3}:SPResent] {RECeiver1 | RECeiver2}, {1 | 0}<NL>`

Restrictions   83492/3/4A modules.

Example   `10 OUTPUT 707;":CRECOVERY3:SPRESENT? RECEIVER2"`

Table 32   Signal Present Return Status vs. Receiver Number

Module Model	Receiver 1 Short Wavelength	Receiver 2 Long Wavelength
83491	0	0
83492[*]	1/0	1/0
83493	0	1/0
83494	0	1/0
83494 Option 103	0	1/0
83494 Option 106	0	1/0
83494 Option 107	0	1/0

* Only one receiver at a time can have a signal present.

TDENsity?

Query   :CRECovery{1 | 3}:TDENsity?

Use this 83496A/B and 86108A/B query returns the calculated edge density of the data signal. The edge density value is the ratio of bit transistions to bits and is returned as a number between zero and one. Changes in edge density can cause the module to lose lock. If the edge density value is invalid, the string "9.99999E+37" is returned.

With 86108A/B modules, the :CRECovery3: command is not available. Use :CRECovery1: instead.

Restrictions   83496A/B and 86108A/B modules. Software revision A.04.20 and above.

Example   10 OUTPUT 707;":CRECOVERY1:TDENSITY?"

Returned Format   [:CRECovery{1 | 3}:TDEN] <edge_density><NL>

T2TFrequency?

Query   :CRECovery1:T2TFrequency?

Queries the Type-2 transition frequency in use for the current CDR loop bandwidth. **Refer to "LSELect" on page 158** for a description of this feature.

Restrictions   86108A/B modules. Software revision A.08.00 and above.

Example   10 OUTPUT 707; ":CRECOVERY1:T2TFREQUENCY?"

Returned Format   [:CRECovery1:T2TFrequency] <frequency><NL>

10 Disk Commands

The DISK subsystem commands allow storage and retrieval of waveforms and setups, remote screen captures, as well as formatting the disk. Some commands in this subsystem operate only on files and directories on "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive. These instances are noted in the command section. When specifying a file name, you must enclose it in quotation marks. For information on file naming, folder, and saving conventions, refer to "Files" on page 23.

BFILe?

Query   `:DISK:BFILe? <filename>`

Returns the requested file from the instrument using a binary block-transfer of data, with no restrictions on file contents or size. To return a text file, use the command "TFILe?" on page 174.

Returned Format   `[:DISK:BFILe]<filename><NL>`

Example   `10 OUTPUT 707;":DISK:BFIL?"`

CDIRectory

Command   `:DISK:CDIRectory ["<directory>" | {CGRade | LSUMmaries | ROOT | SETups | SIMages | SMASks | TDRCal | UMASks | WAVeforms}]`

Changes the present working directory (PWD) to the designated directory name. If an error occurs, the requested directory does not exist. You can view the error with the :SYSTem:ERRor? [{NUMBer | STRing}] query. The PWD is set to "D:\User Files" when the instrument is powered on. The PWD is combined with relative file specifications to produce absolute path specifications. For example, if the PWD is set to "D:\User Files\My Setup", the command :DISK:STORE SETUP, ".\setup1.set" will cause the current setup to be stored in the file "D:\User Files\My Setup\setup1.set". The argument <directory> is a character-quoted ASCII string, which can include the subdirectory designation. You must separate the directory name and any subdirectories with a backslash (\). The ROOT parameter changes the working directory to "D:\User Files".

NOTE   This command operates only on files and directories on "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

Example   `10 OUTPUT 707;":DISK:CDIRECTORY ""D:\USER FILES\DATA"""`

NOTE   You cannot execute the command CDIR "A:\" on 86100A/B instruments. Also, you cannot execute the command CDIR "C:\" or CDIR "D:\" (86100C/D). If you attempt to execute CDIR "C:\" or CDIR "D:\" (86100C/D), the present working directory (PWD) is not changed. The directory specified *must* be below "D:\User Files\".

DELete

Command   `:DISK:DELete "<file_name>"`

Deletes a file from the disk. If no path is specified, it searches for the file using the present working directory. <file_name> is a character-quoted ASCII string which can include subdirectories with the name of the file. The following error is displayed on the analyzer screen if the requested file does not exist: The file "`D:\User Files`" *cannot* be deleted.

NOTE   This command operates only on files and directories on "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

Example   `10 OUTPUT 707;":DISK:CDIRECTORY SETUPS"`
`20 OUTPUT 707;":DISK:DELETE ""FILE1.SET"""`

DIRectory?

Query   `:DISK:DIRectory? [ "<directory>" | {CGRade | ROOT | LSUMmaries | SETups | SIMages | SMASks | TDRCal | UMASks | WAVeforms}]`

Returns the requested directory listing. The directory may be specified as a string, such as "D:\User Files\waveforms", or as a parameter. (C drive on 86100A/B instruments.) If no parameter is used, a listing of the present working directory is returned. Each line in the returned list is terminated in a newline character only. A carriage return character is not included with the newline character.

<directory>   The list of file names and directories.

Returned Format   `[:DISK:DIRectory]<N><NL><directory><NL>`

<N>   The specifier that is returned before the directory listing, indicating the number of lines in the listing.

<directory>   The list of filenames and directories. Each line is separated by a <NL>.

Example   This example displays a number, then displays a list of files and directories in the current directory. The number indicates the number of lines in the listing.

```
10 DIM A$[80]
20 INTEGER Num_of_lines
30 OUTPUT 707;":DISK:DIR?"
40 ENTER 707;Num_of_lines
50 PRINT Num_of_lines
60 FOR I=1 TO Num_of_lines
70 ENTER 707;A$
80 PRINT A$
90 NEXT I
100 END
```

LOAD

Restrictions   Software revision A.04.00 and above (86100C instruments) or 86100D instruments for jitter data memory argument.

Command   `:DISK:LOAD "<file_name>"[,<destination>[,APPend]]`

Restores a setup, waveform, jitter data, or TDR/TDT calibration from the disk. The type of file is determined by the file name suffix if one is present, or by the destination field if one is not present. If a destination is specified, it takes precedence over the file name suffix. You can load .wfm, .txt, .cgs, .msk, .pcm, .set, .jd, and .tdr file types. The TDRTDT option is a file type choice used to load TDR/TDT calibration values into the instrument. For more information on loading files, see **"Files"** on page 23. Horizontal scale and delay information is not saved in jitter data or color grade-gray scale memory files. If you plan on loading these files back into the instrument, be sure to also store the instrument setup. You will need to load (restore) the instrument settings when you load the memory file.

<file_name>   The filename, with a extension: .wfm, .txt, .cgs, .msk, .pcm, .set, .jd, or .tdr as a suffix after the filename. If no file suffix is specified, the default is .wfm. The default directory for the file type is assumed, or you can specify the entire path. For example, you can load the standard setup file "setup0.set" using the command:

`:DISK:LOAD "D:\User Files\Setups\setup0.set",setup`

The default destination for .txt and .wfm files is WMEMory1.

<destination>   `{CGMemory | MASK | WMEMory<N> | SETup | JDMemory | TDRTDT}`

This command operates only on files and directories on "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

Do not use this command with a <destination> specified other than SETup and JDMemory in Jitter Mode. Using other <destination> arguments generate a "Settings conflict" error.

APPend   This optional parameter is used to turn on additional channels in Eye/Mask mode without turning off any channel(s) that are currently on. Without the APPend parameter, all other database signals would be turned off when loading .cgs file.

<N>   An integer from 1 to 4.

Example   `10 OUTPUT 707;":DISK:LOAD ""FILE1.WFM"",WMEM1"`

MDIRectory

Command   `:DISK:MDIRectory "<folder>"`

Creates a new folder (directory). The <folder> argument must be an ASCII string of the entire path. An error occurs if the requested path to the new folder does not exist. This command operates only on files and folders on "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

Example   `10 OUTPUT 707;":DISK:MDIRECTORY ""d:\User Files\cprograms"""`

PWAVeform:LOAD

Command	`:DISK:PWAVeform:LOAD <file_name> [,{CHANnel<N>	FUNCtion<N> }]`

Loads a pattern waveform file into color gray-scale memory. If the pattern waveform file contains data from several sources, only the data from one of the sources can be loaded from the file. Use the CHANnel or FUNCtion arguments to select the source data to load into memory. Source data from CHANnel1 is selected by default.

If you plan on loading a saved pattern waveform back into the instrument, be sure to also save the instrument setup. You will need to load (restore) the instrument settings at the same time that you load the associated pattern waveform.

Restrictions	Software revision 4.10 and above on an 86100C/D. Option 201, Advanced Waveform Analysis Software installed. Eye/Mask or Oscilloscope instrument mode with pattern lock triggering. One or more channels or functions (invert, subtract, or magnify) turned on. Optional MATLAB Filter and Linear Feedforward Equalizer applications closed (not running).
Example	`10 OUTPUT 707;":DISK:PWAVEFORM:LOAD "FILE1""`

PWAVeform:PPBit

Command	`:DISK:PWAVeform:PPBit <number_points>`

Sets or queries the number of samples per bit in a pattern waveform. <number_points> can be an integer from 16 to through 4095. Turn the instrument's pattern lock on before sending this command.

Restrictions	Software revision 4.10 and above on an 86100C/D. Option 201, Advanced Waveform Analysis Software installed.
Query	`:DISK:PWAVeform:PPBit?`
Returned Format	`[:DISK:PWAVeform:PPBit] <number_points><NL>`
Example	`10 OUTPUT 707;":DISK:PWAVEFORM:PPBIT 4095"`

PWAVeform:RANGe

Command	`:DISK:PWAVeform:RANGe {EPATtern	SRANge}`

Sets or queries the range setting for saving pattern waveforms when the DISK:PWAVeform:SAVE command. EPATtern saves the entire pattern waveform. SRANge specifies that a range of bits to save. Set the start and stop bits of the range using the DISK:PWAVeform:RANGe:STARt and DISK:PWAVeform:RANGe:STOP commands. Turn the instrument's pattern lock on before sending this command.

Restrictions	Software revision 4.10 and above on an 86100C/D. Option 201, Advanced Waveform Analysis Software installed.
Query	`:DISK:PWAVeform:RANGe?`
Returned Format	`[:DISK:PWAVeform:RANGe] {EPATtern	SRANge}<NL>`
Example	`10 OUTPUT 707;":DISK:PWAVeform:RANGe EPATtern"`

PWAVeform:RANGe:STARt

Restrictions	Software revision 4.10 and above on an 86100C/D. Option 201, Advanced Waveform Analysis Software installed.

Command	`:DISK:PWAVeform:RANGe:STARt <bit_number>`

Sets or queries the start bit setting for saving a range of pattern waveform bits using the DISK:PWAVeform:SAVE command. <bit_number> is an integer. You must first specify that a range of the pattern will be saved by using the DISK:PWAVeform:RANGe command.

Query	`:DISK:PWAVeform:RANGe:STARt?`

Returned Format	`[:DISK:PWAVeform:RANGe:STARt] <bit_number><NL>`

Example	`10 OUTPUT 707;":DISK:PWAVEFORM:RANGE:START 10"`

PWAVeform:RANGe:STOP

Command	`:DISK:PWAVeform:RANGe:STOP <bit_number>`

Sets or queries the stop bit setting for saving a range of pattern waveform bits using the DISK:PWAVeform:SAVE command. <bit_number> is an integer. You must first specify that a range of the pattern will be saved by using the DISK:PWAVeform:RANGe command.

Restrictions	Software revision 4.10 and above on an 86100C/D. Option 201, Advanced Waveform Analysis Software installed.

Query	`:DISK:PWAVeform:RANGe:STOP?`

Returned Format	`[:DISK:PWAVeform:RANGe:STOP] <bit_number><NL>`

Example	`10 OUTPUT 707;":DISK:PWAVEFORM:RANGE:STOP 20"`

PWAVeform:SAVE

Command	`:DISK:PWAVeform:SAVE <file_name>`

Saves a pattern waveform to a file with the file extension .csv. <file_name> is the name of the file, with a maximum of 254 characters (including the path name, if used). The file name assumes the present working directory if a path does not precede the file name. The data is saved in an ASCII comma separated file (csv), with the amplitude data for each source (channel or function) placed in a separate column. In addition to amplitude values, saved pattern waveform files include a header of setup information. Patterns that include a large number of bits and high resolution involve large amounts of data. Saving these files may require several hours and one or two gigabytes (GB) of memory. Use *OPC or *OPC? with this command in order to synchronize data acquisition with remote control. If you plan on loading a saved pattern waveform back into the instrument, be sure to also save the instrument setup. You will need to load (restore) the instrument settings at the same time that you load the associated pattern waveform.

Restrictions	Software revision 4.10 and above on an 86100C/D. Option 201, Advanced Waveform Analysis Software installed. Eye/Mask or Oscilloscope instrument mode with pattern lock triggering. One or more channels or functions (invert, subtract, or magnify) turned on. Optional MATLAB Filter and Linear Feedforward Equalizer applications closed (not running).

Example	`10 OUTPUT 707;":DISK:PWAVEFORM:SAVE "FILE1";*OPC?"`

PWD?

Query	`:DISK:PWD?`

Returns the name of the present working directory (including the full path).

Returned Format	`[:DISK:PWD] <present_working_directory><NL>`

Example   `20 OUTPUT 707;":DISK:PWD?"`

SIMage

Command   `:DISK:SIMage "<filename>"[,{SCReen | GRATicule} [,{NORMal | INVert | MONochrome}]] [, {CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N> | CGMemory}]`

Captures an image of the display's active window and saves it into a graphics file. To capture a screen image when a limit test fails, use the command **"SSCReen"** on page 206. To capture a screen image when a mask test fails, use the command **"SSCReen"** on page 228. To capture a screen image upon completion of a specified waveform acquisition (number of averages and the number of data points), use the command **"SSCReen"** on page 127.

When using the SIMage command to capture screen images:

· Files can only be created within the folder "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

· Files can not be saved on the root folder of the D: drive (C: on 86100A/B).

· Files can not be saved on USB removable drives. To save files on a USB drive, use front-panel controls. *(Applies only to firmware version A.09.00 and below)*

· Using the command **"CDIRectory"** on page 165 to change the present working directory has no effect on the location of saved files.

The <filename> argument includes the folder (and path) in which to save the file, as well as the file name. The following table shows examples of valid filenames including one invalid filename. The following graphics formats are available by specifying a file extension: PCX files (.pcx), EPS files (.eps), Postscript files (.ps), JPEG files (.jpg), TIFF files (.tif), and GIF files (.gif). The default file type is a bitmap (.bmp). On 86100C/D instruments, if the 86100C/D application has been minimized, an image of the desktop or another application will be captured. When capturing 86100C/D images, first deactivate the Windows XP screen saver. Otherwise, if the screen saver is active, the captured image may be solid black.

Table 33   Example Filenames

File Name	File Saved in Directory...
"test.jpg"	D:\User Files\screen images\ This is the default folder. Filenames without a path are saved to this folder.
"subfolder\test.jpg"	D:\User Files\screen images\subfolder The subfolder must already exist before saving the file.
"D:\User Files\subfolder\test.jpg"	D:\User Files\subfolder The subfolder must already exist before saving the file.
"D:\User Files\test.jpg"	D:\User Files
"D:\test.jpg"	This is not a valid file location. The file is not saved.
"E:test4.pcx"	File saved in the instrument's drive E:, that could be mapped to any disk in the network.
"\\computer-ID\d$\test3.bmp"	File saved in drive D: of computer "computer-ID", provided all permissions are set properly.

NOTE   For .gif and .tif file formats, this instrument uses LZW compression/decompression licensed under U.S. patent No 4,558,302 and foreign counterparts. End user should not modify, copy, or distribute LZW compression/decompression capability. For .jpg file format, this instrument uses the .jpg software written by the Independent JPEG Group.

Selecting GRATicule saves only the display's graticule area. Selecting SCReen saves the entire display. Use the {NORMal | INVert | MONochrome} arguments to specify the color scheme used during the save operation. The default value is INVert which saves the waveforms over a white background.

When multiple waveforms are displayed, you can elect to have only one of the waveforms included on the captured image. None of the other waveforms will be included in the capture. Select the waveform to capture using one of the optional {CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N> | CGMemory} arguments. If this argument is not included, all displayed waveforms will be included in the capture.

Restrictions	Software revision A.10.60 and above for selecting a single waveform to include in the capture.

Example	`10 OUTPUT 707;":DISK:SIM "test.jpg", SCReen, INVert"`

SPARameter:SAVE

Command	`:DISK:SPARameter:SAVE <source>,"<file_name>"[,<format>[,<field>]]`

Saves an S-parameter waveform to ASCII Touchstone files and text files. Before you can save S-parameter data to a file, you must first display the S-parameter graph using the command "TDRSparam" on page 302.

For one-port single-ended devices, save your data (S11 or S22) to Touchstone (.s1p) files. For two-port single-ended devices, save your data (S11, S21, S22, S12) to Touchstone (.s2p) files. For four-port single-ended devices, save your data (S11, S21, S31, ... S44) to touchstone (.s4p) files.

When saving multiple S-parameters to an s2p or s4p file, you must save each S-parameter as a separate save, appending each S-parameter data to the original file. The <field> argument selects the S-parameter for each appended save. Differential and common mode S-parameter measurements can not be saved to Touchstone files. Any single S-parameter (single-ended, differential mode, or common mode) can be saved to a text file that uses the identical format as the Touchstone s1p file. While Touchstone files can not be imported back into the 86100C/D, you can import them into circuit simulators for further analysis.

The <source> argument can be CHANnel<n>, FUNCtion<n>, RESPonse<n>, or WMEMory<n>. The <file_name> argument is the name of the file, with a maximum of 254 characters (including the path name, if used). The file name assumes the present working directory if a path does not precede the file name. The <format> argument can be TEXT (.txt), S1P (Touchstone .s1p), or S2P (Touchstone .s2p), or S4P (Touchstone .s4p). The default file format is TEXT. Use the optional <field> argument when saving Touchstone S2P or S4P files to indicate the S-parameter (S11, S21, ...) being saved. Each of these S-parameters is assigned a fixed field in the Touchstone file as listed in Table 34 and Table 35.

Table 34   S-Parameters and <field> Arguments for S2P Files

S-Parameter	<field> Argument
S11	1
S21	3
S21	2
S22	4

Table 35 S-Parameters and ‹field› Arguments for S4P Files

S-Parameter	‹field› Argument	S-Parameter	‹field› Argument
S11	1	S31	9
S21	2	S31	10
S13	3	S33	11
S14	4	S34	12
S21	5	S41	13
S22	6	S42	14
S23	7	S43	15
S24	8	S44	16

The Touchstone file consists of lines of comma separated ASCII strings. Lines 1 and 2 are commented description lines that begin with the comment delimiter character (!). Line 3 is the option line that specifies measurement parameters for the data content (frequency, magnitude, phase) using the following format:

‹frequency unit› ‹parameter› ‹format› ‹R n›

Line 3 begins with the # character. The ‹frequency units› specifies Hz, KHz, MHz, or GHz. The ‹parameter› field specifies S. The ‹format› field specifies DB for magnitude (logarithmic) –angle. The ‹R n› field specifies the reference resistance in ohms, where n is the positive number of ohms of the real impedance to which the parameters are calibrated.

Line 4 immediately precedes the data and labels the fields contained in the data lines.

The following lines are an example of the first few lines of a TEXT or S1P file:

```
!Agilent Infiniium DCA-J 86100
!1-port S-Parameter file, single frequency point
# Hz S DB R 50
!freq      dbS11 angS11
0.000e+000 0.01    0.0
1.000e+008 0.15    0.1
2.000e+008 0.18   -0.6
3.000e+008 0.15   -1.3
```

The same file saved in the S2P format would have the following entries. Notice that fields that have not been appended to the file yet have all data values entered as 0.0.

```
!Agilent Infiniium DCA-J 86100
!2-port S-Parameter file
!Instrument Configuration - Time/Div: 1.000 nS, Points/Waveform: 4096 points
# Hz S DB R 50
!freq      dbS11 angS11  dbS21 angS21  dbS12 angS12 dbS22 angS22
!
0.000e+000  0.03   0.0   0.00   0.0   0.00   0.0  0.00    0.0
1.000e+008  0.16   0.1   0.00   0.0   0.00   0.0  0.00    0.0
2.000e+008 0.19   -0.1   0.00   0.0   0.00   0.0  0.00    0.0
3.000e+008 0.16   -1.2   0.00   0.0  0.00    0.0   0.00   0.0
```

The following lines are an example of the first few lines of a TEXT or S4P file:

```
!Agilent Infiniium DCA-J 86100
!4-port S-Parameter file
!Instrument Configuration - Time/Div: 500.0 pS, Points/Waveform: 1024 points
# Hz S DB R 50
!freq      dbS11 angS11  dbS12 angS12 dbS13 angS13  dbS14 angS14
!          dbS21 angS21  dbS22  angS2 dbS23 angS23 dbS24 angS24
!          dbS31  angS3  dbS32 angS32  dbS33 angS33 dbS34 angS34
!          dbS41 angS41  dbS42 angS42  dbS43 angS43  dbS44 angS44
!
0.0000000e+000 -63.90 0.0 0.00   0.0   0.00   0.0   0.00   0.0
               0.00      0.0  0.00    0.0  0.00   0.0   0.00   0.0
```

```
                           0.00    0.0    0.00     0.0   0.00    0.0   0.00    0.0
                           0.00    0.0    0.00     0.0   0.00    0.0   0.00    0.0

       2.0000000e+008  -62.65  -172.0 0.00 0.0   0.00    0.0   0.00    0.0
                           0.00    0.0    0.00     0.0   0.00    0.0   0.00    0.0
                           0.00    0.0    0.00     0.0   0.00    0.0   0.00    0.0
                           0.00    0.0    0.00     0.0   0.00    0.0   0.00    0.0
```

Restrictions	Software revision 6.00 and above on an 86100C or an 86100D instrument. Option 202, Enhanced Impedance and S-Parameter Software installed. TDR/TDT mode. Software revision 10.0 and above for saving S4P files.
Examples	`10 OUTPUT 707;":DISK:SPARAMETER:SAVE RESP1, "FILE1", TEXT"` `10 OUTPUT 707;":DISK:SPARAMETER:SAVE RESP3, "FILE1", S2P, 3`

STORe

Command	`:DISK:STORe <source>,"<file_name>"[,<format>]`
	Stores a setup, waveform, jitter data, or TDR response to the disk. The file name does not include a suffix. The suffix is supplied by the instrument depending on the source and file format specified. The TDRTDT option is a file type choice used to store the instrument's TDR/TDT calibration values. For more information on storing files, see "Files" on page 23. Because horizontal scale and delay information is not saved in jitter data or color grade-gray scale memory files, if you plan on loading these files back into the instrument, be sure to also store the instrument setup. You will need to load (restore) the instrument settings when you load the memory file.
Restrictions	Software revision A.04.00 and above (86100C instruments) for jitter data memory argument. Software revision A.05.00 and above (86100C instruments) for XYVerbose <format> argument. Or, an 86100D instrument.
<source>	`{CHANnel<N>	FUNCtion<N>	WMEMory<N>	SETup	RESPonse<N>	CGRade	JDSource	TDRTDT}`
	If a CGRade source has not been selected, CGRade defaults to the lowest valid database available. To set the CGRade source, use the :WAVeform:SOURce:CGRade command.

NOTE	In Jitter Mode, this command generates a "Settings conflict" error if sources other than SETup and JDSource are specified.

	With the <source> argument, <N> represents an integer from 1 to 4, which identifies the channel, function, TDR response or waveform memory number. Name of the file, with a maximum of 254 characters (including the path name, if used). The file name assumes the present working directory if a path does not precede the file name.
<format> for Waveforms	`{INTernal	TEXT {,<YVALues>	<VERBose>	<XYVerbose>}}`
	Include <format> when the <source> argument is WMEMory. The default is INTernal. In TEXT mode, y values may be specified so that only the y values are stored. VERBose is the default in which y values and the waveform preamble are stored. XYVerbose files contain both x and y values. Only waveforms of 128K or less may be written to disk in the TEXT formats. See Chapter 26, "Use the WAVeform subsystem to transfer waveform data between a computer and the instrument." for information on converting data to values.
<format> for Jitter Data	`{INTernal	CSV}`

Include <format> when the <source> argument is JDSource. The CSV argument selects data to be saved as comma separated values in a text file. This text file can be opened in text editors, spreadsheet applications, and word processors. The default argument is INTernal. See Chapter 26, "Use the WAVeform subsystem to transfer waveform data between a computer and the instrument." for information on converting data to values.

NOTE This command operates only on files and directories on "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

Example   10 OUTPUT 707;":DISK:STORE SET,""FILE1"""

TFILe?

Query   :DISK:TFILe? <filename>

Returns the requested text file from the instrument. The file must be smaller than 256,000 characters. If the file does not contain text, the return string will be terminated at the first zero (0) values byte in the file. If Option 202 Enhanced Impedance and S-Parameter Software is installed, you can use this command to return touchstone files. However, group delay information is not included in the file. To return a binary file, use the command "BFILe?" on page 165.

Returned Format   [:DISK:TFILe]<filename><NL>

Example   10 OUTPUT 707;":DISK:TFIL?"

11 Display Commands

The DISPlay subsystem controls the display of data, markers, text, graticules, and the use of color. You select the display mode using the ACQuire:TYPE command. Select the number of averages using ACQuire:COUNt.

CGRade:LEVels?

Query `:DISPlay:CGRade:LEVels? [CHANnel<N> | FUNCtion<N> | CGMemory]`

Returns the range of hits represented by each color for the specified source. If no source is specified, the values for the first database signals turned on is returned. Fourteen values are returned, representing the minimum and maximum count for each of seven colors. The values are returned in the following order:

- Greatest intensity color minimum

- Greatest intensity color maximum

- Next greatest intensity color minimum

- Next greatest intensity color maximum

- . . . .

- Least intensity color minimum

- Least intensity color maximum

Returned Format   `[:DISPlay:CGRade:LEVels] <color format><NL>`

The <color format> argument is integer values from 0 to 63,488.

Example   The following example gets the range of hits represented by each color.
```
10 DIM Setting$[50]!Dimension variable
20 OUTPUT 707;":DISPLAY:CGRADE:LEVELS?"
30 ENTER 707;Cgrade$
```

CONNect

Command   `:DISPlay:CONNect {{ON | 1}|{OFF | 0}}`

When enabled, :DISPlay:CONNect draws a line between consecutive waveform data points. This is also known as linear interpolation. This command has no effect on color grade or gray scale displays.

Example   This example turns on the connect-the-dots feature.
```
10 OUTPUT 707;":DISPLAY:CONNECT ON"
```

Query   `:DISPlay:CONNect?`

The query returns the status of the connect-the-dots feature.

Returned Format   `[:DISPlay:CONNect] {ON | OFF}<NL>`

DATA?

Query   `:DISPlay:DATA? [<format>[,{SCReen | GRATicule} [,<image>]]]`

Returns an image of the current display in the specified file format. If no arguments are specified, the default selections are PCX file type, SCReen mode, and inversion set to INVert. The BMP and JPG file formats are the only formats that are saved with 24 bit color and provide the highest quality image.

The <format> argument is the file format: BMP, PCX, EPS, EPS, GIF, TIF, or JPG. GRATicule selects only the graticule area of the display screen to save; the entire screen is saved if you select the default setting SCReen. The <image> argument specifies the color scheme used during the screen save operation: {NORMal | INVert | MONochrome}. The default value is INVert which saves the waveforms over a white background.

Returned Format   `[:DISPlay:DATA] <binary_block_data><NL>`

Data is returned in the IEEE 488.2 definite block format.

DCOLor

Command	`:DISPlay:DCOLor`
	Resets the screen colors to the predefined factory default colors and resets the grid intensity.
Example	`10 OUTPUT 707;":DISPLAY:DCOLOR"`

ETUNing

Command	`:DISPlay:ETUNing{ON	0 }	{OFF	1}`
	Turns eye tuning on or off.

Use eye tuning to obtain a variable persistence display in Eye/Mask mode. It is especially valuable when you are tuning a device while simultaneously watching changes to the eye diagram and eye diagram measurements. For example, you may tune a laser bias point while monitoring the Extinction Ratio measurement.

It is important to note that eye tuning may decrease the accuracy of the measurement results. When your adjustments are completed, clear the eye tuning selection to ensure the measurement results return to their highest accuracy. Eye tuning is not selectable until Color Grade or Grey Scale persistence is selected.

Eye tuning works by weighting recent waveforms more heavily than older waveforms in the eye diagram database. You can adjust the effective amount of time each waveform remains in the eye diagram database by adjusting the record length. With shorter record lengths, each waveform will decay more quickly, while with longer record length they will persist longer. To change the record length, refer to "POINts" on page 125.

Restrictions	Software revision A.09.00 and above.
Example	`10 OUTPUT 707;":DISPLAY:ETUNING ON"`
Query	`:DISPlay:ETUNing?`
Returned Format	`[:DISPlay:ETUNing] {ON	0 }	{OFF	1}<NL>`

GRATicule

Commands	`:DISPlay:GRATicule {GRID|FRAMe}` `:DISPlay:GRATicule:INTensity <intensity_value>`

Select the type of graticule that is displayed. 86100C analyzers have a 10-by-8 (unit) display graticule grid that you can turn on or off. When the grid is on, a grid line is place on each vertical and horizontal division. When it is off, a frame with tic marks surrounds the graticule edges.

<intensity_value> is a number from 0 to 100, indicating the percentage of display intensity. You can dim the grid's intensity or turn the grid off to better view waveforms that might be obscured by the graticule lines. Otherwise, you can use the grid to estimate waveform measurements such as amplitude and period. When printing, the grid intensity control doesn't affect the hardcopy. To remove the grid from a printed hardcopy, you must turn off the grid before printing.

Example	This example sets up the analyzer's display background with a frame that is separated into major and minor divisions. `10 OUTPUT 707;":DISPLAY:GRATICULE FRAME"`
Queries	`:DISPlay:GRATicule?` `:DISPlay:GRATicule:INTensity?`
	The queries return the type of graticule currently displayed, or the intensity, depending on the query you request.

Returned Format	[:DISPlay:GRATicule] {GRID|FRAMe}<NL> [:DISPlay:GRATicule:INTensity] <value><NL>

Example   This example places the current display graticule setting in the string variable, Setting$.

```
20 OUTPUT 707;":DISPLAY:GRATICULE?"
30 ENTER 707;Setting$
```

JITTer:BATHtub:YSCale

Command   :DISPlay:JITTer:BATHtub:YSCale {BER | Q}

Sets the vertical scale of the bathtub display to either BER or Q.

Restrictions   86100C/D with Jitter Mode. 86100C software revision A.04.10 and above including revision A.07.00. When writing new code for software revision A.07.00 and above, use the recommended command **"SINTegrity:BATHtub:YSCale"** on page 182.

Example   10 OUTPUT 707; ":DISPlay:JITTer:BATHtub:YSCale BER"

Query   :DISPlay:JITTer:BATHtub:YSCale?

Returned Format   [:DISPlay:JITTer:BATHtub:YSCale] {BER | Q}<NL>

JITTer:GRAPh

Command   :DISPlay:JITTer:GRAPh {<graph>}[,{<graph>}[,{<graph>}[,{<graph>}]]]

Turns on the specified graphs. From one to four graphs may be specified, regardless of the current graph layout. The graphs will be selected in order from last to first. The graph specified by the first parameter will be the one displayed on single-graph layout, on top for split layout, and in the upper left corner for quad layout.

The <graph> argument represents {BATHtub | CDDJhist | CTJHist | DDJHist | DDJVsbit | PJWaveform | RJPJhist | SRJSpectrum | TJHist | JSPectrum}.

The BATHtub, PJWaveform, and SRJSpectrum arguments are not available to Option 100 installations of Jitter Mode.

Restrictions   86100C/D with Jitter Mode. 86100C software revision A.04.10 and above including revision A.07.00. When writing new code for software revision A.07.00 and above, use the recommended command **"SINTegrity:GRAPh"** on page 183.

Example   10 OUTPUT 707; ":DISPlay:JITTer:GRAPh TJHist"

Query   Returns a list of the currently displayed graphs. The returned values are comma-separated and listed in the order that they were turned on. The first value is the most recently selected graph.
:DISPlay:JITTer:GRAPh?

Returned Format   [:DISPlay:JITTer:GRAPh] <list of graphs><NL>

JITTer:HISTogram:YSCale

Command   :DISPlay:JITTer:HISTogram:YSCale {LINear | LOGarithmic}

Specifies a linear or lagarithmic vertical scale for the jitter histogram.

Restrictions   86100C/D with Jitter Mode. 86100C software revision A.04.10 and above including revision A.07.00. When writing new code for software revision A.07.00 and above, use the recommended command **"SINTegrity:HISTogram:YSCale"** on page 183.

Example	`10 OUTPUT 707; ":DISPlay:JITTer:HISTogram:YSCale LINear"`
Query	`:DISPlay:JITTer:HISTogram:YSCale?`
Returned Format	`[:DISPlay:JITTer:HISTogram:YSCale] {LINear	LOG}<NL>`

JITTer:LAYout

Command	`:DISPlay:JITTer:LAYout {SINGle	SPLit	QUAD}`
	Sets the number of graphs displayed when in jitter mode. SINGle specified one graph, SPLit specifies two graphs and QUAD specifies four graphs.
Restrictions	86100C/D with Jitter Mode. 86100C software revision A.04.10 and above including revision A.07.00. When writing new code for software revision A.07.00 and above, use the recommended command **"SINTegrity:LAYout"** on page 183.
Example	`10 OUTPUT 707; ":DISPlay:JITTer:LAYout SPLit"`
Query	`:DISPlay:JITTer: LAYout?`
Returned Format	`[:DISPlay:JITTer:LAYout] {SINGle	SPLit	QUAD}<NL>`

JITTer:PJWFrequency

Command	`:DISPlay:JITTer:PJWFrequency <frequency>`
	For the PJ Waveform graph, sets or queries the frequency plotted on the graph. The command, :DISPlay:JITTer:PJWTracking, must be set to "off" before issuing the PJWFrequency command or query.
Restrictions	Jitter mode. Software revision A.04.10 and above (86100C instruments) or an 86100D instrument. Option 200, Enhanced Jitter Analysis Software.
Example	`10 OUTPUT 707;":DISPlay:JITTer:PJWFrequency 10E+6"`
Query	`:DISPlay:JITTer:PJWFrequency?`
Returned Format	`[:DISPlay:JITTer:PJWFrequency] <frequency><NL>`

JITTer:PJWTracking

Command	`:DISPlay:JITTer:PJWTracking {{ON	1}	{OFF	0}}`
	For the PJ Waveform graph, sets or queries the option for automatically tracking the frequency component with the greatest magnitude.
Restrictions	Jitter mode. Software revision A.04.10 and above (86100C instruments) or an 86100D instrument. Option 200, Enhanced Jitter Analysis Software.
Example	`10 OUTPUT 707;":DISPlay:JITTer:PJWTracking ON"`
Query	`:DISPlay:JITTer:PJWTracking?`
Returned Format	`[:DISPlay:JITTer:PJWTracking] {{ON	1}	{OFF	0}}<NL>`

JITTer:SHADe

| Command | `:DISPlay:JITTer:SHADe {{ON | 1}|{OFF | 0}}` |

Shows or removes the display of the jitter shade. The shade is the drop-down screen that is used to display the jitter graphs. Because showing the shade takes some time, use this command to reduce measurement times in situations where testing would continually open and hide the jitter shade.

Restrictions   86100C/D with Jitter Mode. 86100C software revision A.04.10 and above including revision A.07.00. When writing new code for software revision A.07.00 and above, use the recommended command "SINTegrity:SHADe" on page 184.

Example   `10 OUTPUT 707;":DISPlay:JITTer:SHADe ON"`

Query   `:DISPlay:JITTer:SHADe?`

Returned Format   `[:DISPlay:JITTer:SHADe] {{ON | 1}|{OFF | 0}}<NL>`

LABel

Command   `:DISPlay:LABel '<text>' [,<row>[,<column>[,<text_color>[,<background>]]]]`

Places a label on the graticule area of the display. You should periodically clear the labels using the LABel:DALL command.

Arguments   The string argument <text> is any series of ASCII characters enclosed in quotation marks. <row> is 0 to 12, where 0 is the top row and the default. <column> is 0 to 61, where 0 is the left column and the default. <text_color> is {CHANnel<N> | WHITe}. Default is WHITe. The <background> can be {OPAQue | TRANsparent}. Default is TRANsparent.

Example   `10 OUTPUT 707;":DISPLAY:LABEL 'This is a label'"`

LABel:DALL

Command   `:DISPlay:LABel:DALL`

Deletes all displayed labels.

Example   `10 OUTPUT 707;":DISPLAY:LABEL:DALL"`

PERSistence

Command   `:DISPlay:PERSistence {MINimum | INFinite | <persistence_value> | CGRade | GSCale}`

Sets the display persistence. The parameter for this command can be either MINimum (zero persistence), INFinite, or a real number from 0.1 to 40, representing the persistence in seconds, with one digit resolution, color grade, or gray scale. <persistence_value> is a real number, 0.1 to 40, representing the persistence in seconds.

Table 36   Persistence Values and Resolution

Persistence Value in Seconds	Resolution (Step Size)
0.1 - 0.9	0.1s steps
1 - 10	1s steps
10 - 40	10s steps

Mode   Refer to Table 37 on page 181 for CGRade and GSCale arguments.

Example   `10 OUTPUT 707;":DISPLAY:PERSISTENCE INFINITE"`

Table 37   CGRade and GSCale Arguments

Mode	Persistence				
	Minimum	Infinite	Variable	Color Grade	Gray Scale
Eye/Mask				•	•
TDR/TDT	•	•	•		
Oscilloscope	•	•	•	•	•

Query   :DISPlay:PERSistence?

Returned Format [:DISPlay:PERSistence] {MINimum | INFinite | <value> | CGRade | GSCale}<NL>

Example   10 OUTPUT 707;":DISPLAY:PERSISTENCE?"

RRATe

Command   :DISPlay:RRATe <refresh_rate>

Sets the display refresh rate. <refresh_rate> sets the refresh time in seconds. The minimum value is .01seconds, and the maximum value is 3600 seconds. The query returns the display refresh rate.

Example   This example sets the display refresh rate to 3 seconds.
10 OUTPUT 707;":DISPlay:RRATe 3"

Query   :DISPlay:RRATe?

Returned Format [:DISPlay:RRATe] <refresh_rate> <NL>

Example   20 OUTPUT 707;":DISPLAY:RRATE? "

SCOLor

Command   :DISPlay:SCOLor <color_name>, <hue>, <saturation>, <luminosity>

Sets the color of the specified display element and restores the colors to their factory settings. The display elements are described in Table 38 on page 181.

<color_name>   {CGRade1 | CGRADE2 | CGRADE3 | CGRADE4 | CGRADE5 | CGRADE6 | CGRade7 | CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | GRID | IMEasurement | MARGin | MARKers | MASK | MEASurements | WBACkgrnd | WOVerlap | WMEMories | WINText}

Table 38   Color Names

Color Name	Definition
CGRADE1	First range of pixel counts for the color grade persistence display
CGRADE2	Second range of pixel counts for the color grade persistence display
CGRADE3	Third range of pixel counts for the color grade persistence display
CGRADE4	Fourth range of pixel counts for the color grade persistence display
CGRADE5	Fifth range of pixel counts for the color grade persistence display
CGRADE6	Sixth range of pixel counts for the color grade persistence display

Table 38   Color Names (continued)

Color Name	Definition
CGRADE7	Seventh range of pixel counts for the color grade persistence display
CHANnel1	Channel 1 waveform display element.
CHANnel2	Channel 2 waveform display element.
CHANnel3	Channel 3 waveform display element.
CHANnel4	Channel 4 waveform display element.
GRID	Display element for the grid inside the waveform viewing area.
IMEasurement	Display element for the questionable or invalid measurement text.
MARGin	Display element for the margins.
MARKers	Display element for the markers.
MASK	Display element for the masks.
MEASurements	Display element for the measurements text.
WBACkgrnd	Display element for the waveform viewing area's background.
WOVerlap	Display element for waveforms when they overlap each other.
WMEMories	Display element for waveform memories.
WINText	Display element used in dialog box controls and pull-down menus.

<hue>   Sets the color of the chosen display element. As hue is increased from 0%, the color changes from red, to yellow, to green, to blue, to purple, then back to red again at 100% hue. For color examples, see the sample color settings table in the 86100C on-line help file. Pure red is 100%, pure blue is 67%, and pure green is 33%.

<saturation>   Sets the color purity of the chosen display element. The saturation of a color is the purity of a color or the absence of white. A 100% saturated color has no white component. A 0% saturated color is pure white.

<luminosity>   Sets the color brightness of the chosen display element. A 100% luminosity is the maximum color brightness. A 0% luminosity is pure black.

Example   This example sets the hue to 50, the saturation to 70, and the luminosity to 90 for the markers.
```
10 OUTPUT 707;":DISPLAY:SCOLOR MARKERS,50,70,90"
```

Query   `:DISPlay:SCOLor? <color_name>`

Returned Format   `[:DISPlay:SCOLor] <color_name>, <hue>, <saturation>, <luminosity><NL>`

SINTegrity:BATHtub:YSCale

Command   `DISPlay:SINTegrity:BATHtub:YSCale {BER | Q}`

Sets the vertical scale of the jitter bathtub graph and the amplitude bathtub graph to BER or Q. When writing new code, this is the recommended replacement for the command "JITTer:BATHtub:YSCale" on page 178.

Restrictions	86100C/D (software revision A.07.00 and above) with Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":DISPLAY:SINTEGRITY:YSCALE BER"`
Query	`:DISPlay:SINTegrity:BATHtub:YSCale?`
Returned Format	`[:DISPlay:SINTegrity] {BER	Q}<NL>`

SINTegrity:GRAPh

Command	`:DISPlay:SINTegrity:GRAPh {<graph>} [,{<graph>} [,{<graph>} [,{<graph>}] ] ]` Turns on the specified graphs. From one to four graphs may be specified, regardless of the current graph layout. The graphs will be selected in order from last to first. The graph specified by the first parameter will be the one displayed on single-graph layout, on top for split layout, and in the upper left corner for quad layout. When writing new code, this is the recommended replacement for the command **"JITTer:GRAPh"** on page 178. Valid graph parameters: `{JBAThtub	CDDJhist	CTJHist	DDJHist	DDJVsbit	PJWaveform	RJPJhist	SRJSpectrum	TJHist	JSPectrum	ABAThtub	CTIHist	ISIHist	ISIVsbit	RNPIhist	TIHist	NSPectrum }`
Restrictions	86100C/D (software revision A.07.00 and above) with Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":DISPLAY:SINTEGRITY:GRAPh ABAThtub, RNPlhist"`
Query	Returns a list of the currently displayed graphs. The returned values are comma-separated and listed in the order that they were turned on. The first value is the most recently selected graph. The possible return values are the short form of the graph parameters listed above. `:DISPlay:SINTegrity:GRAPh?`
Returned Format	`[:DISPlay:SINTegrity:GRAPh] {graph} [,{graph} [,{graph} [,{graph}] ] ]<NL>`

SINTegrity:HISTogram:YSCale

Command	`:DISPlay:SINTegrity:HISTogram:YSCale {LINear	LOGarithmic}` Specifies a linear or logarithmic vertical scale for the jitter, noise, and interference histograms. When writing new code, this is the recommended replacement for the command **"JITTer:HISTogram:YSCale"** on page 178.
Restrictions	86100C/D (software revision A.07.00 and above) with Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":DISPLAY:SINTEGRITY:YSCALE LINEAR"`
Query	`:DISPlay:SINTegrity:HISTogram:YSCale?`
Returned Format	`[:DISPlay:SINTegrity:YSCale] {LINear	LOG}<NL>`

SINTegrity:LAYout

Command	`:DISPlay:SINTegrity:LAYout {SINGle	SPLit	QUAD}` Specifies the number of plots displayed in Noise Mode and Jitter/Noise mode. SINGle specifies one graphs, SPLit specifies two graphs, and QUAD specifies four graphs. When writing new code, this is the recommended replacement for the command **"JITTer:LAYout"** on page 179.

Restrictions	86100C/D (software revision A.07.00 and above) with Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":DISPLAY:SINTEGRITY:LAYOUT SPLIT"`
Query	`:DISPlay:SINTegrity:LAYout?`
Returned Format	`[:DISPlay:SINTegrity:LAYout] {SINGle	SPLit	QUAD}<NL>`

SINTegrity:LEVel

Command	`:DISPlay:SINTegrity:LEVel {ZERO	ONE	BOTH}`
	On amplitude graphs, displays results that are based on the one level, zero level, or both. Amplitude graphs are displayed using the "SINTegrity:GRAPh" on page 183.
Restrictions	86100C/D (software revision A.07.00 and above) with Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":DISPLAY:SINTEGRITY:LEVEL ONE"`
Query	`:DISPlay:SINTegrity:LEVel?`
Returned Format	`[:DISPlay:SINTegrity:LEVel] {ZERO	ONE	BOTH}<NL>`

SINTegrity:SHADe

Command	`:DISPlay:SINTegrity:SHADe {ON	1	OFF	0}`
	Shows or removes the display of the Jitter or Noise shade. The shade is the drop-down screen that is used to display the jitter, noise, and interference graphs. Because updating the plots takes some time, use this command to reduce measurement times when display of the data is not essential. When writing new code, this is the recommended replacement for the command "JITTer:SHADe" on page 179.
Restrictions	86100C/D (software revision A.07.00 and above) with Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":DISPLAY:SINTEGRITY:SHADE ON"`
Query	`:DISPlay:SINTegrity:SHADe?`
Returned Format	`[:DISPlay:SINTegrity:SHADe] {ON	OFF} <NL>`

SPARameter:GRAPh

Command	`:DISPlay:SPARameter:GRAPh {MAGGraph	PGRaph	GDGRaph},[{MAGGraph	PGRaph	GDGRaph}]`
	Selects which graphs to display in the S-parameter shade window. If one graph is specified, it is placed on the top half of the shade window. If two graphs are specified, the second graph is placed on the bottom half of the shade window.
Restrictions	Software revision A.08.00 and above (86100C instruments) or an 86100D.
Example	`10 OUTPUT 707; ":DISP:SPAR:GRAP PGR"`
Query	Returns a list of the currently displayed graphs. The returned values are comma-separated and listed in the order that they were turned on. The first value is the most recently selected graph.

```
                  :DISPlay:SPARameter:GRAPh?
```

Returned Format   [:DISPlay:SPARameter:GRAPh] {MAGGraph | PGRaph | GDGRaph},[{MAGGraph | PGRaph | GDGRaph}]<NL>

SPARameter:LAYout

Command   :DISPlay:SPARameter:LAYout {SINGle|SPLit}

Sets the S-parameter shade window layout to single or split modes.

Restrictions   Software revision A.08.00 and above (86100C instruments) or an 86100D.

Example   10 OUTPUT 707; ":DISP:SPAR:LAY SPL"

Query   :DISPlay:SPARameter: LAYout?

Returned Format   [:DISPlay:SPARameter:LAYout] {SINGle | SPLit }<NL>

SPARameter:SHADe

Command   :DISPlay:SPARameter:SHADe {{ON | 1}|{OFF | 0}}

Turns on and off the S-parameter measurements, which also displays or hides the S-parameter graphs. Because the S-parameter calculations occure only when the graph shade is displayed, the graph must be displayed before S-parameter data can be saved to a file.

Restrictions   Software revision A.08.00 and above (86100C instruments) or an 86100D.

Example   10 OUTPUT 707;":DISP:SPAR:SHAD ON"

Query   :DISPlay:SPARameter:SHADe?

Returned Format   [:DISPlay:SPARameter:SHADe] {{ON | 1}|{OFF | 0}}<NL>

SSAVer

Commands   :DISPlay:SSAVer {DISabled|ENABled}
:DISPlay:SSAVer:AAFTer <time>

Disables or enables an 86100A/B instrument's screen saver and specifies a time before the screen saver turns on. <time> is An integer; either 2, 3, 4, 5, 6, 7, or 8. The time value specifies the amount of time, in hours, that must pass before the screen saver will turn on.

Restrictions   86100A/B only. The 86100C/D screen saver is controlled from the operating system.

Example   10 OUTPUT 707;":DISPLAY:SSAVER ENABLED"
20 OUTPUT 707;":DISPLAY:SSAVER:AAFT 4"

Queries   :DISPlay:SSAVer?
:DISPlay:SSAVer:AAFTer?

Returned Format   [:DISPlay:SSAVer] {DISabled|ENABled}<NL>
[:DISPlay:SSAVer:AAFTer] <time><NL>

12 Function Commands

The FUNCtion subsystem defines up to four functions: 1 through 4. The function is indicated in the FUNCtion<N> syntax, for example FUNCtion1. Use the following commands (math operators) to define a funtion: ADD, DIFF, INVert, MAGNify, MAXimum, MINimum, MULTiply, PEELing, SUBTract, and VERSus. The functions operands can be any of the installed channels, waveform memories (1 through 4), functions (1 through 4), or a constant and have the following characteristics:

- If a channel is not on but is used as an operand, then that channel will acquire waveform data.
- If the source waveforms have different record lengths, the function is performed over the shorter record length. The instrument finds the nearest point in the longer waveform record that corresponds to the current point in the shorter record. It then performs math functions on those points and skips non-corresponding points in the longer record.
- If the two sources have the same time base scale, the resulting function has the same time scale which results in the same time base scale for the function. If the sources cover two different time intervals, the function is performed on the portion of the sources that overlap. If the sources don't overlap, the function cannot be performed.

KEYSIGHT
TECHNOLOGIES

- If the operands have different time scales, the resulting function has no valid time scale. This is because operations are performed based on the displayed waveform data position, and the time relationship of the data records cannot be considered. When the time scale is not valid, delta time pulse parameter measurements have no meaning, and the unknown result indicator is displayed on the screen.

- Numeric constant sources have the same horizontal scale as the associated waveform source.

- You can use a function as a source for another function subject to the following constraints:

- F4 can have F1, F2, or F3 as a source.

- F3 can have F1 or F2 as a source.

- F2 can have F1 as a source.

- F1 cannot have any other function as a source.

Use the RANGe and OFFSet commands in this subsystem control the vertical scaling and offset. Use the HORizontal:RANge and HORizontal:POSition queries to obtain horizontal scaling and position values.

ADD

Command   `:FUNCtion<N>:ADD <operand>,<operand>`

Defines a function that adds source 1 to source 2, point by point, and places the result in the selected function waveform. When vertical scaling is set to Auto, the instrument automatically sets vertical scale and offset to display the entire function on the display. Any changes to vertical scale or offset to the source waveform are tracked. In Manual mode, you set the function's vertical scale and offset; tracking is disabled.

Restrictions   Not available in Jitter mode.

<operand>   `{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N> | <float_value>}`

Example   `10 OUTPUT 707;":FUNCTION1:ADD CHANNEL1,WMEMORY1"`

DIFF

Command   `:FUNCtion<N>:DIFF <operand>`

Defines a function that differentiates source 1 and places the result in the selected function waveform. Differential is only available in TDR/TDT Mode.

Restrictions   Available only in TDR/TDT mode.

<operand>   `{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N> | <float_value>}`

Example   `10 OUTPUT 707;":FUNCTION1:DIFF CHANNEL1"`

DISPlay

Command   `:FUNCtion<N>:DISPlay {{ON | 1} | {OFF | 0}}[,APPend]`

This command either displays the selected function or removes it from the display. The APPend argument is used to turn on additional functions in Eye/Mask mode without turning off any other database signals that are currently on. Without the APPend parameter, all other database signals would be turned off when turning a function on.

Example   `10 OUTPUT 707;":FUNCTION1:DISPLAY ON"`

| Query | `:FUNCtion<N>:DISPlay?` |

The query returns the displayed status of the specified function.

| Returned Format | `[:FUNCtion<N>:DISPlay] {1 | 0}[,APPend]<NL>` |

FUNCtion

| Query | `:FUNCtion<N>?` |

This query returns the currently defined source(s) for the function.

| Returned Format | `[:FUNCtion<N>:<operator>] {<operand> [,<operand>]}<NL>` |

The <operator> is any active math operation for the selected function. The <operand> is any allowable source for the selected FUNCtion, including channels, waveform memories, or functions. If the function is applied to a constant, the source returns the constant.

| Example | `10 OUTPUT 707;":FUNCTION1?"` |

If the headers are off (see :SYSTem:HEADers), the query returns only the operands, not the operator.

```
10 :SYST:HEADER ON
20 :FUNC1:SUBTRACT CHAN1,CHAN2
30 :FUNC1? !returns :FUNC1:SUBTRACT CHAN1,CHAN2
40 :SYST:HEADER OFF
50 :FUNC1? !returns CHAN1,CHAN2
```

HORizontal

| Command | `:FUNCtion<N>:HORizontal {AUTO | MANual}` |

Sets the horizontal tracking to either AUTO or MANual. The HORizontal command also includes a subsystem consisting of the commands POSition and RANGe.

| Restrictions | Applies only to the Magnify and Versus operators. On software revisions A.06.00 and above, using this function on operators other than Magnify or Versus returns the error message "–224, Illegal parameter value". On software revisions below A.06.00, the error message is not returned. |

| Query | `:FUNCtion<N>:HORizontal?` |

The query returns the current horizontal scaling mode of the specified function.

| Returned Format | `[:FUNCtion<N>:HORizontal] {AUTO | MANual}<NL>` |

| Example | `10 OUTPUT 707;":FUNCTION1:HORIZONTAL?"` |

HORizontal:POSition

| Command | `:FUNCtion<N>:HORizontal:POSition <position_value>` |

This command sets the time value at center screen for the selected function. The <position_value> argument is the position value in time, in seconds.

| Restrictions | Applies only to the Magnify and Versus operators. If this function is used on operators other than Magnify or Versus, no error message is returned regardless of software revision. |

| Query | `:FUNCtion<N>:HORizontal:POSition?` |

The query returns the current time value at center screen of the selected function.

| Returned Format | `[:FUNCtion<N>:HORizontal:POSition] <position><NL>` |

Example   This example places the current horizontal position setting for function 2 in the numeric variable, Value.

```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":FUNCTION2:DISPLAY ON"
30 OUTPUT 707;":FUNCTION2:HORIZONTAL:POSITION?"
40 ENTER 707;Value
```

HORizontal:RANGe

Command   `:FUNCtion<N>:HORizontal:RANGe <range_value>`

Sets the current time range for the specified function. This automatically selects manual mode. <range_value> is the width of screen in current X-axis units (usually seconds).

Restrictions   This command applies only to the Magnify and Versus operators. If this function is used on operators other than Magnify or Versus, no error message is returned regardless of software revision.

Query   `:FUNCtion<N>:HORizontal:RANGe?`

The query returns the current time range setting of the specified function.

NOTE   This query returns the current time range setting of the specified function only when the respective function display is ON.

Returned Format   `[:FUNCtion<N>:HORizontal:RANGe] <range><NL>`

Example
```
20 OUTPUT 707;":FUNCTION2:DISPLAY ON"
30 OUTPUT 707;":FUNCTION2:HORIZONTAL:RANGE?"
```

INVert

Command   `:FUNCtion<N>:INVert <operand>`

Defines a function that inverts the defined operand's waveform by multiplying by –1.

Restrictions   Not available in Jitter mode.

<operand>   `{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N> | <float_value>}`

Example   This example sets up function 2 to invert the signal on channel 1.
```
10 OUTPUT 707;":FUNCTION2:INVERT CHANNEL1"
```

MAGNify

Command   `:FUNCtion<N>:MAGNify <operand>`

Defines a function that is a copy of the operand. The magnify function is a software magnify. No hardware settings are altered as a result of using this function. It is useful for scaling channels, another function, TDR/TDT responses or memories with the RANGe and OFFSet commands in this subsystem.

<operand>   `{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N> | <float_value>}`

Example   `10 OUTPUT 707;":FUNCTION1:MAGNIFY CHANNEL1"`

MAXimum

Command   `:FUNCtion<N>:MAXimum <operand>`

Defines a function that computes the maximum value of the operand waveform in each time bucket.

Restrictions   Not available in Jitter mode.

<operand>   `{CHANnel<N> | FUNCtion<N> | WMEMory<N> | <float_value>}`

Example   `10 OUTPUT 707;":FUNCTION1:MAXIMUM CHANNEL1"`

MINimum

Command   `:FUNCtion<N>:MINimum <operand>`
Defines a function that computes the minimum value of each time bucket for the defined operand's waveform.

Restrictions   Not available in Jitter mode.

<operand>   `{CHANnel<N> | FUNCtion<N> | WMEMory<N> | <float_value>}`

Example   `10 OUTPUT 707;":FUNCTION1:MINIMUM CHANNEL1"`

MULTiply

Command   `:FUNCtion<N>:MULTiply <operand>,<operand>`
Defines a function that multiplies source 1 by source 2, point by point, and places the result in the selected function waveform. When vertical scaling is set to Auto, the instrument automatically sets vertical scale and offset to display the entire function on the display. Any changes to vertical scale or offset to the source waveform are tracked. In Manual mode, you set the function's vertical scale and offset; tracking is disabled.

Restrictions   Not available in Jitter mode.

<operand>   `{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N> | <float_value>}`

Example   This example defines a function that subtracts waveform memory 1 from channel 1.
`10 OUTPUT 707;":FUNCTION1:MULTIPLY CHANNEL1,WMEMORY1"`

OFFSet

Command   `:FUNCtion<N>:OFFSet <offset_value>`
Sets the voltage represented at the center of the screen for the selected function. This automatically changes the mode from auto to manual. <offset_value> is limited to being within the vertical range that can be represented by the function data.

Example   This example sets the offset voltage for function 1 to 2 mV.
`10 OUTPUT 707;":FUNCTION1:OFFSET 2E-3"`

Query   `:FUNCtion<N>:OFFSet?`
The query returns the current offset value for the selected function.

> **NOTE**   This query returns the current offset value of the specified function only when the respective function display is ON.

Returned Format   `[:FUNCtion<N>:OFFSet] <offset_value><NL>`

| **Example** | ```
20 OUTPUT 707;":FUNCTION2:DISPLAY ON"
30 OUTPUT 707;":FUNCTION2:OFFSET?"
``` |

PEELing

Command	`:FUNCtion<N>:PEELing <operand>`

Defines a function that applies TDR peeling to source 1 and places the result in the selected function waveform. The TDR peeling is provided with Option 202, Enhanced Impedance and S-Parameter software, and is used in TDR mode to analyze reflected signals at the source and deconvolve the time domain reflections to create an impedance profile of the device being tested. For differential and common mode responses, apply the TDR peeling to the differential or common-mode response trace. TDR peeling can not be applied to TDT responses. TDR Peeling is only available in TDR/TDT Mode.

Restrictions	Available only in TDR/TDT mode. Software revision A.06.00 and above (86100C instruments) or an 86100D.
<operand>	`{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>}`
Example	`10 OUTPUT 707;":FUNCTION1:PEELING CHANNEL1,WMEMORY1"`

RANGe

Command	`:FUNCtion<N>:RANGe <full_scale_range>`

Defines the full scale vertical axis of the selected function. This automatically changes the mode from auto to manual. <full_scale_range> is the full-scale vertical range.

Example	This example sets the full scale range for function 1 to 400 mV. `10 OUTPUT 707;":FUNCTION1:RANGE 400E-3"`
Query	`:FUNCtion<N>:RANGe?` This query returns the current full scale range setting of the specified function only when the respective function display is ON.
Returned Format	`[:FUNCtion<N>:RANGe] <full_scale_range><NL>`
| **Example** | ```
20 OUTPUT 707;":FUNCTION2:DISPLAY ON"
30 OUTPUT 707;":FUNCTION2:RANGE?"
``` |

SUBTract

Command	`:FUNCtion<N>:SUBTract <operand>,<operand>` Defines a function that algebraically subtracts the second operand from the first operand.
<operand>	`{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N> | <float_value>}`
Example	This example defines a function that subtracts waveform memory 1 from channel 1. `10 OUTPUT 707;":FUNCTION1:SUBTRACT CHANNEL1,WMEMORY1"`

VERSus

Command	`:FUNCtion<N>:VERSus <operand>,<operand>`

This command defines a function for an X-versus-Y display. The first operand defines the Y axis and the second defines the X axis. The Y-axis range and offset are initially equal to that of the first operand and can be adjusted with the RANGe and OFFSet commands in this subsystem.

Restrictions	Not available in Jitter mode.
\<operand\>	`{CHANnel<N>	FUNCtion<N>	RESPonse<N>	WMEMory<N>	<float_value>}`
Example	This example defines function 1 as an X-versus-Y display. Channel 1 is the X axis and waveform memory 2 is the Y axis. `10 OUTPUT 707;":FUNCTION1:VERSUS WMEMORY2,CHANNEL1"`

VERTical

Command	`:FUNCtion<N>:VERTical {AUTO	MANual}` Sets the vertical scaling mode of the specified function to either AUTO or MANual. The VERTical command also includes a subsystem consisting of the commands POSition and RANGe.
Query	`:FUNCtion<N>:VERTical?` The query returns the current vertical scaling mode of the specified function.
Returned Format	`[:FUNCtion<N>:VERTical] {AUTO	MANual}<NL>`
Example	`10 OUTPUT 707;":FUNCTION1:VERTICAL?"`

VERTical:OFFSet

Command	`:FUNCtion<N>:VERTical:OFFSet <offset_value>` Sets the voltage represented at center screen for the selected function. This automatically changes the mode from auto to manual. \<offset_value\> is the offset value and is limited only to being within the vertical range that can be represented by the function data.
Query	`:FUNCtion<N>:VERTical:OFFset?` The query returns the current offset value of the selected function.

NOTE   This query returns the current offset value of the specified function only when the respective function display is ON.

Returned Format	`[:FUNCtion<N>:VERTical:OFFset] <offset_value><NL>`
Example	`10 OUTPUT 707;":FUNCTION2:DISPLAY ON"` `30 OUTPUT 707;":FUNCTION2:VERTICAL:OFFSET?"`

VERTical:RANGe

Command	`:FUNCtion<N>:VERTical:RANGe <full_scale_range>` Defines the full-scale vertical axis of the selected function. This automatically changes the mode from auto to manual, if the scope is not already in manual mode. \<full_scale_range\> is the full-scale vertical range.
Query	`:FUNCtion<N>:VERTical:RANGe?` This query returns the current range setting of the specified function only when the respective function display is ON.
Returned Format	`[:FUNCtion<N>:VERTical:RANGe] <range><NL>`
Example	`10 OUTPUT 707;":FUNCTION2:DISPLAY ON"` `20 OUTPUT 707;":FUNCTION2:VERTICAL:RANGE?"`

13 Hardcopy Commands

The HARDcopy subsystem commands set various parameters for printing the screen. The print sequence is activated when the root level :PRINt command is sent.

AREA

Command   `:HARDcopy:AREA {GRATicule | SCReen}`

Selects which data from the screen is to be printed. When you select GRATicule, only the graticule area of the screen is printed (this is the same as choosing Waveforms Only in the Configure Printer dialog box). When you select SCReen, the entire screen is printed.

Example   This example selects the graticule for printing.
```
10 OUTPUT 707;":HARDCOPY:AREA GRATICULE"
```

Query   `:HARDcopy:AREA?`

The query returns the current setting for the area of the screen to be printed.

Returned Format   `[:HARDcopy:AREA] {GRATicule | SCReen}<NL>`

Example   This example places the current selection for the area to be printed in the string variable, Selection$.
```
10 DIM Selection$[50]!Dimension variable
20 OUTPUT 707;":HARDCOPY:AREA?"
30 ENTER 707;Selection$
```

DPRinter

Command   `:HARDcopy:DPRinter {<printer_number>|<printer_string>}`

Selects the default printer to be used. <printer_number> is an integer representing the attached printer. This number corresponds to the number returned with each printer name by the :HARDcopy:PRINters? query. <printer_string> is a string of alphanumeric characters representing the attached printer. The HARDcopy:DPRinter command specifies a number or string for the printer attached to the analyzer. The printer_string must exactly match the character strings in the File, Print Setup dialog boxes, or the strings returned by the :HARDcopy:PRINters? query.

Examples   This example sets the default printer to the second installed printer returned by the :HARDcopy:PRINters? query.
```
10 OUTPUT 707;":HARDCOPY:DPRINTER 2"
```

KEYSIGHT TECHNOLOGIES

This example sets the default printer to the installed printer with the name "Laser".
```
10 OUTPUT 707;":HARDCOPY:DPRINTER "Laser""
```

Query   `:HARDcopy:DPRinter?`

The query returns the current printer number and string.

Returned Format   `[:HARDcopy:DPRinter?] {<printer_number>,<printer_string>,DEFAULT}<NL>`

Or, if there is no default printer (no printers are installed), only a <NL> is returned.

Example   This example places the current setting for the hardcopy printer in the string variable, Setting$.
```
10 DIM Setting$[50]!Dimension variable
20 OUTPUT 707;":HARDCOPY:DPRinter?"
30 ENTER 707;Setting$
```

It takes several seconds to change the default printer. Any programs that try to set the default printer must wait (10 seconds is a safe amount of time) for the change to complete before sending other commands. Otherwise the analyzer will become unresponsive.

FACTors

Command   `:HARDcopy:FACTors {{ON | 1}|{OFF | 0}}`

Determines whether the instrument setup factors will be appended to screen or graticule images. FACTors ON is the same as choosing Include Setup Information in the Configure Printer dialog box.

Example   This example turns on the setup factors.
```
10 OUTPUT 707;":HARDCOPY:FACTORS ON"
```

Query   `:HARDcopy:FACTors?`

The query returns the current setup factors setting.

Returned Format   `[:HARDcopy:FACTors] {1|0}<NL>`

Example   This example places the current setting for the setup factors in the string variable, Setting$.
```
10 DIM Setting$[50]!Dimension variable
20 OUTPUT 707;":HARDCOPY:FACTORS?"
30 ENTER 707;Setting$
```

IMAGe

Command   `:HARDcopy:IMAGe {NORMal | INVert | MONochrome}`

Prints the image normally, inverted, or in monochrome. IMAGe INVert is the same as choosing Invert Waveform Colors in the Configure Printer dialog box.

Example   This example sets the hardcopy image output to normal.
```
10 OUTPUT 707;":HARDCOPY:IMAGE NORMAL"
```

Query   `:HARDcopy:IMAGe?`

The query returns the current image setting.

Returned Format   `[:HARDcopy:IMAGe] {NORMal | INVert | MONochrome}<NL>`

Example   This example places the current setting for the hardcopy image in the string variable, Setting$.
```
10 DIM Setting$[50]!Dimension variable
20 OUTPUT 707;":HARDCOPY:IMAGE?"
30 ENTER 707;Setting$
```

PRINters?

Query   `:HARDcopy:PRINters?`

This query returns the currently available printers.

Returned Format   `[:HARDcopy:PRINters]<printer_count><NL><printer_data><NL>[,<printer_data><NL>]`

<printer_count> is the number of printers currently installed. <printer_data> is the printer number and the name of an installed printer. The word DEFAULT appears next to the printer that is the currently selected default printer.

Example   This example places the number of installed printers into the variable Count, loops through that number of times, and prints the installed printer names to the computer screen.

```
10 DIM Setting$[50]!Dimension variable
20 OUTPUT 707;":HARDCOPY:PRINTERS?"
30 ENTER 707;Count
40 IF Count>0 THEN
50 FOR Printer_number=1 TO Count
60 ENTER 707;Setting$
70 PRINT Setting$
80 NEXT Printer_number
90 END IF
100 END
```

14 Histogram Commands

The Histogram commands and queries control the histogram features. A histogram is a probability distribution that shows the distribution of acquired data within a user-definable histogram window. You can display the histogram either vertically, for voltage measurements, or horizontally, for timing measurements. The most common use for histograms is measuring and characterizing noise or jitter on displayed waveforms. Noise is measured by sizing the histogram window to a narrow portion of time and observing a vertical histogram that measures the noise on a waveform. Jitter is measured by sizing the histogram window to a narrow portion of voltage and observing a horizontal histogram that measures the jitter on an edge.

To return the histogram's source data, use the :WAVeform:SOURce command to select the HISTogram as the source and then use the :WAVeform:DATA? query to retrieve the histogram data.

The histograms, mask testing, and color-graded (including gray scale) display use a specific database that uses a different memory area from the waveform record for each channel. When any of these features are turned on, the instrument starts building the database. The database is the size of the graticule area. Behind each pixel is a 16-bit counter that is incremented each time data from a channel or function hits a pixel. The maximum count (saturation) for each counter is 63,488. You can use the :MEASure:CGRade:PEAK? or DISPlay:CGRade:LEVels? queries to see if any of the counters are close to saturation.

The database continues to build until the instrument stops acquiring data or all three functions (color-graded display, mask testing, and histograms) are turned off. You can set the ACQuisition:RUNTil (Run Until) mode to stop acquiring data after a specified number of waveforms or samples are acquired. You can clear the database by turning off all three features that use the database.

The database does not differentiate waveforms from different channels or functions. If three channels are turned on and the waveform from each channel happens to light the same pixel at the same time, the counter is incremented by three. However, it is not possible to tell how many hits came from each waveform. To separate waveforms, you can set the display to two graphs or position the waveforms vertically with the channel offset. By separating the waveforms, you can avoid overlapping data in the database caused by multiple waveforms.

KEYSIGHT
TECHNOLOGIES

Suppose that the database is building because color-graded display is ON; when mask testing or histograms are turned on, they can use the information already established in the database as though they had been turned on the entire time. To avoid erroneous data, clear the display after you change instrument setup conditions or device under test (DUT) conditions and acquire new data before extracting measurement results.

AXIS

Command	`:HISTogram:AXIS {VERTical	HORizontal}`
	Creates a histogram with a horizontal or vertical axis.
Example	`10 OUTPUT 707;":HISTOGRAM:AXIS VERTICAL"`
Query	`:HISTogram:AXIS?`
	Returns the currently selected histogram axis.
Returned Format	`[:HISTogram:AXIS] {VERTical	HORizontal} <NL>`

Example	`10 DIM Axis$[50]`
	`20 OUTPUT 707;":HISTOGRAM:AXIS?"`
	`30 ENTER 707;Axis$`

MODE

Command	`:HISTogram:MODE {ON	OFF	WAVeform}`
	Selects the histogram mode, off or on, to track the waveform database. WAVeform is the same as ON and exists for backward compatibility.

NOTE   Do not use this command in Jitter Mode. It generates a "Control is set to default" error.

Example	`10 OUTPUT 707;":HISTOGRAM:MODE ON"`
Query	`:HISTogram:MODE?`
	Returns the currently selected histogram mode.
Returned Format	`[:HISTogram:MODE] {ON	OFF } <NL>`

SCALe:SIZE

Command	`:HISTogram:SCALe:SIZE <size> [,{HORizontal	VERTical}]`
	Sets the histogram size for vertical and horizontal mode. <size> is the size and can range from 1.0 to 8.0 for the horizontal mode and from 1.0 to 10.0 for the vertical mode. Separate values are maintained for each axis. If the optional axis parameter is not specified, the size of the current axis is set.
Example	`10 OUTPUT 707;":HISTOGRAM:SCALE:SIZE 3.5"`
Query	`:HISTogram:SCALe:SIZE? [HORizontal	VERTical]`
	Returns the correct size of the histogram.
Returned Format	`[:HISTogram:SCALe:SIZE] <size><NL>`

SOURce

Command `:HISTogram:SOURce {CHANnel<N> | FUNCtion<N> | RESPonse<N> | CGMemory}`

Selects the source of the histogram window. The histogram window will track the source's vertical and horizontal scale. If the optional append parameter is not used when a .cgs file is loaded, the window source is set to CGMemory. No other source may be selected until the histogram database is cleared. <N> is an integer, 1through 4.

Example `10 OUTPUT 707;":HISTOGRAM:SOURCE CHANNEL1"`

Query `:HISTogram:SOURce?`

Returns the currently selected histogram source.

Returned Format `[:HISTogram:SOURce] {CHANnel<N> | FUNCtion<N> | RESPonse<N> | CGM}<NL>`

WINDow:BORDer

Command `:HISTogram:WINDow:BORDer {ON | 1 | OFF | 0}`

Turns the display of the histogram window border on or off.

Example `10 OUTPUT 707;":HISTOGRAM:WINDOW:BORDER ON"`

Query `:HISTogram:WINDow:BORDer?`

Returned Format `[:HISTogram:WINDow:BORDer] {ON | OFF}<NL>`

WINDow:DEFault

Command `:HISTogram:WINDow:DEFault`

Positions the histogram markers to a default location on the display. Each marker will be positioned one division off the left, right, top, and bottom of the display.

Example `10 OUTPUT 707;":HISTogram:WINDow:DEFault"`

WINDow:SOURce

Command `:HISTogram:WINDow:SOURce {CHANnel<N> | FUNCtion<N> | RESPonse<N> | CGMemory}`

Selects the source of the histogram window. The histogram window will track the source's vertical and horizontal scale. If the optional append parameter is not used when a .cgs file is loaded, the window source is set to CGMemory. No other source may be selected until the histogram database is cleared. <N> is an integer, 1through 4. This command serves the same function as the HISTogram:SOURce command and has been retained for compatibility with the Agilent 83480A/54750A.

Example `10 OUTPUT 707;":HISTOGRAM:WINDOW:SOURCE CHANNEL1"`

Query `:HISTogram:WINDow:SOURce?`

Returned Format `[:HISTogram:WINDow:SOURce] {CHANnel<N> | FUNCtion<N> | RESPonse<N> | CGM}<NL>`

WINDow:X1Position

Command `:HISTogram:WINDow:X1Position <X1 position>`

Moves the X1 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the histogram window source.

Example   The following example sets the X1 position to –200 microseconds.
```
10 OUTPUT 707;":HISTOGRAM:WINDOW:X1POSITION -200E-6"
```

Query   `:HISTogram:WINDow:X1Position?`

Returned Format   `[:HISTogram:WINDow:X1Position]<X1 position><NL>`

WINDow:X2Position

Command   `:HISTogram:WINDow:X2Position <X2 position>`

Moves the X2 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the histogram window source.

Example   The following example sets the X2 marker to 200 microseconds.
```
10 OUTPUT 707;":HISTOGRAM:WINDOW:X2POSITION 200E-6"
```

Query   `:HISTogram:WINDow:X2Position?`

Returned Format   `[:HISTogram:WINDow:X2Position] <X2 position><NL>`

WINDow:Y1Position

Command   `:HISTogram:WINDow:Y1Position <Y1 position>`

Moves the Y1 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the histogram window source.

Example   The following example sets the position of the Y1 marker to –250 mV.
```
10 OUTPUT 707;":HISTOGRAM:WINDOW:Y1POSITION -250E-3"
```

Query   `:HISTogram:WINDow:Y1Position?`

Returned Format   `[:HISTogram:WINDow:Y1Position] <Y1 position><NL>`

WINDow:Y2Position

Command   `:HISTogram:WINDow:Y2Position <Y2 position>`

Moves the Y2 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the histogram window source.

Example   The following example sets the position of the Y2 marker to 1.
```
10 OUTPUT 707;":HISTOGRAM:WINDOW:Y2POSITION 1"
```

Query   `:HISTogram:WINDow:Y2Position?`

Returned Format   `[:HISTogram:WINDow:Y2Position] <Y2 position><NL>`

15 Limit Test Commands

The Limit Test commands and queries control the instrument's limit test features. Limit testing automatically compares measurement results with pass or fail limits. The limit test tracks up to four measurements. The action taken when the test fails is also controlled with commands in this subsystem.

FAIL

Command   `:LTESt:FAIL {INSide | OUTSide | ALWays | NEVer}`

Sets the fail condition for an individual measurement. The conditions for a test failure are set on the source selected with the last LTESt:SOURce command. When a measurement failure is detected by the limit test, the fail action conditions are executed, and there is the potential to generate an SRQ.

The argument INSide causes the instrument to fail a test when the measurement results are within the parameters set by the LTESt:LLIMit and LTESt:ULIMit commands. OUTside causes the instrument to fail a test when the measurement results exceed the parameters set by LTESt:LLIMit and LTESt:ULIMit commands. ALWays ALWays causes the instrument to fail a test every time the measurement is executed, and the parameters set by the LTESt:LLIMit and LTESt:ULIMit commands are ignored. The FAIL:ALWays mode logs the action each time the measurement is executed. FAIL:ALWays can monitor trends in measurements, for example, tracking a measurement during an environmental test while the instrument is running a measurement for a long time, as the temperature or humidity is changed. Each time the measurement is executed, the results are logged as determined by the fail action set with the LTESt:SSCReen, LTESt:SSUMmary, or LTESt:SWAVeform commands. NEVer sets the instrument so a measurement never fails a test. Use the FAIL:NEVer mode to observe one measurement but determine a failure from a different measurement. The FAIL:NEVer mode monitors a measurement without any fail criteria.

KEYSIGHT
TECHNOLOGIES

Query	`:LTESt:FAIL?`
	The query returns the current value set for the fail condition.

Returned Format	`[:LTESt:FAIL] {INSIDELIMITS| OUTSIDELIMITS| ALWAYSFAIL| NEVERFAIL}<NL>`

Example	`10 OUTPUT 707;":LTEST:FAIL OUTSIDE"`

JITTer:SELect

Command	`:LTESt:JITTer:SELect {TJ|DJ|RJ|PJ|PJRMS|DDJ|ISI|DCD}`
	This command selects a measurement for measurement limit testing in Jitter Mode. Up to four measurements at a time may be limit tested. This requires using the command four times, as each issue of the command selects one measurement. Executing this command when four measurements are already selected causes the oldest measurement selection to be cleared and the new measurement to be added. All measurements may be cleared by executing the :MEASure:CLEar command. Use the :MEASure:RESults? query to get the names of the currently selected measurements.

Restrictions	When writing new code for software revision A.07.00 and above, use the recommended command "SINTegrity:SELect" on page 205. The JITTer:SELect command applies to software revision A.04.00 through A.06.01.

Example	The following example selects the total jitter measurement for limit testing.
	`10 OUTPUT 707;":LTEST:JITTer:SELect TJ"`

LLIMit

Command	`:LTESt:LLIMit <lower_value>`
	This command sets the lower test limit for the active measurement currently selected by the :LTESt:SOURce command. <lower_value> is a real number. For example, if you chose to measure volts peak-peak and want the smallest acceptable signal swing to be one volt, you could use a <lower_value> of 1, then set the limit test to fail when the signal is outside the specified limit.

Query	`:LTESt:LLIMit?`
	The query returns the current value set by the command.

Returned Format	`[:LTESt:LLIMit]<lower_value><NL>`

Example	`10 OUTPUT 707;":LTEST:LLIMIT 1"`

MNFound

Command	`:LTESt:MNFound {FAIL	PASS	IGNore}`
	This command sets the action to take when the measurement cannot be made. This command affects the active measurement currently selected by the last LTESt:SOURce command. This command tells the instrument how to treat a measurement that cannot be made. For example, if a risetime between 1 to 5 volts is requested and the captured signal is between 2 to 3 volts, this control comes into play. Another use for this command is when trying to measure the frequency of a baseline waveform.
	FAIL is used when the instrument cannot make a measurement, for example, when an edge is expected to be present and is not found. This is the mode to use for most applications.
	The total number of waveforms is incremented, and the total number of failures is incremented.

PASS might be used when triggering on one event and measuring another event which may not occur for every trigger. For example, in a communications test system, you might want to trigger on the clock and test the risetime of edges in the data stream. However, there may be no way to guarantee that a rising edge will be present to measure in the data stream at every clock edge. By using the PASS parameter, the limit test will not log a failure if there is no edge found in the data stream.

If the measurement cannot be made, the total number of waveforms measured is incremented, but the total number of failures is not.

IGNore is similar to PASS, except the totals for the number of waveforms and failures are not incremented. Therefore, the total indicates the number of tests when the measurement was made.

Query   `:LTESt:MNFound?`

The query returns the current action set by the command.

Returned Format   `[:LTESt:MNFound] {FAIL | PASS | IGNore}<NL>`

Example   `10 OUTPUT 707;":LTEST:MNFOUND PASS"`

RUNTil

Command   `:LTESt:RUNTil FAILures, <total_failures>`

This command determines the termination conditions for the test. The keywords RUN or RUMode (Run Until Mode) may also be used. This command is compatible with the Agilent83480/54750. To run for a number of waveforms or samples, refer to ACQuire:RUNTil command on .

FAILures   FAILures runs the limit test until a set number of failures occur. When FAILures is sent, the test executes until the selected total failures are obtained. The number of failures are compared against this number to test for termination. Use the FAILures mode when you want the limit test to reach completion after a set number of failures. The total number of failures is additive for all of the measurements. For example, if you select 10 failures, the total of 10 failures can come from several measurements. The 10 failures can be the sum of four rise time failures, four +width failures, and two overshoot failures.

<total_failures>   An integer: 1 to 1,000,000,000.

Example   The following example causes limit test to run until two failures occur.
```
10 OUTPUT 707;":LTEST:RUNTil FAILures, 2"
20 OUTPUT 707;":RUN
```

Query   `:LTESt:RUNTil?`

The query returns the currently selected termination condition and value.

Returned Format   `[:LTESt:RUNTil] {FAILures, <total_failures>}<NL>`

SINTegrity:SELect

Command

```
:LTESt:SINTegrity:SELect {RJ | PJ | PJRMS | DJ | TJ | DDJ | DCD | ISIJ | TI1 | TI0
| RN1 | RN0 | DI1 | DI0 | ISI1 | ISI0 | PI1 | PI0 | PIRMs1 | PIRMs0 | SAMplitude |
EOPening | Q | RINoise | BERFloor}
```

Selects an amplitude measurement for measurement limit testing. Note that the last character for the command arguments TI0, RN0, DI0, ISI0, PI0, and PIRMs0 is the numeral 0 (zero) representing a "low" signal level and not the capital letter "O" (oh).

Up to four measurements at a time may be limit tested. This requires using the command four times, as each issue of the command selects one measurement. Executing this command when four measurements are already selected causes the oldest measurement selection to be cleared and the new measurement to be added. All measurements may be cleared by executing the :MEASure:CLEar command. Use the :MEASure: RESults? query to get the names of the currently selected measurements. When writing new code, this command (SINTegrity) is the recommended replacement for the command **"JITTer:SELect"** on page 204.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Returned Format   `[:LTESt:SINTegrity:SELect] {RJ | PJ | PJRMS | DJ | TJ | DDJ | DCD | ISIJ | TI1 | TI0 | RN1 | RN0 | DI1 | DI0 | ISI1 | ISI0 | PI1 | PI0 | PIRMs1 | PIRMs0 | SAMplitude | EOPening | Q | RINoise | BERFloor}<NL>`

Example   `10 OUTPUT 707;":LTEST:SINTEGRITY:SELECT RN"`

SOURce

Command   `:LTESt:SOURce {1 | 2 | 3 | 4}`

Selects the current source for the ULIMit, LLIMit, MNFound, and FAIL commands. It selects one of the active measurements as referred to by their position in the measurement window on the bottom of the screen. Source 1 is the measurement on the top line, 2 is on the second line, and so on. As a measurement is activated, the associated measurement limit test is programmed according to default values expressed by the following script:

```
:LTESt:SOURce <N>
:LTESt:FAIL OUTSIde
:LTESt:LLIMIt -10
:LTESt:ULIMIt 10
:LTESt:MNFound FAIL
:LTESt:RUNTil FAILUres, 1
```

Before a measurement limit test is initiated, you must make the necessary adjustments to the default values otherwise these values will be used during the limit test.

Example   The following example selects the first measurement as the source for the limit testing commands.
`10 OUTPUT 707;":LTEST:SOURCE 1"`

Query   `:LTESt:SOURce?`

The query returns the currently selected measurement source.

Returned Format   `[:LTESt:SOURce] {1 | 2 | 3 | 4} <NL>`

Example   The following example returns the currently selected measurement source for the limit testing commands.
```
10 DIM SOURCE$[50]
20 OUTPUT 707;":LTEST:SOURCE?"
30 ENTER 707;SOURCE$
```

See Also   Measurements are started in the Measurement subsystem.

SSCReen

Command   `:LTESt:SSCReen {OFF | DISK [,<filename>]}`

Saves a copy of the screen in the event of a limit test failure. To capture a screen image at any time, use the command **"SIMage"** on page 170. To capture a screen image when the waveform acquisitions has completed as specified with the acquire subsystem (number of averages and the number of data points), use the command **"SSCReen"** on page 127.

The OFF argument turns off the save action. DISK saves a copy of the screen to disk in the event of a failure. <filename> is an ASCII string enclosed in quotations marks. If no filename is specified, a filename will be assigned. The default filename is *MeasLimitScreenX.bmp*, where X is an incremental number assigned by the instrument.

> **NOTE**
>
> The save screen options established by the commands LTESt:SSCReen DISK, LTESt:SSCReen:AREA, and LTESt:SSCReen:IMAG are stored in the instrument's memory and will be employed in consecutive save screen operations, until changed by the user. This includes the <filename> parameter for the LTESt:SSCReen DISK command. If the results of consecutive limit tests must be stored in different files, omit the <filename> parameter and use the default filename instead. Each screen image will be saved in a different file named *MeasLimitScreenX.bmp*, where X is an incremental number assigned by the instrument.

The *filename* field includes the network path and the directory in which the file will be saved, as well as the file format that will be used. The following is a list of valid file locations:

- Files can only be created within the folder "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

- Files can not be saved on the root folder of the D: drive (C: on 86100A/B).

- Files can not be saved on USB removable drives. To save files on a USB drive, use front-panel controls. *(Applies only to firmware version A.09.00 and below)*

- Using the command "CDIRectory" on page 165 to change the present working directory has no effect on the location of saved files.

Table 39   Example Filenames

File Name	File Saved in Directory...
"test.jpg"	D:\User Files\screen images\ This is the default folder. Filenames without a path are saved to this folder.
"subfolder\test.jpg"	D:\User Files\screen images\subfolder The subfolder must already exist before saving the file.
"D:\User Files\subfolder\test.jpg"	D:\User Files\subfolder The subfolder must already exist before saving the file.
"D:\User Files\test.jpg"	D:\User Files
"D:\test.jpg"	This is not a valid file location. The file is not saved.
"E:test4.eps"	File saved in the instrument's drive E:, that could be mapped to any disk in the network.
"\\computer-ID\d$\test3.bmp"	File saved in drive D: of computer "computer-ID", provided all permissions are set properly.

The default file type is a bitmap (.bmp). The following graphics formats are available by specifying a file extension: PCX files (.pcx), EPS files (.eps), Postscript files (.ps), JPEG (.jpg), TIFF (.tif) and GIF files (.gif).

Example   `10 OUTPUT 707;":LTEST:SSCREEN DISK"`

Query   `:LTESt:SSCReen?`

The query returns the current state of the SSCReen command.

Returned Format   `[:LTESt:SSCReen] {OFF | DISK [,<filename>]}<NL>`

Example   The following example returns the destination of the save screen when a failure occurs.
`20 OUTPUT 707;":LTESt:SSCREEN?"`

SSCReen:AREA

Command   `:LTESt:SSCReen:AREA {GRATicule | SCReen}`

This command selects which data from the screen is to be saved to disk when the run until condition is met. When you select GRATicule, only the graticule area of the screen is saved (this is the same as choosing Waveforms Only in the Specify Report Action for measurement limit test dialog box). When you select SCReen, the entire screen is saved.

Example   This example selects the graticule for printing.
`10 OUTPUT 707;":LTESt:SSCReen:AREA GRATICULE"`

Query   `:LTESt:SSCReen:AREA?`

The query returns the current setting for the area of the screen to be saved.

Returned Format   `[:LTESt:SSCReen:AREA] {GRATicule | SCReen}<NL>`

Example   This example places the current selection for the area to be saved in the string variable, Selection$.
```
10 DIM Selection$[50]!Dimension variable
20 OUTPUT 707;":LTEST:SSCREEN:AREA?"
30 ENTER 707;Selection$
```

SSCReen:IMAGe

Command   `:LTESt:SSCReen:IMAGe {NORMal | INVert | MONochrome}`

This command saves the image normally, inverted, or in monochrome. IMAGe INVert is the same as choosing Invert Waveform Background in the Specify Report Action for measurement limit test dialog box.

Example   This example sets the image output to normal.
```
10 OUTPUT 707;":LTESt:SSCReen:IMAGE NORMAL"
```

Query   `:LTESt:SSCReen:IMAGe?`

The query returns the current image setting.

Returned Format   `[:LTESt:SSCReen:IMAGe] {NORMal | INVert | MONochrome}<NL>`

Example   This example places the current setting for the image in the string variable, Setting$.
```
10 DIM Setting$[50]!Dimension variable
20 OUTPUT 707;":LTEST:SSCREEN:IMAGE?"
30 ENTER 707;Setting$
```

SSUMmary

Command   `:LTESt:SSUMmary {OFF | DISK [,<filename>]}`

This command saves the summary in the event of a failure.

When set to disk, the summary is written to the disk drive. The summary is a logging method where the user can get an overall view of the test results. The summary is an ASCII file that the user can read on the computer or place into a spreadsheet.

<filename>   An ASCII string enclosed in quotation marks. If no filename is specified, the default filename will be *MeasLimitSummaryX.sum*, where X is an incremental number assigned by the instrument. If a filename is specified without a path, the default path will be `D:\User files\limit summaries`. (C drive on 86100A/B instruments.)

NOTE   If the summary of consecutive limit tests is to be stored in separate files, omit the <filename> parameter. Limit test summaries will be stored in files named *MeasLimitSummaryX.sum*, where X is an incremental number assigned by the instrument.

Example   The following example saves the summary to a disk file named *TEST.sum*.
```
10 OUTPUT 707;":LTEST:SSUMMARY DISK,TEST"
```

Query   `:LTESt:SSUMmary?`

The query returns the current specified destination for the summary.

Returned Format   `[:LTESt:SSUMmary] {OFF | DISK {,<filename>}}<NL>`

Example   The following example returns the current destination for the summary.
```
10 DIM SUMM$[50]
20 OUTPUT 707;":LTEST:SSUMMARY?"
30 ENTER 707;SUMM$
```

SWAVeform

Command   `:LTESt:SWAVeform <source>, <destination>[,<filename>[, <format>]]`

This command saves waveforms from a channel, function, TDR response or waveform memory in the event of measurement limit test termination, as specified by the :LTEST:RUNTil command. Each waveform source can be individually specified, allowing multiple channels, responses or functions to be saved to disk or waveform memories. Setting a particular source to OFF removes any waveform save action from that source.

NOTE This command operates on waveform and color grade gray scale data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

<source>	{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}
<destination>	{OFF | WMEMory<N> | DISK}
<filename>	An ASCII string enclosed in quotation marks. If no filename is specified, the assigned filename will be *MeasLimitChN_X*, *MeasLimitFnN_X*, *MeasLimitRspN_X*, or *MeasLimitMemN_X*, where X is an incremental number assigned by the instrument. If no path is specified, the default path will be `D:\ User Files\waveforms`. (C drive on 86100A/B instruments.)

NOTE If the selected waveforms of consecutive limit tests are to be stored in individual files, omit the <filename> parameter. The waveforms will be stored in the default format (INTERNAL) using the default naming scheme.

<format>	{TEXT [,YVALues | VERBose] | INTernal} where INTernal is the default value, and VERBose is the default value for TEXT.
Example	The following example turns off the saving of waveforms from channel 1 in the event of a limit test failure. `10 OUTPUT 707;":LTEST:SWAVEFORM CHAN1,OFF"`
Query	:LTESt:SWAVeform? <source> The query returns the current state of the :LTESt:SWAVeform command.
Returned Format	[:LTESt:SWAVeform]<source>, <destination>, [<filename>[,<format>]]<NL>

Example	The following example returns the current parameters for saving waveforms in the event of a limit test failure.
	`10 DIM SWAV$[50]`
	`20 OUTPUT 707;":LTEST:SWAVEFORM? CHANNEL1"`
	`30 ENTER 707;SWAV$`

SWAVeform:RESet

Command	:LTESt:SWAVeform:RESet This command sets the save destination for all waveforms to OFF. Setting a source to OFF removes any waveform save action from that source. This is a convenient way to turn off all saved waveforms if it is unknown which are being saved.
Example	`10 OUTPUT 707;":LTEST:SWAVeform:RESet"`

TEST

Command	:LTESt:TEST {ON | 1 | OFF | 0}

This command controls the execution of the limit test function. ON allows the limit test to run over all of the active measurements. When the limit test is turned on, the limit test results are displayed on screen in a window below the graticule. The results of the MEAS:RESults? query have three extra fields when LimitTESt:TEST is ON (failures, total, status). Failures is a number, total is a number, and status is one of the following values:

```
0   OK
1   failed high
2   failed low
3   failed inside
4   other failures
```

Query   `:LTESt:TEST?`

The query returns the state of the TEST control.

Returned Format   `[:LTESt:TEST] {1 | 0} <NL>`

Example   `10 OUTPUT 707;":LTEST:TEST OFF"`

ULIMit

Command   `:LTESt:ULIMit <upper_value>`

This command sets the upper test limit for the active measurement currently selected by the last :LTESt:SOURce command. <upper_value> is a real number.

Example   The following example sets the upper limit of the currently selected measurement to 500 mV.
`10 OUTPUT 707;":LTEST:ULIMIT 500E-3"`

Suppose you are measuring the maximum voltage of a signal with Vmax, and that voltage should not exceed 500 mV. You can use the above program and set the LTESt:FAIL OUTSide command to specify that the limit subsystem will fail a measurement when the voltage exceeds 500 mV.

Query   `:LTESt:ULIMit?`

The query returns the current upper limit of the limit test.

Returned Format   `[:LTESt:ULIMit] <upper_value><NL>`

Example   The following example returns the current upper limit of the limit test.
```
10 DIM ULIM$[50]
20 OUTPUT 707;":LTEST:ULIMIT?"
30 ENTER 707;ULIM$
```

16 Marker Commands

The commands in the MARKer subsystem are used to specify and query the settings of the time markers (X axis) and current measurement unit markers (volts, amps, and watts for the Y axis). The Y-axis measurement units are typically set using the CHANnel:UNITs command.

PROPagation

Command   `:MARKer:PROPagation {DIELectric | METer},<propagation>`

Sets the propagation velocity for TDR and TDT measurements. The propagation may be specified as a dielectric constant or in meters per second. The value is used to determine the distance from the reference plane in TDR and TDT marker measurements. To ensure accurate marker measurements, you must ensure that the propagation value is accurate, and that the units are set correctly (:MARKer:XUNITs). Propagation delay is always measured with respect to the reference plane. <propagation> is the dielectric constant or propagation value. You must specify one of the modifiers DIELectric or METer.

Query   `:MARKer:PROPagation?`

The query returns the current propagation value.

Returned Format   `[:MARKer:PROPagation]<propagation> {DIELectric | METer}<NL>`

Examples   The following example sets the propagation to 30 million meters per second.
`10 OUTPUT 707;":MARKER:PROPAGATION METER, 3E7"`

The following example gets the propagation value from the instrument, puts it into the variable, Prop$.

```
10 DIM Prop$[20]!Declare variable
20 OUTPUT 707;":MARKER:PROPAGATION?"
30 ENTER 707;Prop$
```

REACtance?

Query   `:MARKer:REACtance?`

In TDR mode, returns the excess reactance value when both markers are turned on. It returns the value as follows: <reactance_value>,<units> where reactance value is in scientific notation and units are F (farads) or H (henrys). When there is no reactance value, zero is returned and default units of F.

Returned Format   `[:MARKer:REACtance] <reactance_value>,<units><NL>`

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":MARKER:REACTANCE?"
```

REFerence

Command   `:MARKer:REFerence {TRIGger | REFPlane}`

Specifies the marker reference for TDR and TDT style markers. If the references is TRIGger, then all horizontal axis marker measurements are made with respect to the trigger point. If the reference is REFPlane, then all horizontal axis marker measurements are made with respect to the reference plane. This feature is available only TDR/TDT mode.

Query   `:MARKer:REFerence?`

The query returns the status of the marker reference.

Returned Format   `[:MARKer:REFerence] {TRIGger | REFPlane} <NL>`

Example   `10 OUTPUT 707;":MARKER:REFERENCE TRIGGER "`

RPANnotation

Command   `:MARKer:RPANnotation { {OFF | 0} | {ON | 1}}`

Sets the reference plane annotation on or off. The annotation is depicted as an inverted orange triangle positioned along the top of the graticule.

Query   `:MARKer:RPANnotation?`

Returned Format   `[:MARKer:RPANnotation] {1 | 0} <NL>`

Example   `10 OUTPUT 707;":MARKER:RPANNOTATION OFF"`

STATe

Command   `:MARKer:STATe {X1Y1 | X2Y2},<X_marker_state>,<Y_marker_state>`

Sets the state of a marker pair, X1Y1 or X2Y2, and specifies which marker pair state is set. Turn the X marker on or off with <X_marker_state> set to OFF or MANual. Set the <Y_marker_state> to OFF, MANual, or TRACk which turns the Y marker off, or sets to manual placement, or sets to tracking the source waveform at the X position. TRACk is allowed only when the <X_marker_state> is set to MANual. TRACk is not allowed in Eye/Mask mode.

Query   `:MARKer:STATe? {X1Y1 | X2Y2}`

Returns the states of the specified marker pair.

Returned Format	`[:MARKer:STATe] {X1Y1	X2Y2},<X_marker_state>,<Y_marker_state>`

Examples   This example sets the X1 marker to manual and the Y1 marker to track the source waveform at the current X1 position.

```
10 OUTPUT 707;":MARKer:STATe X1Y1, MANual, TRACk"
```

This example returns the current state of the X2 and Y2 markers to the string variable Marker_state$.

```
10 DIM Marker_state$[50]
20 Output 707;":MARKer:STATe? X2Y2"
30 ENTER 707;Marker_state$
```

X1Position

Command   `:MARKer:X1Position <X1_position>`

Sets the X1 marker position, and moves the X1 marker to the specified time with respect to the trigger time, if the X1 marker is on. <X1_position> is the time at X1 marker in seconds.

Query   `:MARKer:X1Position?`

The query returns the time at the X1 marker position.

Returned Format   `[:MARKer:X1Position] <X1_position><NL>`

Examples   This example sets the X1 marker to 90 ns.

```
10 OUTPUT 707;":MARKER:X1POSITION 90E-9"
```

This example returns the current setting of the X1 marker to the numeric variable, Value.

```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":MARKER:X1POSITION?"
30 ENTER 707;Value
```

X1Y1source

Command   `:MARKer:X1Y1source {CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>}`

Sets the source for the X1 and Y1 markers. <N> specifies channels, functions, TDR responses and waveform memories: 1, 2, 3, or 4. The source you specify must be enabled for markers to be displayed. If the channel, function, TDR response or waveform memory that you specify is not on, an error message is issued and the query will return NONE.

Query   `:MARKer:X1Y1source?`

The query returns the current source for markers X1 and Y1.

Returned Format   `[:MARKer:X1Y1source] {CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>}<NL>`

Examples   This example selects channel 1 as the source for markers X1 and Y1.

```
10 OUTPUT 707;":MARKER:X1Y1SOURCE CHANNEL1"
```

This example returns the current source selection for the X1 and Y1 markers to the string variable, Selection$.

```
10 DIM Selection$[50]!Dimension variable
20 OUTPUT 707;":MARKER:X1Y1SOURCE?"
30 ENTER 707;Selection$
```

X2Position

Command   `:MARKer:X2Position <X2_position>`

Sets the X2 marker position and moves the X2 marker to the specified time with respect to the trigger time, if the X2 marker is on. <X2_position> is the time at X2 marker in seconds.

Query   `:MARKer:X2Position?`

The query returns the time at the X2 marker in seconds.

Returned Format   `[:MARKer:X2Position] <X2_position><NL>`

Example   This example sets the X2 marker to 90 ns.
`10 OUTPUT 707;":MARKER:X2POSITION 90E-9"`

X2Y2source

Command   `:MARKer:X2Y2source {CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>}`

Sets the source for the X2 and Y2 markers. <N> specifies channels, functions, TDR responses and waveform memories: 1, 2, 3, or 4. The source you specify must be enabled for markers to be displayed. If the channel, function, TDR response or waveform memory that you specify is not on, an error message is issued and the query will return NONE.

Query   `:MARKer:X2Y2source?`

The query returns the current source for markers X2 and Y2.

Returned Format   `[:MARKer:X2Y2source] {CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>}<NL>`

Examples   This example selects channel 1 as the source for markers X2 and Y2.
`10 OUTPUT 707;":MARKER:X2Y2SOURCE CHANNEL1"`

This example returns the current source selection for the X2 and Y2 markers.
`20 OUTPUT 707;":MARKER:X2Y2SOURCE?"`

XDELta?

Query   `:MARKer:XDELta?`

This query returns the time difference in seconds between X1 and X2 time markers if they are both on. If both markers are not on, 9.999999E+37 will be returned.

Xdelta = time at X2 – time at X1

Returned Format   `[:MARKer:XDELta] <time><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MARKER:XDELTA?"`

XUNits

Command   `:MARKer:XUNits {SECond | METer}`

Sets the units for horizontal display in TDR and TDT applications. The units may be in seconds or meters relative to the reference plane. The marker mode must be TDR/TDT to use this feature.

Query   `:MARKer:XUNits?`

Returned Format   `[:MARKer:XUNits]{SECond | METer}<NL>`

Examples   `10 OUTPUT 707;":MARKER:XUNITS METER"`

Y1Position

Command   `:MARKer:Y1Position <Y1_position>`

Sets the Y1 manual marker position and moves the Y1 manual marker to the specified value on the specified source if the Y1 marker is in manual state. <Y1_position> is the current measurement unit value at Y1.

Query :MARKer:Y1Position?

The query returns the current measurement unit level at the Y1 marker position.

Returned Format [:MARKer:Y1Position] <Y1_position><NL>

Example This example sets the Y1 marker to 10 mV.
10 OUTPUT 707;":MARKER:Y1POSITION 10E-3"

Y2Position

Command :MARKer:Y2Position <Y2_position>

Sets the Y2 manual marker position and moves the Y2 manual marker to the specified value on the specified source if the Y2 marker is in manual state. <Y2_position> is the current measurement unit value at Y2.

Query :MARKer:Y2Position?

The query returns the current measurement unit level at the Y2 marker position.

Returned Format [:MARKer:Y2Position] <Y2_position><NL>

Examples This example sets the Y2 marker to –100 mV.
10 OUTPUT 707;":MARKER:Y2POSITION -100E-3"

YDELta?

Query :MARKer:YDELta?

This query returns the current measurement unit difference between Y1 and Y2 if they are both on and both have the same source. If not, 9.999999E+37 is returned.

Vdelta = value at Y2 – value at Y1

Returned Format [:MARKer:YDELta] <value><NL>

<value> Measurement unit difference between Y1 and Y2.

Example 10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MARKER:YDELTA?"

YUNits

Command :MARKer:YUNits {VOLT | OHM | REFLect}

Sets the units for vertical display in TDR and TDT applications. The units may be in volts, ohms, or % reflection. The marker mode must be TDR/TDT to use this feature.

Query :MARKer:YUNits?

This query returns the current marker units setting.

Returned Format [:MARKer:YUNits]{VOLT | OHM | REFLect}<NL>

Example 10 OUTPUT 707;":MARKER:YUNITS OHM"

17 Mask Test Commands

KEYSIGHT
TECHNOLOGIES

The Mask Test commands and queries control the mask test features. Mask testing automatically compares measurement results with the boundaries of the mask you select. Any waveform or sample that falls within the boundaries of the mask is recorded as a failure. The following lines provide a simple series of commands that run a mask test:

```
10 OUTPUT 707;":STOP"
20 OUTPUT 707;":MTESt:SOURce CHANnel1"
30 OUTPUT 707;":MTESt:ARUN OFF"
40 OUTPUT 707;":MTESt:LOAD ""FILE1.MSK"
50 OUTPUT 707;":MTESt:RUNTil FSAMples,50"
50 OUTPUT 707;":MTEST:STARt"
60 OUTPUT 707;":RUN"
```

NOTE   Compatibility with the Agilent 83480A/54750A. In commands with a REGion parameter, POLYgon may be used in place of REGion for compatibility with the Agilent 83480A/54750A.

For histograms, mask testing, and color grade-gray scale display, the instrument uses a specific database. This database uses a different memory area than the waveform record for each channel. When any of the histograms, mask testing, and color grade-gray scale display features is turned on, the instrument starts building the database. The database is the size of the graticule area, which is 321 pixels high by 451 pixels wide. Behind each pixel is a 16-bit counter. Each counter is incremented each time a pixel is hit by data from a channel or function. The maximum count (saturation) for each counter is 63,488. You can check to see if any of the counters is close to saturation by using the :MEASure:CGRade:PEAK? query. The color-graded display uses colors to represent the number of hits on various areas of the display.

The database continues to build until the instrument stops acquiring data or all three functions (color grade-gray scale display, mask testing, and histograms) are turned off. The instrument stops acquiring data when the power is cycled, the Stop/Single hardkey is pressed, after a specified number of waveforms or samples are acquired, or as another module is plugged in.

You can clear the database by pressing the Clear Display hardkey, cycling the power, turning off all three features that use the database, or sending a CDISplay command.

Before firmware revision 3.00, the database does not differentiate waveforms from different channels or functions. If three channels are turned on and the waveform for each channel happens to light the same pixel at the same time, the counter is incremented by three. However, you cannot tell how many hits came from each waveform. For this reason, mask test is available in Eye/Mask mode only, which allows only one channel to function at a time. For firmware revisions 3.00 and above multiple data bases are supported.

To avoid erroneous data, clear the display after you change instrument setup conditions or device under test (DUT) conditions and acquire new data before extracting measurement results.

The instrument provides a series of standard masks defined according to telecom and datacom standards. For a complete list of masks and templates, refer to the online Help. Load a mask file using the DISK:LOAD or :MTESt:LOAD commands. Mask files have the *.msk* or *.pcm* extensions.

ALIGn

Command   :MTESt:ALIGn

Automatically aligns and scales the mask to the current waveform.

Example   10 OUTPUT 707;":MTEST:ALIGN"

AMARgin:BER

Command   :MTESt:AMARgin:BER <value>

Enters or queries the BER value that the MTESt:AMARgin:CALCulate command uses to automatically set a mask-test margin based on BER. The preferred command for new programs (due to improved performance with *OPC queries) is the MTESt:AMARgin:HRATio command.

Query   `:MTESt:AMARgin:BER?`

Returned Format   `[:MTESt:AMARgin:BER] <value><NL>`

Example   `10 OUTPUT 707;":MTEST:AMARgin:BER 1E-12"`

AMARgin:CALCulate

Command   `:MTESt:AMARgin:CALCulate`

Automatically determines a mask-margin percent and displays the mask margin that has the expected number of errors (hit ratio) for the bit-error-ratio (BER) level entered with the MTEST:AMARgin:HRATio or MTESt:AMARgin:BER commands.

Example   `10 OUTPUT 707;":MTEST:AMARgin:CALCulate"`

AMARgin:HRATio

Command   `:MTESt:AMARgin:HRATio <value>`

Enters or queries the Hit Ratio value that the MTESt:AMARgin:CALCulate command uses to automatically set a mask-test margin based on BER. This command is preferred to using the MTESt:AMARgin:BER command.

Query   `:MTESt:AMARgin:HRATio?`

Restrictions   86100D or 86100C (software revision A.10.60 and above).

Returned Format   `[:MTESt:AMARgin:HRATio] <value><NL>`

Example   `10 OUTPUT 707;":MTEST:AMARgin:HRATio 1E-12"`

AMEThod

Command   `:MTESt:AMEThod {NRZeye | RZeye | ECMean | NONE}`

Sets the mask alignment method. Use this command in the setup section of a mask file when defining a custom mask. It ensures that the mask will be properly aligned if more alignment methods become available in the future. NRZeye aligns the mask reference point to the first eye crossing on screen for non-return to zero (NRZ) measurements. RZeye aligns the mask reference point to the first center location of the eye-closing for return to zero (RZ) measurements. ECMean aligns the mask reference point to the eye crossing mean of the rise and fall time at waveform average power at the first eye crossing point for NRZ eye measurements. This is currently applicable to 10 GbEthernet masks. NONE specifies no alignment takes place.

Query   `:MTESt:AMEThod?`

Returned Format   `[:MTESt:AMEThod] NRZ<NL>`

Example   `10 OUTPUT 707;":MTEST:AMEThod NRZ"`

AOPTimize

Command   `:MTESt:AOPTimize {ON | 1 | OFF | 0}`

Enables or disables optimization of the placement of the center mask region during mask alignment. This command affects the operation of mask alignment which is performed by the :MTESt:STARt and :MTESt:ALIGn commands. When optimization is turned, on the center region (Region 1) is offset along the X-axis to achieve the best mask test margin when mask alignment is performed. The amount of offset is in the range of ±25% of the unit interval. Optimization is reset to off whenever a mask file is loaded. Optimization may be enabled for a specific mask file by embedding the command :MTESt:AOPTimize ON in the setup block at the end of the mask file.

NOTE   Not all mask test standards allow optimization. Optimization is enabled in mask files provided by Agilent Technologies as allowed by relevant standards. To ensure conformance, consult appropriate standards documents before enabling optimization.

Restrictions	Software revision A.03.05 and above.
Query	`:MTESt:AOPTimize?`
	The query returns the state of alignment optimization.
Returned format	`[:MTESt:AOPTize] {1	0}<NL>`
Example	`10 OUTPUT 707;":MTEST:AOPTIMIZE ON"`

ARUN

Command	`:MTESt:ARUN {ON	1	OFF	0}`
	Turns on or off the automatic starting of a mask test. If the instrument acquisition mode is set to Run ("RUN" on page 113), the mask test always start immediately after sending the :MTESt:STARt command. If the instrument acquisition mode is set to Stop ("STOP" on page 114) and :MTESt:ARUN is set to ON, mask testing starts when the :MTESt:STARt command is sent. However, if :MTESt:ARUN is set to OFF in stop mode, mask testing does not start until the acquisition mode is set to Run.
Restrictions	Software revision A.10.60 and above.
Query	`:MTESt:ARUN?`
Example	`10 OUTPUT 707;":MTEST:ARUN"`

COUNt:FAILures?

Query	`:MTESt:COUNt:FAILures? {{POLYgon	REGion}{1:8}} [,{TOTal	MARGin	MASK}]`
	Returns the number of failures that occurred within a particular region. By defining regions within regions, then counting the failures for each individual region, you can implement testing at different tolerance levels for a given waveform. The value 9.999E37 is returned if mask testing is not enabled or if you specify a region number that is not used. The integer, 1 through 8, designates the region for which you want to determine the failure count. The TOTal, MARGin, and MASK (default) arguments specifies which portion of the region to return failures (hits) for. TOTal returns the total number of failed data points. For positive margins, this is the sum of the MASK and MARGin counts. For negative margins, this is the same as the MASK count. MARGin returns the number of data points that occurred *between* the margin mask and the standard mask. This is the margin area. This definition is true for both positive and negative margins. If the query argument is ommited, the default argument used is MASK.
Restrictions	Software revision A.07.00 and above for POLYgon, TOTal, MARGin, and MASK arguments.
Returned Format	`[:MTESt:COUNt:FAILures] <number_of_failures><NL>`

<number_of_failures> is the number of failures that have occurred for the designated region.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MTEST:COUNT:FAILURES? REGION3"

COUNt:FSAMples?

Query   :MTESt:COUNt:FSAMples?

Returns the total number of failed samples in the current mask test run. This count is for all regions and all waveforms, so if you wish to determine failures by region number, use the COUNt:FAILures? query. The count value returned is not the sum of the failure counts for each region. For example, assume a region 2 enclosed completely by region 1. If region 1 has 100 failures, the value returned is 100, regardless of how many failures are in region 2. Because region 2 is completely enclosed, the failure count for region 2 must be less than or equal to 100 in this instance.

The value 9.999E37 is returned if mask testing is not enabled.

Returned Format   [:MTESt:COUNt:FSAMples] <number_of_failed_samples><NL>

<number_of_failed _samples> is the total number of failed samples for the current test run.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MTEST:COUNT:FSAMPLES?"

COUNt:HITS?

Query   :MTESt:COUNt:HITS? {TOTal | MARGin | MASK}

Returns the number of failed data points (or hits) that occurred when using margin mask testing. TOTal returns the total number of failed data points. For positive margins, this is the sum of the MASK and MARGin counts. For negative margins, this is the same as the MASK count. MARGin returns the number of data points that occurred *between* the margin mask and the standard mask. This is the margin area. This definition is true for both positive and negative margins. To determine a negative margin, increase the magnitude of the negative margin until the number of margin hits goes to zero. All data acquired since mask margin testing was enabled will be compared to the margin. Sampled points acquired before the margin was activated, that fall into the margin region, will also show up as mask hits. MASK returns the number of data points that failed the standard mask.

Returned Format   [:MTESt:COUNt:HITS] <number_of_hits><NL>

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MTEST:COUNT:HITS? MARGin"

COUNt:SAMPles?

Query   :MTESt:COUNt:SAMPles?

Returns the total number of samples captured in the current mask test run. The value 9.999E37 is returned if mask testing is not enabled.

Returned Format   [:MTESt:COUNt:SAMPles] <number_of_samples><NL>

<number_of _samples> is the total number of samples for the current test run.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MTEST:COUNT:SAMPLES?"

COUNt:WAVeforms?

Query   :MTESt:COUNt:WAVeforms?

Returns the total number of waveforms gathered in the current mask test run. The value 9.999E37 is returned if mask testing is not enabled.

Returned Format   `[:MTESt:COUNt:WAVeforms] <number_of_waveforms><NL>`

<number_of_ waveforms> is the total number of waveforms for the current test run.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MTEST:COUNT:WAVEFORMS?"`

DELete

Command   `:MTESt:DELete`

Clears the currently loaded mask. MTESt:DELete is the preferred command. (See also MTESt:MASK:DELete.)

Example   `10 OUTPUT 707;":MTEST:DELETE"`

EXIT

Command   `:MTESt:EXIT`

Terminates mask testing.

Example   `10 OUTPUT 707;":MTEST:EXIT"`

LOAD

Command   `:MTESt:LOAD "<file_name>"`

This command loads the specified mask file. This command operates only on files and directories on "A:\", "D:\User Files", "C:\scope\masks" and any mapped network drive. <file_name> is the filename, with the extension .msk or .pcm. You can specify the entire path, or use a relative path such as "." or ".." If you use a relative path, the present working directory is assumed. Use DISK:CDIRectory to change the present working directory, and DISK:PWD? to query it.

If no path is specified, a search path is followed. The directory C:\scope\masks is searched first, then D:\User Files\masks.

If no filename extension is specified, an attempt will be made to open a file having the specified filename with a '.msk' extension appended. If unsuccessful, an attempt will be made to open a file having the specified filename with a '.pcm' extension appended.

Example   `10 OUTPUT 707;":MTESt:LOAD ""FILE1.MSK"`

MASK:DELete

Command   `:MTESt:MASK:DELete`

This command deletes the complete currently defined mask and is provided for compatibility with the Agilent 83480A/54750A. For new programs, use the :MTESt:DELete form which performs the same function.

Example   `10 OUTPUT 707;":MTEST:MASK:DELETE"`

MMARgin:PERCent

Command   `:MTESt:MMARgin:PERCent <margin_percent>`

Sets the amount of mask margin to apply to the selected mask.

<margin_percent> is an integer, –100 to 100, expressing the mask margin in percent.

Query `:MTESt:MMARgin:PERCent?`

The query returns the current mask margin.

Returned Format `[:MTESt:MMARgin:PERCent] <margin_percent><NL>`

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MTEST:MMARGIN:PERCENT?"
```

MMARgin:STATe

Command `:MTESt:MMARgin:STATe {ON | 1 | OFF | 0}`

Controls the activation of the mask margin.

Query `:MTESt:MMARgin:STATe?`

The query returns the current mask margin state.

Returned Format `[:MTESt:MMARgin:STATe] {1 | 0}<NL>`

Example `10 OUTPUT 707;":MTEST:MMARgin:STATe ON"`

RUNTil

Command `:MTESt:RUNTil {OFF | FSAMples, <number_of_failed_samples>}`

Selects the acquisition run until mode and is provided for compatibility with the Agilent 83480A/54750A. For new programs, use the ACQuire:RUNTil command on **page 126** which performs the same function.

The acquisition may be set to run until *n* fsamples have been acquired or to run forever (OFF). If more than one limit test criteria is set, then the instrument will act upon the completion of whichever limit test criteria is achieved first. To run the acquisition for a specific number of waveforms or samples, use ACQuire:RUNTil. A mask test must be running (:MTESt:TEST ON or :MTESt:STARt) before setting acquisition to run until *n* fsamples.

<number_of_failed_samples> is an integer from 1 to 1,000,000,000.

Query `:MTESt:RUNTil?`

The query returns the currently selected run until state.

Returned Format `[:MTESt:RUNTil] {OFF | FSAMples, <n fsamples>}<NL>`

Example The following example specifies that the acquisition runs until 50 samples have been obtained.
```
10 OUTPUT 707;":MTESt:STARt"
20 OUTPUT 707;":MTESt:RUNTIL FSAMples,50"
```

SAVE

Command `:MTESt:SAVE "<file_name>"`

Saves user-defined (custom) masks in either the .msk or the .pcm format. The filename argument, <file_name>, uses the file extension *.msk* or *.pcm.* If no file suffix is specified, .pcm is appended. You can specify the entire path, or use a relative path such as "." or ".." Valid destinations are any mapped network drive, the floppy drive (A:) and D:\User Files and its subdirectories. If no path is specified, the file is saved in the directory D:\User Files\masks. (C drive on 86100A/B instruments.) If you use a relative path, the present working directory is assumed. Use DISK:CDIRectory to change the present working directory, and DISK:PWD? to query it.

SCALe:DEFault

Command   :MTESt:SCALe:DEFault

Sets the scaling markers to default values. The X1, Y1, and Y2 markers are set to values corresponding to graticule positions that are two divisions in from the left, top, and bottom of the graticule, respectively. Y1 and Y2 are not set for fixed voltage masks. These values are defined in the setup section of the mask file.

Example   The following example selects the default scale.

10 OUTPUT 707;":MTEST:SCALE:DEFAULT"

SCALe:MODE

Command   :MTESt:SCALe:MODE {XANDY| XONLy}

Sets the mask scaling mode and should be used in the setup section of a mask file when defining a custom mask. It ensures the mask will be properly loaded and adjusted on the screen. Scale mode needs to be specified for fixed voltage masks. All other masks are loaded as XANDY mode. XANDY specifies that when a mask is loaded and aligned, the time value reference point (X) and vertical scaling points (Y) are adjusted. This parameter applies to all non-fixed voltage masks. XONLy specifies that when a mask is loaded and aligned, only the time value reference point (X) is adjusted. The vertical scaling points (Y) remain fixed. This parameter applies to fixed voltage masks.

Query   :MTESt:SCALe:MODE?

The query returns the scaling mode.

Returned Format   [:MTESt:SCALe:MODE] {XANDY | XONL}<NL>

Examples   10 OUTPUT 707;" :MTEST:SCALe:MODE XONLy"

SCALe:SOURce?

Query   :MTESt:SCALe:SOURce?

The query returns the name of the source currently used to interpret the Y1 and Y2 scale factors.

Returned Format   [:MTESt:SCALe:SOURce] FUNCtion<N> | CHANnel<N> | CGMemory} <NL>

Example   20 OUTPUT 707;":MTEST:SCALE:SOURCE?"

SCALe:X1

Command   :MTESt:SCALe:X1 <x1_value>

Defines where X=0 in the base coordinate system used for mask testing. The other X coordinate is defined by the SCALe:XDELta command. Once the X1 and XDELta coordinates are set, all X values of vertices in region masks are defined with respect to this value, according to the equation:

$$X = (X \times \text{XDELta}) + \text{X1}$$

Thus, if you set X1 to 100 μs, and XDELta to 100 μs, an X value of .100 in a vertex is at 110 μs. The instrument uses this equation to normalize vertex values. This simplifies reprogramming to handle different data rates. For example, if you halve the period of the waveform of interest, you need only to adjust the XDELta value to set up the mask for the new waveform. The <x1_value> argument is a time value specifying the location of the X1 coordinate, which will then be treated as X=0 for region vertex coordinates.

Query	`:MTESt:SCALe:X1?`
Returned Format	`[:MTESt:SCALe:X1] <x1_value> <NL>`
Examples	`10 OUTPUT 707;":MTEST:SCALE:X1 150E-6"`

SCALe:XDELta

Command   `:MTESt:SCALe:XDELta <xdelta_value>`

Defines the position of the X2 marker with respect to the X1 marker. In the mask test coordinate system, the X1 marker defines where X=0; thus, the X2 marker defines where X=1. Because all X vertices of regions defined for mask testing are normalized with respect to X1 and ΔX, redefining ΔX also moves those vertices to stay in the same locations with respect to X1 and ΔX. Thus, in many applications, it is best if you define XDELta as a pulse width or bit period. Then a change in data rate, without corresponding changes in the waveform, can easily be handled by changing ΔX. The X-coordinate of region vertices are normalized using the equation:

$$X = (X \times \text{XDELta}) + \text{X1}$$

`<xdelta_value>` is a time value specifying the distance of the X2 marker with respect to the X1 marker.

Query   `:MTESt:SCALe:XDELta?`

The query returns the current value of ΔX.

Returned Format   `[:MTESt:SCALe:XDELta] <xdelta_value> <NL>`

Examples   Assume that the period of the waveform you wish to test is 1 μs. Then the following example will set ΔX to 1 μs, ensuring that the waveform's period is between the X1 and X2 markers.
`10 OUTPUT 707;":MTEST:SCALE:XDELTA 1E-6"`

SCALe:Y1

Command   `:MTESt:SCALe:Y1 <y1_value>`

Defines where Y=0 in the coordinate system for mask testing. All Y values of vertices in the coordinate system are defined with respect to the boundaries set by SCALe:Y1 and SCALe:Y2, according to the equation:

$$Y = (Y \times (Y2 - Y1)) + Y1$$

Thus, if you set Y1 to 100 mV, and Y2 to 1V, a Y value of .100 in a vertex is at 190 mV. The `<y1_value>` argument is a voltage value specifying the point at which Y=0.

Query   `:MTESt:SCALe:Y1?`

The query returns the current setting of the Y1 marker.

Returned Format   `[:MTESt:SCALe:Y1] <y1_value><NL>`

Example   `10 OUTPUT 707;":MTEST:SCALE:Y1 -150E-3"`

SCALe:Y2

Command	`:MTESt:SCALe:Y2 <y2_value>`

Defines Y=1 in the coordinate system for mask testing. All Y values of vertices in the coordinate system are defined with respect to the boundaries defined by SCALe:Y1 and SCALe:Y2, according to the following equation:

$$Y = (Y \times (Y2 - Y1)) + Y1$$

Thus, if you set Y1 to 100 mV, and Y2 to 1V, a Y value of .100 in a vertex is at 190 mV. The <y2_value> argument is a voltage value specifying the location of the Y2 marker.

Query	`:MTESt:SCALe:Y2?`

Returns the current setting of the Y2 marker.

Returned Format	`[:MTESt:SCALe:Y2] <y2_value> <NL>`
Example	`10 OUTPUT 707;":MTEST:SCALE:Y2 2.5"`

SOURce

Command	`:MTESt:SOURce {CHANnel<N>	FUNCtion<N>	CGMemory}`

Sets the database source for mask tests. The default is the lowest numbered database signal displayed. <N> is an integer, 1 through 4.

Query	`:MTESt:SOURce?`

This query returns the current database source for the mask test.

Returned Format	`[:MTESt:SOURce] {CHANnel<N>	FUNCtion<N>	CGMemory}<NL>`
Example	`10 OUTPUT 707;":MTEST:SOURCE CHANNEL1"`

SCALe:YTRack

Command	`:MTESt:SCALe:YTRack {{ON	1} {OFF	0}}`

Enables or disables tracking between the Y1 and Y2 levels.

Query	`:MTESt:SCALe:YTRack?`

The query returns the current state of the tracking.

Returned Format	`[:MTESt:SCALe:YTRack] {1	0}<NL>`
Example	`10 OUTPUT 707;":MTEST:SCALE:YTRACK:ON"`

SSCReen

Command	`:MTESt:SSCReen {OFF	DISK [,<filename>]}`

Saves a copy of the screen in the event of a failure. OFF turns off the save action. DISK saves a copy of the screen to disk in the event of a failure. Additional disk-related controls are set using the SSCReen:AREA and SSCReen:IMAGe commands. The <filename> argument is an ASCII string enclosed in quotations marks. If no filename is specified, a filename will be assigned. The default filename is *MaskLimitScreenX.bmp*, where X is an incremental number assigned by the instrument.

The save screen options established by the commands MTESt:SSCReen DISK, MTESt:SSCReen:AREA, and MTESt:SSCReen:IMAG are stored in the instrument's memory and will be employed in consecutive save screen operations, until changed by the user. This includes the ‹filename› parameter for the MTESt:SSCReen DISK command. If the results of consecutive limit tests must be stored in different files, omit the ‹filename› parameter and use the default filename instead. Each screen image will be saved in a different file named *MaskLimitScreenX.bmp*, where X is an incremental number assigned by the instrument.

The *filename* field includes the network path and the directory in which the file will be saved, as well as the file format that will be used. The following is a list of valid file locations:

• Files can only be created within the folder "D:\User Files" (C: on 86100A/B) or on any external drive or mapped network drive.

• Files can not be saved on the root folder of the D: drive (C: on 86100A/B).

• Files can not be saved on USB removable drives. To save files on a USB drive, use front-panel controls. *(Applies only to firmware version A.09.00 and below)*

• Using the command "CDIRectory" on page 165 to change the present working directory has no effect on the location of saved files.

Table 40   Example Filenames

File Name	File Saved in Directory...
"test.jpg"	D:\User Files\screen images\ This is the default folder. Filenames without a path are saved to this folder.
"subfolder\test.jpg"	D:\User Files\screen images\subfolder The subfolder must already exist before saving the file.
"D:\User Files\subfolder\test.jpg"	D:\User Files\subfolder The subfolder must already exist before saving the file.
"D:\User Files\test.jpg"	D:\User Files
"D:\test.jpg"	This is not a valid file location. The file is not saved.
"E:test4.eps"	File saved in the instrument's drive E:, that could be mapped to any disk in the network.
"\\computer-ID\d$\test3.bmp"	File saved in drive D: of computer "computer-ID", provided all permissions are set properly.

The following graphics formats are available by specifying a file extension: PCX files (.pcx), EPS files (.eps), Postscript files (.ps), JPEG (.jpg), TIFF (.tif), and GIF files (.gif).

Query   `:MTESt:SSCReen?`

The query returns the current state of the SSCReen command.

Returned Format   `[:MTESt:SSCReen] {OFF | DISK [,<filename>]}<NL>`

Example   `10 OUTPUT 707;":MTESt:SSCREEN DISK"`

SSCReen:AREA

Command   `:MTESt:SSCReen:AREA {GRATicule | SCReen}`

Selects which data from the screen is to be saved to disk when the run until condition is met. When you select GRATicule, only the graticule area of the screen is saved (this is the same as choosing Waveforms Only in the Specify Report Action for mask limit test dialog box). When you select SCReen, the entire screen is saved.

Query   `:MTESt:SSCReen:AREA?`

The query returns the current setting for the area of the screen to be saved.

Returned Format   `[:MTESt:SSCReen:AREA] {GRATicule | SCReen}<NL>`

Example   `10 OUTPUT 707;":MTEST:SSCREEN:AREA GRATICULE"`

SSCReen:IMAGe

Command   `:MTESt:SSCReen:IMAGe {NORMal | INVert | MONochrome}`

Saves the screen image to disk normally, inverted, or in monochrome. IMAGe INVert is the same as choosing Invert Waveform Background Color in the Specify Report Action for acquisition limit test dialog box.

Query   `:MTESt:SSCReen:IMAGe?`

The query returns the current image setting.

Returned Format	`[:MTESt:SSCReen:IMAGe] {NORMal | INVert | MONochrome}<NL>`
Example	`10 OUTPUT 707;":MTEST:SSCREEN:IMAGE NORMAL"`

SSUMmary

Command	`:MTESt:SSUMmary {OFF | DISK [,<filename>]}`

Saves the summary in the event of a failure. When set to disk, the summary is written to the disk drive. The summary is a logging method where the user can get an overall view of the test results. The summary is an ASCII file that the user can read on the computer or place into a spreadsheet. The <filename> argument is an ASCII string enclosed in quotation marks. If no filename is specified, the default filename will be MaskLimitSummaryX.sum, where X is an incremental number assigned by the instrument. If a filename is specified without a path, the default path will be D:\User Files\limit summaries. (C drive on 86100A/B instruments.)

NOTE If the summary of consecutive limit tests is to be stored in individual files, omit the <filename> parameter. Limit test summaries will be stored in files named *MaskLimitSummaryX.sum*, where X is an incremental number assigned by the instrument.

Query	`:MTESt:SSUMmary?`

The query returns the current specified destination for the summary.

Returned Format	`[:MTESt:SSUMmary] {OFF | DISK {,<filename>}}<NL>`
Examples	The following example saves the summary to a disk file named TEST.sum. `10 OUTPUT 707;":MTEST:SSUMMARY DISK,TEST"`

STARt

Command	`:MTESt:STARt`

Aligns the currently loaded mask to the current waveform and starts testing. If no mask is currently loaded, a warning message will be displayed, but no error will be generated.

Example	`10 OUTPUT 707;":STOP"`
	`20 OUTPUT 707;":MTESt:SOURce CHANnel1"`
	`30 OUTPUT 707;":MTESt:ARUN OFF"`
	`40 OUTPUT 707;":MTESt:LOAD ""FILE1.MSK"`
	`50 OUTPUT 707;":MTESt:RUNTil FSAMples,50"`
	`50 OUTPUT 707;":MTEST:STARt"`
	`60 OUTPUT 707;":RUN"`

SWAVeform

Command	`:MTESt:SWAVeform <source>, <destination>[,<filename>[, <format>]]`

Saves waveforms from a channel, function, or waveform memory in the event of a failure detected by the limit test. Each waveform source can be individually specified, allowing multiple channels,or functions to be saved to disk or waveform memories. Setting a particular source to OFF removes any waveform save action from that source. The source <source> argument can be CHANnel<N>, FUNCtion<N>, or WMEMory<N>. The < destination> argument can be OFF, WMEMory<N>, or DISK. The <filename> argument is an ASCII string enclosed in quotation marks. If no filename is specified, the assigned filename will be *MaskLimitChN_X*, *MaskLimitFnN_X*, *MaskLimitRspN_X*, or *MaskLimitMemN_X*, where X is an incremental number assigned by the instrument. If no path is

specified, the default path will be D:\User Files\waveforms. (C drive on 86100A/B instruments.) The <format> argument can be {TEXT [,YVALues | VERBose] | INTernal}. INTernal is the default value, and VERBose is the default value for TEXT.

NOTE	This command operates on waveform and color grade gray scale data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

NOTE	If the selected waveforms of consecutive limit tests are to be stored in individual files, omit the <filename> parameter. The waveforms will be stored in the default format (INTERNAL) using the default naming scheme.

Example The following example turns off the saving of waveforms from channel 1 in the event of a limit test failure.
```
10 OUTPUT 707;":MTEST:SWAVEFORM CHAN1,OFF"
```

Query `:MTESt:SWAVeform? <source>`

The query returns the current state of the :MTESt:SWAVeform command.

Returned Format `[:MTESt:SWAVeform] <source>, <destination>, [<filename>[,<format>]]<NL>`

Example The following example returns the current parameters for saving waveforms in the event of a limit test failure.
```
10 DIM SWAV$[50]
20 OUTPUT 707;":MTEST:SWAVEFORM? CHANNEL1"
30 ENTER 707;SWAV$
```

SWAVeform:RESet

Command `:MTESt:SWAVeform:RESet`

Sets the save destination for all waveforms to OFF. Setting a source to OFF removes any waveform save action from that source. This is a convenient way to turn off all saved waveforms if it is unknown which are being saved.

Example `10 OUTPUT 707;":MTEST:SWAVeform:RESet"`

TEST

Command `:MTESt:TEST {ON | 1 | OFF | 0}`

Controls the execution of the Mask Test function and is provided for compatibility with the Agilent 83480A/54750A. For new programs, use :MTESt:STARt on and :MTEST:EXIT on .

Restrictions Mask limit testing only.

Query `:MTESt:TEST?`

Returned Format `[:MTESt:TEST] {1 | 0}<NL>`

Example `10 OUTPUT 707;":MTEST:TEST?"`

TITLe?

Query `:MTESt:TITLe?`

Returns the string of the currently loaded mask. If no mask is loaded, a null string is returned.

Returned Format   `[:MTESt:TITLe] <"title">`

YALign

Command   `:MTESt:YALign {DISPlay | EWINdow}`

Sets the vertical axis alignment mode of the mask. It ensures the mask will be properly adjusted on the screen. Alignment mode needs to be specified for optical NRZ masks. DISPlay specifies that instrument aligns the mask using Vtop and Vbase of the eye diagram. This parameter applies to fixed voltage masks. EWINdow specifies that instrument aligns the mask using the one level and zero level of the eye diagram. This parameter applies to optical NRZ masks.

Query   `:MTESt:YALign?`

The query returns the alignment mode.

Returned Format   `[:MTES:YAL] {DISP | EWIN}<NL>`

Example   `10 OUTPUT 707;" :MTEST:YALign EWINdow"`

18 Measure Commands

KEYSIGHT
TECHNOLOGIES

The commands in the MEASure subsystem are used to make parametric measurements on displayed waveforms. The instrument has four modes: Eye/Mask, Jitter, TDR/TDT, and Oscilloscope. Each mode has a set of measurements. In Eye/Mask mode, all of the measurements are made on the color grade/gray scale data, with the exception of average optical power and histogram measurements.

Introduction

Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must be displayed on the analyzer.

· For a period or frequency measurement, at least one and one half complete cycles must be displayed.

· For a pulse width measurement, the entire pulse must be displayed.

· For a rise time measurement, the leading (positive-going) edge of the waveform must be displayed.

· For a fall time measurement, the trailing (negative-going) edge of the waveform must be displayed.

· A valid source for the measurement must be designated. This can be done globally with the MEASure:SOURce command or locally with the optional source parameter in each measurement.

User-Defined Measurements

When user-defined measurements are made, the defined parameters must be set before actually sending the measurement command or query.

Measurement Incomplete or Error

If a measurement cannot be made because of the lack of data, because the source signal is not displayed, the requested measurement is not possible (for example, a period measurement on an FFT waveform), or for some other reason, 9.99999E+37 is returned as the measurement result. In TDR mode with ohms specified, the returned value is 838 MΩ.

Use SENDvalid to determine if the measurement results are complete or valid. SENDvalid casuses an additional error code to be returned with the results. If SENDvalid is off, any questionable restults are returned as 9.999E+37. If SENDvalid is on, current values are returned for any questionable results *and* an additional result telling you if the data is valid or not. A zero "0" indicates valid data, a "1" indicates the data is questionable.

Making Measurements

If more than one period, edge, or pulse is displayed, time measurements are made on the first, left-most portion of the displayed waveform. When any of the defined measurements are requested, the analyzer first determines the top (100%) and base (0%) voltages of the waveform. From this information, the analyzer determines the other important voltage values (10%, 90%, and 50% voltage values) for making measurements. The 10% and 90% voltage values are used in the rise-time and fall-time measurements when standard measurements are selected. The 50% voltage value is used for measuring frequency, period, pulse width, and duty cycle with standard measurements selected.

You can also make measurements using user-defined parameters, instead of the standard measurement values. When the command form of a measurement is used, the analyzer is placed in the continuous measurement mode. The measurement result will be displayed on the front panel. There may be a maximum of four measurements running continuously. Use the SCRatch command to turn off the measurements. When the query form of the measurement is used, the measurement is made one time, and the measurement result is returned.

· If the current acquisition is complete, the current acquisition is measured and the result is returned.

- If the current acquisition is incomplete and the analyzer is running, acquisitions will continue to occur until the acquisition is complete. The acquisition will then be measured and the result returned.

- If the current acquisition is incomplete and the analyzer is stopped, the measurement result will be 9.99999E+37 and the incomplete result state will be returned if SENDvalid is ON.

All measurements are made using the entire display, except for VRMS which allows measurements on a single cycle, and eye measurements in the defined eye window. Therefore, if you want to make measurements on a particular cycle, display only that cycle on the screen. Measurements are made on the displayed waveforms specified by the SOURce command. The SOURce command allows two sources to be specified. Most measurements are only made on a single source. Some measurements, such as the DELTatime measurement, require two sources. The measurement source for remote measurements can not be set from the front panel. The measurement source is not reset by power cycles or default setup. If the signal is clipped, the measurement result may be questionable. In this case, the value returned is the most accurate value that can be made using the current scaling. You might be able to obtain a more accurate measurement by adjusting the vertical scale to prevent the signal from being clipped. The measurement result 9.99999E+37 may be returned in some cases of clipped signals.

Commands

AMPLitude:ANALysis

Command   `:MEASure:AMPLitude:ANALysis {ON | 1 | OFF | 0}`

Turns on analysis of noise and interference. This includes the following measurements: Total Interference (TI), Random Noise (RN), Deterministic Interference (DI), Inter-Symbol Interference (ISI), Periodic Interference (PI), BER floor, BER limit, Signal Amplitude, Eye Opening, Q, and Relative Intensity Noise (RIN).

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example   `10 OUTPUT 707;":MEASURE:AMPLITUDE:ANALYSIS ON"`

Query   `:MEASure:AMPLitude:ANALysis?`

Returned Format   `[:MEASure:AMPLitude:ANALysis:] {1 | 0}<NL>`

AMPLitude:DI?

Query   `:MEASure:AMPLitude:DI? {ONE | ZERO}`

Returns the deterministic interference measurement for the specified logic level. Uses the current noise and interference units.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example   `10 OUTPUT 707;":MEASURE:AMPLITUDE:DI? ONE "`

Returned Format   `[:MEASure:AMPLitude:DI] <value><NL>`

AMPLitude:EOPening?

Query   `:MEASure:AMPLitude:EOPening?`

Returns the vertical eye opening measurement. The eye opening is defined at the same bit error ratio as the total noise and total jitter measurments. Uses the current noise and interference units.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example   `10 OUTPUT 707;":MEASURE:AMPLITUDE:EOPENING?"`

Returned Format   `[:MEASure:AMPLitude:EOPening] <value><NL>`

AMPLitude:ISI?

Query   `:MEASure:AMPLitude:ISI? {ONE | ZERO}`

Returns the inter-symbol interference measurement for the specified logic level. Uses the current noise and interference units.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:ISI? ONE "
Returned Format	[:MEASure:AMPLitude:ISI] <value><NL>

AMPLitude:ISIVsbit?

Query	:MEASure:AMPLitude:ISIVsbit?

Returns definite-length block data. The data block contains ISI values for each bit that has been measured. Each ISI value is 32-bit floating point (4 bytes). The data block is followed by a linefeed termination character (0A hex). The ISI value has units as specified by "AMPLitude:UNITs" on page 247. Use the query "SINTegrity:PATTern?" on page 277 to return the corresponding bit sequence. Use the query "AMPLitude:ISIVsbit:BITS?" on page 242 to return a list of corresponding bits for which AMPLitude:ISIVsbit? has returned values.

> **NOTE**   This query returns data in the LSB (Least Significant Byte) first format. This format can affect the ability of your programs to correctly interpret the returned data as explained in "Definite-Length Block Response Data" on page 17.

Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:ISIVSBIT?"
Returned Format	[:MEASure:AMPLitude:ISIVsbit] <block data><NL>

AMPLitude:ISIVsbit:BITS?

Query	:MEASure:AMPLitude:ISIVsbit:BITS?

Returns definite-length block data. The data block contains the list of bits for which AMPLitude:ISIVsbit? has returned values. Each bit value is a 32-bit integer (4 bytes). The data block is followed by a linefeed termination character (0A hex).

> **NOTE**   This query returns data in the LSB (Least Significant Byte) first format. This format can affect the ability of your programs to correctly interpret the returned data as explained in "Definite-Length Block Response Data" on page 17.

Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:ISIVSBIT:BITS?"
Returned Format	[:MEASure:AMPLitude:ISIVsbit:BITS] <block data><NL>

AMPLitude:ISIVsbit:HIGHest?

Query	:MEASure:AMPLitude:ISIVsbit:HIGHest? {ONE	ZERO}

For the highest one or zero bit, returns the bit number and the measured ISI at that bit. The bit value is a 32-bit integer (4 bytes).

Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:ISIVSBIT:HIGHEST?"

Returned Format   `[:MEASure:AMPLitude:ISIVsbit:HIGHest] <bit>,<ISI value><NL>`

AMPLitude:ISIVsbit:LOWest?

Query   `:MEASure:AMPLitude:ISIVsbit:LOWest? {ONE | ZERO}`

For the lowest one or zero bit, returns the bit number and the measured ISI at that bit. The bit value is a 32-bit integer (4 bytes).

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example   `10 OUTPUT 707;":MEASURE:AMPLITUDE:ISIVSBIT:LOWEST?"`

Returned Format   `[:MEASure:AMPLitude:ISIVsbit:LOWest] <bit>,<ISI value><NL>`

AMPLitude:LEVel:CIDigits:LAGGing

Command   `:MEASure:AMPLitude:LEVel:CIDigits:LAGGing <num_digits>`

Specifies the minimum number of lagging consecutive identical digits to be uses in defining the one and zero levels.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example   `10 OUTPUT 707;":MEASURE:AMPLITUDE:LEVEL:CIDIGITS:LAGGING 5"`

Query   `:MEASure:AMPLitude:LEVel:CIDigits:LAGGing?`

Returned Format   `[:MEASure:AMPLitude:LEVel:CIDigits:LAGGing] <num_digits><NL>`

AMPLitude:LEVel:CIDigits:LEADing

Command   `:MEASure:AMPLitude:LEVel:CIDigits:LEADing <num_digits>`

Specifies the minimum number of leading consecutive identical digits to be uses in defining the one and zero levels.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example   `10 OUTPUT 707;":MEASURE:AMPLITUDE:LEVEL:CIDIGITS:LEADING 5"`

Query   `:MEASure:AMPLitude:LEVel:CIDigits:LEADing?`

Returned Format   `[:MEASure:AMPLitude:LEVel:CIDigits:LEADing] <num_digits><NL>`

AMPLitude:LEVel:DEFine

Command   `:MEASure:AMPLitude:LEVel:DEFine {AVERage | CIDigits}`

Specifies whether the one and zero levels are defined as the average one and zero levels, or whether they are specified in terms of a minimum number of consecutive identical digits.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example   `10 OUTPUT 707;":MEASURE:AMPLITUDE:LEVEL:DEFINE AVERAGE"`

Query	`:MEASure:AMPLitude:LEVel:DEFine?`
Returned Format	`[:MEASure:AMPLitude:LEVel:DEFine] {AVERage	CIDigits}<NL>`

AMPLitude:LOCation

Command	`:MEASure:AMPLitude:LOCation <location_value>`
	Specifies the location within the unit interval at which the noise and interference will be measured. The location is specified as a percentage. 0% corresponds to the left crossing point, 100% corresponds to the right crossing point, and 50% (the default) corresponds to the center of the eye. The specified location must be within the range of 5.0% and 95.0%.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:LOCATION 25"`
Query	`:MEASure:AMPLitude:LOCation?`
Returned Format	`[:MEASure:AMPLitude:LOCation] <location_value><NL>`

AMPLitude:OLEVel?

Query	`:MEASure:AMPLitude:OLEVel?`
	Returns the one level measurement. Always uses the vertical units of the channel. Use **"AMPLitude:ZLEVel?"** on page 247 to return the zero level measurement.
Restrictions	86100C (software revision A.07.00 and above) with Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:OLEVEL?"`
Returned Format	`[:MEASure:AMPLitude:OLEVel] <value><NL>`

AMPLitude:PI?

Query	`:MEASure:AMPLitude:PI? {ONE	ZERO}`
	Returns the dual-dirac periodic interference measurement for the specified logic level.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:PI? ONE "`
Returned Format	`[:MEASure:AMPLitude:PI] <value><NL>`

AMPLitude:PIRMs?

Query	`:MEASure:AMPLitude:PIRMs? {ONE	ZERO}`
	Returns the RMS periodic interference measurement for the specified logic level. Uses the current noise and interference units.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:PIRMS? ONE "
Returned Format	[:MEASure:AMPLitude:PIRMs] <value><NL>

AMPLitude:Q?

Query	:MEASure:AMPLitude:Q? Returns the Q measurement.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:Q?"
Returned Format	[:MEASure:AMPLitude:Q] <value><NL>

AMPLitude:RINoise?

Query	:MEASure:AMPLitude:RINoise? [{OMAMplitude | OLEVel}] Returns the RIN (Relative Intensity Noise) measurement value which is only available for optical signals. It can be based on the one level (OLEVel) or based on the RIN OMA measurement (OMAMplitude). If no argument is given, the measurement value returned is for the currently selected type of RIN measurement as specified by the **"AMPLitude:RINoise:DEF"** on page 245.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:RINOISE? OLEVel"
Returned Format	[:MEASure:AMPLitude:RINoise] <value><NL>

AMPLitude:RINoise:DEF

Command	:MEASure:AMPLitude:RINoise:DEF {OMAMplitude | OLEVel} Defines the type of RIN measurement performed. OLEVel specifies a RIN measurement that is based on the one level. OMAMplitude specifies a RIN OMA measurement.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	10 OUTPUT 707;":MEASURE:AMPLITUDE:RINOISE:DEF OMAMplitude"
Query	Returns the type of RIN measurement currently selected. :MEASure:AMPLitude:RINoise:DEF?
Returned Format	[:MEASure:AMPLitude:RINoise:DEF] {OMAMplitude | OLEVel}<NL>

AMPLitude:RINoise:UNITs

Command	:MEASure:AMPLitude:RINoise:UNITs { DB | DBHZ } Specifies the units for the Random Intensity Noise (RIN) measurement. The measurement is only available for optical signals. Units of dB/Hz are only available when a reference filter is in use.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:RINOISE:UNITS DB"`
Query	`:MEASure:AMPLitude:RINoise:UNITs?`
Returned Format	`[:MEASure:AMPLitude:RINoise:UNITs] { DB	DBHZ }<NL>`

AMPLitude:RN?

Query	`:MEASure:AMPLitude:RN? {ONE	ZERO}`
	Returns the random noise measurement for the specified logic level. Uses the current noise and interference units.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:RN? ONE"`
Returned Format	`[:MEASure:AMPLitude:RN] <value><NL>`

AMPLitude:RNSTabilize

Command	`:MEASure:AMPLitude:RNSTabilize {ON	1	OFF	0}`
	Turns the RN stabilization on or off.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:RNSTABILIZE ON"`
Query	`:MEASure:AMPLitude:RNSTabilize?`
Returned Format	`[:MEASure:AMPLitude:RNSTabilize] {1	0}<NL>`

AMPLitude:RNSValue

Command	`:MEASure:AMPLitude:RNSValue {ONE	ZERO},<RN_value>`
	Sets the RN stabilization value for the specified signal level.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:RNSVALUE ONE,1E-5"`
Query	`:MEASure:AMPLitude:RNSValue?`
Returned Format	`[:MEASure:AMPLitude:RNSValue] {ONE	ZERO},<RN_value><NL>`

AMPLitude:SAMPlitude?

Query	`:MEASure:AMPLitude:SAMPlitude?`
	Returns the signal amplitude measurement. Always uses the vertical units of the channel.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:SAMPLITUDE?"`
Returned Format	`[:MEASure:AMPLitude:SAMPlitude] <value><NL>`

AMPLitude:TI?

Query	`:MEASure:AMPLitude:TI? {ONE	ZERO}`
	Returns the total interference measurement for the specified logic level. Uses the current noise and interference units.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:TI? ONE"`
Returned Format	`[:MEASure:AMPLitude:TI] <value><NL>`

AMPLitude:TI:DEFine

Command	`:MEASure:AMPLitude:TI:DEFine <ber_level>`
	Specifies the Bit Error Ratio (BER) at which the total interference (TI) is measured. The default value is 10^{-12}. The value is shared with the total jitter (TJ) measurement. **Refer to "JITTer:TJ:DEFine" on page 269**.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:TI:DEFINE -50"`
Query	`:MEASure:AMPLitude:TI:DEFine?`
Returned Format	`[:MEASure:AMPLitude:TI:DEFine] <ber_level><NL>`

AMPLitude:UNITs

Command	`:MEASure:AMPLitude:UNITs {CHANnel	UAMPlitude}`
	Sets and queries the units of the noise and interference measurements. The units can be set either to the current vertical channel units or to unit amplitude. If the units are unit amplitude, the measurements are normalized to the signal amplitude.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:UNITS CHANNEL"`
Query	`:MEASure:AMPLitude:UNITs?`
Returned Format	`[:MEASure:AMPLitude:UNITs] {CHANnel	UAMPlitude}<NL>`

AMPLitude:ZLEVel?

Query	`:MEASure:AMPLitude:ZLEVel?`
	Returns the zero level measurement Always uses the vertical units of the channel. Use **"AMPLitude:OLEVel?"** on page 244 to return the one level measurement.

Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Returned Format	`[:MEASure:AMPLitude:ZLEVel] <value><NL>`
Example	`10 OUTPUT 707;":MEASURE:AMPLITUDE:ZLEVEL?"`

ANNotation

Command	`:MEASure:ANNotation {ON	1	OFF	0}`
	Turns measurement annotations on or off. If there are no active measurements, you can still turn on or off measurement annotations. The instrument will remain in the defined state and will be activated (if on) the next time measurements are performed.
Mode	All instrument modes.
Query	`:MEASure:ANNotation?`
Returned Format	`[:MEASure:ANNotation] {1	0}`
Example	`10 OUTPUT 707;":MEASURE:ANNOTATION ON"`

APOWer

Command	`:MEASure:APOWer {WATT	DECibel} [,{CHANnel<N>}]`
	Measures the average power. Sources are specified with the MEASure:SOURce command or with the optional parameter following the APOWer command. The average optical power can only be measured on an optical channel input. For channels, this value is dependent on the type of module and its location in the instrument. It will work only on optical channels.
Mode	Eye or Oscilloscope modes
Query	`:MEASure:APOWer? {WATT	DECibel} [,{CHANnel<N>}]`
Returned Format	`[:MEASure:APOWer] <average_power>[,<result_state>]<NL>`
	If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:APOWER? WATT"`

APOWer:CORRection

Command	`:MEASure:APOWer:CORRection {ON	1	OFF	0}`
	Applies dark calibration data to Average Power measurements. This removes an optical channel's dark current from the measurement results. Perform an extinction ratio calibration on a module's optical channel to create the dark current factor. Remove any input from the optical channel before performing the extinction ratio calibration.
Mode	Eye or Oscilloscope modes
Query	`:MEASure:APOWer:CORRection?`
Returned Format	`[:MEASure:APOWer:CORRection] {ON	OFF}<NL>`

CGRade:AMPLitude

Command   `:MEASure:CGRade:AMPLitude [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the eye amplitude of the displayed source. The eye amplitude is the difference between the one level and the zero level.

Mode   Eye mode only.

Query   `:MEASure:CGRade:AMPLitude? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:AMPLitude] <eye_amplitude>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Examples   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:CGRADE:AMPLITUDE?"`

CGRade:BITRate

Command   `:MEASure:CGRade:BITRate [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the bit rate of the displayed signal. The bit rate is the number of bits per second. It is measured as the inverse of the bit period. In NRZ eye mode, the bit period is the time interval between two successive crossing points of an eye. In RZ eye mode, the bit period is the time interval between the 50% falling (or rising) edges of 2 consecutive eyes.

Mode   Eye mode only.

Query   `:MEASure:CGRade:BITRate? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returns the bit rate in bits/s.

Returned Format   `[:MEASure:CGRade:BITRate] <bit_rate>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   The following example measures the bit rate of the displayed eye.
`10 OUTPUT 707;":MEASURE:CGRADE:BITRATE"`

CGRade:COMPlete

Command   `:MEASure:CGRade:COMPlete <comp_hits>`

Sets the color grade measurement completion criterion. Auto skew (page 139) also uses the current color grade measurement completion criterion. If auto skew fails to make the bit rate measurement or determine the time of the crossing points needed to compute the skew, it may be necessary to increase the color grade completion criterion. Increasing the value will increase the time for auto skew to complete, allowing it to collect more data points before executing the bit rate and crossing time measurements. <comp_hits> is the number of hits that the peak-numbers-of-hits, in the color grade database, must equal or exceed before a color grade measurement is executed.

Mode   Eye mode

Query   `:MEASure:CGRade:COMPlete?`

Returns the current setting for color grade completion.

Returned Format   `[:MEASure:CGRade:COMPlete] <comp_hits><NL>`

A color grade measurement query will return 9.99999E+37 until the measurement is complete.

Example   The following example sets the completion criterion to 10 hits.
```
10 OUTPUT 707;":MEASURE:CGRADE:COMPLETE 10"
```

CGRade:CRATio

Command   `:MEASure:CGRade:CRATio <format> [,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the contrast ratio of the RZ (Return-to-Zero) eye diagram on the color graded display. The dark level or dc offset of the input channel must have been previously calibrated. See "ERATio:STARt" on page 134 to perform a dark level calibration. If the source is not set, the lowest numbered signal that is on will be the source of the measurements. <format> is {RATio | DECibel | PERCent}.

Mode   Eye mode only. Ensure that the eye type is set to RZ. See "DEFine" on page 255.

Query   `:MEASure:CGRade:CRATio? <format> [,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:CRATio] <contrast_ratio>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:CGRADE:CRATIO? PERCENT"
```

CGRade:CROSsing

Command   `:MEASure:CGRade:CROSsing [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the crossing level percent of the current eye diagram on the color grade or gray scale display. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal that is on will be the source of the measurement.

Mode   Eye mode only. Ensure that the eye type is set to NRZ. See "DEFine" on page 255.

Query   `:MEASure:CGRade:CROSsing? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:CROSsing] <crossing_level>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:CGRade:CROSsing?"
```

CGRade:DCDistortion

Command   `:MEASure:CGRade:DCDistortion <format>[,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the duty cycle distortion on the eye diagram of the current color grade or gray scale display. The parameter specifies the format for reporting the measurement. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal that is on will be the source of the measurement. <format> is {TIME | PERCent}.

Mode   Eye mode only. Ensure that the eye type is set to NRZ. See "DEFine" on page 255.

Query   `:MEASure:CGRade:DCDistortion? <format> [,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:DCDistortion] <duty_cycle_distortion>[,<result_state>] <NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:CGRADE:DCDistortion? PERCENT"
```

CGRade:DCYCle

Command   `:MEASure:CGRade:DCYCle [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the duty cycle of the RZ (Return-to-Zero) eye diagram on the color graded display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement.

Mode   Eye mode only. Ensure that the eye type is set to RZ. See "DEFine" on page 255.

Query   `:MEASure:CGRade:DCYCle? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:DCYCle]<duty_cycle>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":MEASURE:CGRADE:DCYCle"`

CGRade:EHEight

Command   `:MEASure:CGRade:EHEight [{RATio | DECibel} [,{CHANnel<N> | FUNCtion<N> | CGMemory}]]`

Measures the eye height on the eye diagram of the current color grade display. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement. The default argument is RATio. If the channel is specified, the format (RATio or DECibel) must also be specified.

Mode   Eye mode only.

Query   `:MEASure:CGRade:EHEight? [{RATio | DECibel} [,{CHANnel<N> | FUNCtion<N> | CGMemory}]]`

Returned Format   `[:MEASure:CGRade:EHEight] <eye_height>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Examples
```
20 OUTPUT 707;":MEASURE:CGRADE:EHEight"
20 OUTPUT 707;":MEASURE:CGRADE:EHEight RATio"
20 OUTPUT 707;":MEASURE:CGRADE:EHEight RATio,CHANnel3"
20 OUTPUT 707;":MEASURE:CGRADE:EHEight CHANnel3" // Invalid command
```

CGRade:ERATio

Command   `:MEASure:CGRade:ERATio {RATio | DECibel | PERCent}[,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the extinction ratio on the eye diagram of the current color grade display. The dark level or dc offset of the input channel must have been previously calibrated. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement.

Mode   Eye mode only.

Query   `:MEASure:CGRade:ERATio? {RATio | DECibel | PERCent} [,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:ERATio] <extinction_ratio>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:CGRADE:ERATIO? RATIO"
```

CGRade:ERFactor

Command   `:MEASure:CGRade:ERFactor CHANnel<N>,{ON|OFF}[,<correction_factor>]`

Turns on or off the extinction ratio correction and, optionally, to set the correction factor used when correction is turned on. <N> specifies a channel, where <N> is 1, 2, 3 or 4. Each channel has its own setting for on or off and for correction factor. <correction_factor> is a percentage value that is used to offset the measured extinction ratio value. Correction factor is always specified as a percentage, regardless of the format or units specified for extinction ratio measurement results.

Mode   Eye mode only.

Restrictions   86100D or 86100C (Software revision A.04.00 and above).

Query   `:MEASure:CGRade:ERFactor? CHANnel<N>`

Returns the extinction ratio correction settings for the specified channel. A correction factor value is returned regardless of whether correction is on or off.

Returned Format   `[:MEASure:CGRade:ERFactor] {ON|OFF}<NL>`

Example   `10 OUTPUT 707; ":MEASure:CGRade:ERFactor CHANnel4,ON,80"`

CGRade:ESN

Command   `:MEASure:CGRade:ESN [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the eye signal-to-noise. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement. This measurement was called Q-factor in the 83480A/54750A.

Mode   Eye mode only.

Query   `:MEASure:CGRade:ESN? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:ESN] <signal_to_noise>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:CGRADE:ESN?"`

CGRade:EWIDth

Command   `:MEASure:CGRade:EWIDth [{RATio | TIME} [,{CHANnel<N> | FUNCtion<N> | CGMemory}]]`

Measures the eye width on the eye diagram of the current color grade display. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement. The default format is TIME.

Mode   Eye mode only.

Query   `:MEASure:CGRade:EWIDth? [{RATio | TIME} [,{CHANnel<N> | FUNCtion<N> | CGMemory}]]`

Returned Format   `[:MEASure:CGRade:EWIDth] <eye_width>[,<result_state>] <NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:CGRADE:EWIDTH?"`

CGRade:JITTer

Command   `:MEASure:CGRade:JITTer {PP | RMS} [,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the jitter at the eye diagram crossing point in Eye mode. In Oscilloscope mode, it measures the mean of the first complete rising or falling edge. The parameter specifies the format in which the results are reported: peak-to-peak or RMS. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement. The optional source argument can be a channel, function, or color-grade memory. Use the CGMemory argument in Eye mode only.

Mode   Eye or Oscilloscope modes.

Query   `:MEASure:CGRade:JITTer? {PP | RMS} [,{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returns the jitter of the color grade database. In Oscilloscope mode, the measurement turns on the color grade database but not the color grade display persistence.

Returned Format   `[:MEASure:CGRade:JITTer] <jitter> [,<result_state>] <NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:CGRADE:JITTER? RMS"`

CGRade:OFACtor

Command   `:MEASure:CGRade:OFACtor [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the opening factor of the RZ (Return-to-Zero) eye diagram on the color graded display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement.

Mode   Eye mode only. Ensure that the eye type is set to RZ. See "DEFine" on page 255.

Query   `:MEASure:CGRade:OFACtor? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:OFACtor] <opening_factor>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:CGRade:OFACtor?"`

CGRade:OLEVel

Command   `:MEASure:CGRade:OLEVel [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures the logic one level inside the eye window. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement. .

Mode   Eye mode only.

Query   `:MEASure:CGRade:OLEVel? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Returned Format   `[:MEASure:CGRade:OLEVel] <logic_one_level>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:CGRADE:OLEVEL?"`

CGRade:PEAK?

Query   `:MEASure:CGRade:PEAK? [<source>]`

Returns the maximum number of hits of the color grade display. The data for color grade display is the same as for gray scale display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement. <source> is {CHANnel<N> | FUNCtion<N> | CGMemory}.

Mode   Eye or Oscilloscope modes.

Returned Format   `[:MEASure:CGRade:PEAK] <number_of_hits>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:CGRADE:PEAK?"
```

CGRade:PWIDth

Command   `:MEASure:CGRade:PWIDth [<source>]`

Measures the pulse width of the eye diagram on the color graded display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement. <source> is {CHANnel<N> | FUNCtion<N> | CGMemory}.

Mode   Eye mode only. Ensure that the eye type is set to RZ. See "DEFine" on page 255.

Query   `:MEASure:CGRade:PWIDth? [<source>]`

This query returns the pulse width of the color graded display.

Returned Format   `[:MEASure:CGRade:PWIDth] <pulse_width>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASure:CGRade:PWIDth?"
```

CGRade:SMOothing

Command   `:MEASure:CGRade:SMOothing <N-point>`

This eye-mode command improves measurements of eye diagrams that have noisy eyes or eyes that have few samples. The command turns on and sets the amount of smoothing that is applied to the crossing-region histograms in the form of an an *N-point* moving average.

Eye measurement algorithms can become confused if an eye diagram's different crossing regions include a dissimilar number of edges. This trait is an artifact of using a divided trigger with a relatively short pattern. Applying smoothing increases the measurement's tolerance of this situation. Start with a smoothing value of 9 and increase as needed for extremely noisy or jittery eye diagrams. Odd values are preferred as they avoid biasing the histograms towards one direction.

The default argument (*N-point*) is 1, which turns off the added smoothing. Values of 2 through 100 provide increasing amounts of smoothing.

Mode   Eye mode only.

Restrictions   Requires software revision A.09.01 and above.

Query   `:MEASure:CGRade:SMOothing?`

Returned Format   `[:MEASure:CGRade:SMOothing] <N-point><NL>`

Example	`10 OUTPUT 707;":MEASure:CGRade:SMOothing 9"`

CGRade:SOURce

Command	`:MEASure:CGRade:SOURce {CHANnel<N> | FUNCtion<N> | CGMemory}`

Sets the default source for color grade-gray scale measurements. If this source is not set, the lowest numbered color grade-gray scale signal that is on will be the source of the measurements. This command is similar to the :MEASure:SOURce command, with the exception of specifying a color grade-gray scale signal. <N> is an integer, from 1 through 4.

Mode	Eye and Oscilloscope modes.
Query	`:MEASure:CGRade:SOURce?`
Returned Format	`[:MEASure:CGRade:SOURce] {CHANnel<N> | FUNCtion<N> | CGMemory}<NL>`
Example	`10 OUTPUT 707;":MEASure:CGRade:SOURce CHANNEL1"`

CGRade:ZLEVel

Command	`:MEASure:CGRade:ZLEVel [{CHANnel<N> | FUNCtion<N> | CGMemory}]`

Measures logic zero level inside the eye window on the eye diagram of the current color grade display. If the source is not set, the lowest numbered signal display that is on will be the source of the measurement.

Mode	Eye mode only.
Query	`:MEASure:CGRade:ZLEVel? [{CHANnel<N> | FUNCtion<N> | CGMemory}]`
Returned Format	`[:MEASure:CGRade:ZLEVel] <zero_level>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASure:CGRade:ZLEVel?"`

CLEar

Command	`:MEASure:CLEar`

Clears the measurement results from the screen. It is identical to the :MEASure:SCRatch command.

Example	`10 OUTPUT 707;":MEASure:CLEAR"`

DEFine

Command	`:MEASure:DEFine {CGRade, DELTatime, EWINdow, RZEWindow, THResholds,TOPBase, TREFerence}`

Sets up the definition for measurements. The following paragraphs define each argument. Changing these values may affect other measure commands. Table 41 on page 256 identifies the relationships between user-DEFined values and other MEASure commands.

CGRade	`:MEASure:DEFine CGRade,{RZ | NRZ}`

Defines the eye type of the data pattern: return-to-zero (RZ) or non-return-to-zero (NRZ).

DELTatime	`:MEASure:DEFine DELTatime, {<start edge_direction>,<start edge_number>,<start edge_position>,<stop edge_direction>,<stop edge_number>,<stop edge_position>}`

Sets up edge parameters for delta time measurement. <edge_direction> is {RISing | FALLing | EITHer}. <edge_number> is an integer, from 1 to 20. <edge_position> is {UPPer | MIDDle | LOWer}.

EWINdow   `:MEASure:DEFine EWINdow,<ewind1pct>,<ewind2pct>`

<ewind1pct> and <ewind2pct> are real floating-point numbers (rounded to the nearest tenth) that specify an eye window as a percentage of the bit period unit interval. If one source is specified, both parameters apply to that signal. If two sources are specified, the measurement is from the first positive edge on source 1 to the second negative edge on source 2.

RZEWindow   `:MEASure:DEFine RZEWindow, <%bit_rate>`

Defines the width of an RZ eye window as a percentage of the bit rate.

THResholds   `:MEASure:DEFine THResholds,{{STANdard} |`
`{PERCent,<upper_pct>,<middle_pct>,<lower_pct>} |`
`{UNITs,<upper_volts>,<middle_volts>,<lower_volts>}}`

Where <upper_pct>, <middle_pct>, and <lower_pct> are integers ranging from –25 to 125. <upper_units>, <middle_units>, and <lower_units> are real numbers specifying amplitude units.

TOPBase   `:MEASure:DEFine TOPBase,{{STANdard} |{<top_volts>,<base_volts>}}`

<top_volts> and <base_volts> are real numbers specifying voltage.

TREFerence   `:MEASure:DEFine TREFerence,{TBASe | ONEZero}`

Selects a threshold reference for use in risetime and falltime measurements. The threshold reference can set to either V_{top} and V_{base} (TBASe) or the one and zero levels (ONEZero). Eye mode is required and the :MEASure:DEFine command's THResholds argument must *not* be set to UNITs.

Restrictions   The TREFerence argument, requires software revision A.07.00 and above.

Query   `:MEASure:DEFine? {CGRade | DELTatime | EWINdow | RZEWindow | THResholds | TOPBase`
`| TREFerence}`

Returned Format   `[:MEASure:DEFine] CGR {RZ | NRZ}`
`[:MEASure:DEFine] DELT, {<start edge_direction>,<start edge_number>,<start`
`edge_position>,<stop edge_direction>,<stop edge_number>,<stop edge_position>}<NL>`
`[:MEASure:DEFine] EWIN,<signal_type><NL>`
`[:MEASure:DEFine] RZAEW,<%bit_rate><NL>`
`[:MEASure:DEFine] THR {{STAN} | {PERcent,<upper_pct>,<middle_pct>,<lower_pct>} |`
`{VOLTage, <upper_volts>,<middle_volts>,<lower_volts>}}<NL>`
`[:MEASure:DEFine] TOPB {{STAN} |{<top_volts>,<base_volts>}}<NL>`
`[:MEASure:DEFine] TREF {TBASe | ONEZero}`

NOTE   Using "mV" or "V" following the numeric value for the voltage value will cause Error 138-Suffix not allowed. Instead, use the convention for the suffix multiplier as described in "Command Syntax" on page 14.

Example   `10 OUTPUT 707;":MEASURE:DEFINE? THRESHOLDS"`

Table 41   :MEASure:DEFine Interactions (Sheet 1 of 2)

MEASure Commands	THResholds	TOPBase	EWINdow	CGRAde	DELTatime	TREFerence
RISEtime	x	x				x
FALLtime	x	x				x
PERiod	x	x				
FREQuency	x	x				

Table 41   :MEASure:DEFine Interactions (continued) (Sheet 2 of 2)

MEASure Commands	THResholds	TOPBase	EWINdow	CGRAde	DELTatime	TREFerence
VTOP		x				
VBASe		x				
VAMPlitude		x				
PWIDth	x	x				
NWIDth	x	x				
OVERshoot	x	x				
DUTYcycle	x	x				
DELTatime	x	x				
VRMS	x	x				
PREShoot	x	x				
VLOWer	x	x				
VMIDdle	x	x				
VUPPer	x	x				
VAVerage	x	x				
DELTatime	x	x			x	
CGRade:CRATio			x	x		
CGRade:CROSsing			x	x		
CGRade:DCDistortion	x			x		x
CGRade:DCYCle	x			x		x
CGRade:ERATio			x			
CGRade:EHEight			x			
CGRade:ESN			x			
CGRade:OFACtor				x		
CGRade:OLEVel			x			
CGRade:PWIDth	x					x
CGRade:ZLEVel			x			

DELTatime

Command   :MEASure:DELTatime [<source>[,<source>]]

Measures the time delay between two edges; it is the time difference from the first specified edge on one source to the next specified edge on another source. If no source is specified, then the sources specified using the :MEASure:SOURce command are used. If only one source is specified, then the edges used for computing delta time belong to that source. If two sources are specified, then the first

edge used in computing to delta time belongs to the first source and the second edge belongs to the second source. <source> is {CHANnel<N>| FUNCtion<N> | WMEMory<N> | RESPonse <N>} where <N> is an integer, from 1 through 4.

Mode	Oscilloscope and TDR modes
Query	`:MEASure:DELTatime? [<source>[,<source>]]`
Returned Format	`[:MEASure:DELTatime] <delta_time> [,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Examples	`10 OUTPUT 707;":MEASURE:DELTATIME CHANNEL1,CHANNEL2"`

```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:DELTATIME?"
```

NOTE   When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

DUTYcycle

Command	`:MEASure:DUTYcycle [{CHANnel<N>	FUNCtion<N>	WMEMory<N>}]`

Measures the ratio of the positive pulse width to the period. Sources are specified with the MEASure:SOURce command or with the optional parameter following the DUTYcycle command. <N> for channels is dependent on the type of plug-in and its location in the instrument. For functions: 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4.

Mode	Oscilloscope mode only.
Query	`:MEASure:DUTYcycle? [{CHANnel<N>	FUNCtion<N>	WMEMory<N>}]`
Returned Format	`[:MEASure:DUTYcycle] <duty_cycle>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:DUTYCYCLE?"`

FALLtime

Command	`:MEASure:FALLtime [{CHANnel<N>	FUNCtion<N>	RESPonse<N>	WMEMory<N>	CGRade}]`

Measures the time at the upper threshold of the falling edge, measures the time at the lower threshold of the falling edge, then calculates the fall time. Sources are specified with the MEASure:SOURce command or with the optional parameter following the FALLtime command. The first displayed falling edge is used for the fall-time measurement. Therefore, for best measurement accuracy, set the sweep speed as fast as possible while leaving the falling edge of the waveform on the display.

Fall time = time at lower threshold point – time at upper threshold point.

CHANnel<N>, FUNCtion<N>, RESPonse<N> and WMEMory<N> apply in Oscilloscope and TDR modes only. CGRade applies in Eye mode only. <N> for channels, functions, TDR responses and waveform memories is 1, 2, 3, or 4.

Mode	All instrument modes except Jitter Mode.

| Query | :MEASure:FALLtime?[{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N> | CGRade}] |

| Returned Format | [:MEASure:FALLtime] <falltime>[,<result_state>]<NL> |

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

| Example | 10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:FALLTIME?" |

FREQuency

| Command | :MEASure:FREQuency [{CHANnel<N> | FUNCtion<N> | WMEMory<N>}] |

Measures the frequency of the first complete cycle on the screen using the mid-threshold levels of the waveform (50% levels if standard measurements are selected). The source is specified with the MEASure:SOURce command or with the optional parameter following the FREQuency command.

The algorithm is:

If the first edge on screen is rising, then

frequency = 1/(time at second rising edge – time at first rising edge)

else,

frequency = 1/(time at second falling edge – time at first falling edge).

<N> for channels is dependent on the type of plug-in and its location in the instrument. For functions: 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4.

| Mode | Oscilloscope mode only |

| Query | :MEASure:FREQuency? [{CHANnel<N> | FUNCtion<N> | WMEMory<N>}] |

Returns the measured frequency, in Hertz.

| Returned Format | [:MEASure:FREQuency] <frequency>[,<result_state>]<NL> |

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

| Example | 10 OUTPUT 707;":MEASURE:FREQUENCY" |

HISTogram:HITS?

| Query | :MEASure:HISTogram:HITS? [{HISTogram}] |

Returns the number of hits within the histogram. The source can be specified with the optional parameter following the HITS query. The HISTogram:HITS? query only applies to the histogram.

| Returned Format | [:MEASure:HISTogram:HITS] <hits>[,<result_state>]<NL> |

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

| Example | 10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:HISTOGRAM:HITS?" |

HISTogram:M1S?

| Query | :MEASure:HISTogram:M1S? [{HISTogram}] |

Returns the percentage of points that are within one standard deviation of the mean of the histogram. The source can be specified with the optional parameter following the M1S query. The HISTogram:M1S? query only applies to the histogram waveform.

Returned Format   `[:MEASure:HISTogram:M1S] <percentage>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:HISTOGRAM:M1S?"`

HISTogram:M2S?

Query   `:MEASure:HISTogram:M2S? [{HISTogram}]`

Returns the percentage of points that are within two standard deviations of the mean of the histogram. The sources can be specified with the optional parameter following the M2S query. The HISTogram:M2S? query only applies to the histogram waveform.

Returned Format   `[:MEASure:HISTogram:M2S] <percentage>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:HISTOGRAM:M2S?"`

HISTogram:M3S?

Query   `:MEASure:HISTogram:M3S? [{HISTogram}]`

Returns the percentage of points that are within three standard deviations of the mean of the histogram. The source can be specified with the optional parameter following the M3S query. The HISTogram:M3S? query only applies to the histogram waveform.

Returned Format   `[:MEASure:HISTogram:M3S] <percentage>[,<result_state>] <NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:HISTOGRAM:M3S?"`

HISTogram:MEAN?

Query   `:MEASure:HISTogram:MEAN? [{HISTogram}]`

Returns the mean of the histogram. The mean of the histogram is the average value of all the points in the histogram. The source can be specified with the optional parameter following the MEAN query. The HISTogram:MEAN? query only applies to the histogram waveform.

Returned Format   `[:MEASure:HISTogram:MEAN] <mean>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:HISTOGRAM:MEAN?"`

HISTogram:MEDian?

Query   `:MEASure:HISTogram:MEDian? [{HISTogram}]`

Returns the median of the histogram. The median of the histogram is the time or voltage of the point at which 50% of the histogram is to the left or right (above or below for vertical histograms). The source can be specified with the optional parameter following the MEDian query. The HISTogram:MEDian? query only applies to the histogram waveform.

Returned Format   `[:MEASure:HISTogram:MEDian] <median>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:HISTOGRAM:MEDIAN?"

HISTogram:PEAK?

Query   :MEASure:HISTogram:PEAK? [{HISTogram}]

Returns the number of hits in the histogram's greatest peak. The source can be specified with the optional parameter following the PEAK query. The HISTogram:PEAK? query only applies to the histogram waveform.

Returned Format   [:MEASure:HISTogram:PEAK] <hits>[,<result_state>]<NL>

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:HISTOGRAM:PEAK?"

HISTogram:PP?

Query   :MEASure:HISTogram:PP? [{HISTogram}]

Returns the width of the histogram. The width is measured as the time or voltage of the last histogram bucket with data in it minus the time or voltage of the first histogram bucket with data in it. The source can be specified with the optional parameter following the PP query. The HISTogram:PP? query only applies to the histogram waveform.

Returned Format   [:MEASure:HISTogram:PPos] <width>[,<result_state>]<NL>

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:HISTOGRAM:PP?"

HISTogram:PPOSition?

Query   :MEASure:HISTogram:PPOSition? [{HISTogram}]

Returns the position of the greatest peak of the histogram. If there is more than one peak, then it returns the position of the first peak from the lower boundary of the histogram window for vertical axis histograms. Otherwise, in the case of horizontal axis histograms, it returns the position of the first peak from the leftmost boundary of the histogram window. The optional parameter MEASure:SOURce command can be used to specify the source for the measurement. This query can only be applied to histogram data, therefore the histogram must be turned on in order to use this query.

Returned Format   [:MEASure:HISTogram:PPosition] <position>[,<result_state>]<NL>

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:HISTOGRAM:PPOSITION? HISTOGRAM"

HISTogram:SCALe?

Query   :MEASure:HISTogram:SCALe? [{HISTogram}]

Returns the scale of the histogram in hits per division. The source can be specified with the optional parameter following the SCALe query. The HISTogram:SCALe? query only applies to the histogram waveform.

Returned Format	`[:MEASure:HISTogram:SCALe] <scale>[,<result_state>]<NL>`
	If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:HISTOGRAM:SCALE?"`

HISTogram:STDDev?

Query	`:MEASURE:HISTogram:STDDev? [{HISTogram}]`
	Returns the standard deviation of the histogram. The source can be specified with the optional parameter following the STDDev query. The HISTogram:STDDev? query only applies to the histogram waveform.
Returned Format	`[:MEASure:HISTogram:STDDev] <standard_deviation>[,<result_state>]<NL>`
	If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:HISTOGRAM:STDDEV?"`

JITTer:DCD?

Query	`:MEASure:JITTer:DCD?`
	Returns the duty cycle distortion value measured on the current source.
Restrictions	Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).
Returned Format	`[:MEASure:JITTer:DCD] <value><NL>`
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASure:JITTer:DCD?"`

JITTer:DDJ?

Query	`:MEASure:JITTer:DDJ?`
	Returns the data-dependent jitter value measured on the current source.
Restrictions	Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).
Returned Format	`[:MEASure:JITTer:DDJ] <value><NL>`
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASure:JITTer:DDJ?"`

JITTer:DDJVsbit?

Query	`:MEASure:JITTer:DDJVsbit?`
	For each measured edge, returns the DDJ values as definite-length block data. DDJ values are returned for only the edge types specified by the command MEASure:JITTer:EDGE. Each DDJ value is 32-bit floating point (4 bytes) returned in MSB (Most Significant Byte) first order. MSB first order is used by microprocesors like Motorola where the most significant byte resides at the lower address. When using a LSB (Least Significant Byte) first microprocessor, like Intel's, you will need to reverse the byte order of the returned data. The data block is followed by a linefeed terminator character (0A hex). The DDJ value has units of time or unit interval as specified by **"JITTer:UNITs"** on page 269. Use

the query **"JITTer:PATTern?"** on page 267 to return the edge type values. Use the query **"JITTer:DDJVsbit:BITS?"** on page 263 to return a list of corresponding bits for which JITTer:DDJVsbit? has returned values.

This query returns data in the LSB (Least Significant Byte) first format. This format can affect the ability of your programs to correctly interpret the returned data as explained in **"Definite-Length Block Response Data"** on page 17.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:DDJVsbit] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:JITTer:DDJVsbit?"`

JITTer:DDJVsbit:BITS?

Query   `MEASure:JITTer:DDJVsbit:BITS?`

Returns definite-length block data. The data block contains the list of bits for which JITTer:DDJVsbit? has returned values. Each bit value is a 32-bit integer (4 bytes) returned in MSB (Most Significant Byte) first order. MSB first order is used by microprocesors like Motorola where the most significant byte resides at the lower address. When using a LSB (Least Significant Byte) first microprocessor, like Intel's, you will need to reverse the byte order of the returned data. The data block is followed by a linefeed termination character (0A hex).

NOTE   This query returns data in the LSB (Least Significant Byte) first format. This format can affect the ability of your programs to correctly interpret the returned data as explained in **"Definite-Length Block Response Data"** on page 17.

Restrictions   86100D or 86100C (Software revision A.07.00 and above). Jitter Mode.

Example   `10 OUTPUT 707;":MEASURE:JITTER:DDJVSBIT:BITS?"`

Returned Format   `[:MEASure:JITTer:DDJVsbit:BITS] <block data><NL>`

JITTer:DDJVsbit:EARLiest?

Query   `:MEASure:JITTer:DDJVsbit:EARLiest?`

Returns comma-separated values (string) for the earliest measured edge in the DDJ vs. bit graph. The string includes the bit number followed by the DDJ value. The DDJ value has units of time or unit interval as specified by the :MEASure:JITTer:UNITs command.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.20 and above). Option 200, Enhanced Jitter Analysis Software.

Returned Format   `[:MEASure:JITTer:DDJVsbit:EARLiest] <string><NL>`
The following is an example of a returned string: "30, 3.4339e-12"

Example   `10 OUTPUT 707;":MEASURE:JITTER:DDJVSBIT:EARLIEST?"`

JITTer:DDJVsbit:LATest?

Query   `:MEASure:JITTer:DDJVsbit:LATest?`

Returns comma-separated values (string) for the latest measured edge in the DDJ vs. bit graph. The string includes the bit number followed by the DDJ value. The DDJ value has units of time or unit interval as specified by the :MEASure:JITTer:UNITs command.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.20 and above). Option 200, Enhanced Jitter Analysis Software.

Returned Format   `[:MEASure:JITTer:DDJVsbit:LATest] <string><NL>`

The following is an example of a returned string: "30, 3.4339e-12"

Example   `10 OUTPUT 707;":MEASURE:JITTER:DDJVSBIT:LATEST?"`

JITTer:DJ?

Query   `:MEASure:JITTer:DJ?`

This query returns the deterministic jitter value measured on the current source.

Restrictions   Jitter mode. Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:DJ] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:JITTer:DJ?"`

JITTer:EBITs?

Query   `:MEASure:JITTer:EBITs?`

Returns an ordered list of edge bit numbers returned as definite-length block data. Each value is the number of the bit in the pattern preceding the edge transition and is in the range of 0 to PatternLength-1. Each bit number is a four byte integer. Only the edges of the type specified by the command :MEASure:JITTer:EDGE are included in the list. The data block is followed by a terminator character, 0A hex (linefeed). This query will return an incomplete list of edges, if all of the data needed to determine the pattern has not yet been acquired. This query produces an error if jitter signal type is set to clock signal. Use the :MEASure:JITTer:DDJVsbit? query to return the DDJ values. Use the :MEASure:JITTer:PATTern? query to return the edge type values.

NOTE   This query returns data in the LSB (Least Significant Byte) first format. This format can affect the ability of your programs to correctly interpret the returned data as explained in "Definite-Length Block Response Data" on page 17.

Restrictions   Jitter mode. Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:EBITs] <value><NL>`

JITTer:EDGE

Command   `:MEASure:JITTer:EDGE {RISing|FALLing|ALL}`

Specifies which edge for which to display measurement results.

Restrictions   Jitter mode. Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Query   `:MEASure:JITTer:EDGE?`

This query returns the current edge setting for jitter mode measurements.

Returned Format	`[:TRIGger:] {RIS|FALL|ALL}<NL>`
Example	`:MEASure:JITTer:EDGE ALL`

JITTer:FREQuency:ANALysis

Command	`:MEASure:JITTer:FREQuency:ANALysis {ON | 1 | OFF | 0}`
	Turns jitter frequency analysis on (1) and off (0). If the instrument is not already in Jitter Mode (with Option 200 installed), a "Settings Conflict" error is generated by this command. After sending this command, allow approximately five seconds before sending any other analysis related MEASure:JITTer:FREQuency commands. This ensures that any measurement data will be valid.
Restrictions	Jitter mode. 86100D or 86100C (Software revision A.04.10 and above). Option 200, Enhanced Jitter Analysis Software.
Query	`:MEASure:JITTer:FREQuency:ANALysis?`
	This query returns the current state of jitter frequency analysis.
Returned Format	`[:MEASure:JITTer:FREQuency:ANALysis] {1 | 0}<NL>`
Example	`10 OUTPUT 707;":MEASURE:JITTER:FREQUENCY:ANALYSIS ON"`

JITTer:FREQuency:COMPonents?

Query	`:MEASure:JITTer:FREQuency:COMPonents?`
	Returns a comma-separated list (as a string) of the detected frequency components. For each component, the format is magnitude, frequency, subrate. Subrate is either the string "rate/N" where N is the subrate number, or "-----" for asynchronous components. Both the magnitude and frequency values have units appended to them. Set the instrument in single sweep mode or send the DIGitize root-level command before sending this query to ensure valid measurement data exists.
Restrictions	Jitter mode. 86100D or 86100C (Software revision A.04.10 and above). Option 200, Enhanced Jitter Analysis Software.
Returned Format	`[:MEASure:JITTer:FREQuency:COMPonents] <string><NL>`
	The following is an example of a returned string:
	`930 fs,78.37 MHz,rate/127,420 fs,622.1 MHz,rate/16,210 fs,1.244 GHz,rate/8, 121 fs, 56.43 MHz,----"`
Example	`10 OUTPUT 707;":MEASURE:JITTER:FREQUENCY:COMPONENTS?"`

JITTer:FREQuency:MAXNumber

Command	`:MEASure:JITTer:FREQuency:MAXNumber <max_async_freqs>`
	Sets the maximum number of asynchronous frequency components that the instrument will detect. Detected components are analyzed in order of descending magnitude until the number of components specified with this command is obtained.
Restrictions	Jitter mode. 86100D or 86100C (Software revision A.04.10 and above). Option 200, Enhanced Jitter Analysis Software.
Query	`:MEASure:JITTer:FREQuency:MAXNumber?`
	This query returns the maximum number of components setting.
Returned Format	`[:MEASure:JITTer:FREQuency:MAXNumber] <max_async_freqs><NL>`

Example   `10 OUTPUT 707;":MEASURE:JITTER:FREQUENCY:MAXNUMBER 10"`

JITTer:FREQuency:SCAN

Command   `:MEASure:JITTer:FREQuency:SCAN`

Initiates a scan that calculates the absolute frequency of any significant asynchronous frequency components up to the maximum number of components specified with the MEASure:JITTer:FREQuency:MAXNumber command. If the instrument is not in Jitter Mode (with Option 200 installed), a "Settings Conflict" error is generated by this command.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.10 and above). Option 200, Enhanced Jitter Analysis Software.

Example   `10 OUTPUT 707;":MEASURE:JITTER:FREQUENCY:SCAN"`

JITTer:ISI?

Query   `:MEASure:JITTer:ISI?`

Returns the inter-symbol interference value measured on the current source.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:ISI] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:JITTer:ISI?"`

JITTer:LEVel?

Query   `:MEASure:JITTer:LEVel?`

Returns the amplitude level at which jitter measurements are made.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:LEVel] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:JITTer:LEVel?"`

JITTer:LEVel:DEFine

Command   `:MEASure:JITTer:LEVel:DEFine {PERCent,<percentage_value> | UNITs,<level_value> | AVERage}`

Defines the jitter sampling level. It may be specified as a percentage in the range of 30% to 70%, as an absolute amplitude level, or as the average amplitude of the test signal. If you specify UNITs, the level value is interpreted as Watts or Volts depending on the type of input channel selected: optical or electrical. For example, if a value of 5.00E-3 is entered, it will be interpreted as 5 mW when applied to an optical channel and 5 mV when applied to an electrical channel.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Query   `:MEASure:JITTer:LEVel:DEFine?`

Returned Format   `[:MEASure:JITTer:LEVel:DEFine] <current_setting><NL>`

Example   `:MEASure:JITTer:LEVel:DEFine PERCent,40`

JITTer:PATTern?

Query   `:MEASure:JITTer:PATTern?`

Returns definite-length block data. The data block contains the pattern as determined by the instrument. Each value in the pattern is a single byte. Values in the pattern are the ASCII values for '0' and '1' (30 hex and 31 hex, respectively). The data block is followed by a terminator character, 0A hex (linefeed). This query will return an incomplete description of the pattern if all of the data needed to determine the pattern has not yet been acquired. This query produces an error if jitter signal type is set to clock signal. Use the :MEASure:JITTer:DDJVsbit? query to return the DDJ values. Use the :MEASure:JITTer:EBITs? query to return the bit numbers.

NOTE   This query returns data in the LSB (Least Significant Byte) first format. This format can affect the ability of your programs to correctly interpret the returned data as explained in "Definite-Length Block Response Data" on page 17.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above). When writing new code for software revision A.07.00 and above, use the recommended command "SINTegrity:PATTern?" on page 277.

Returned Format   `[:MEASure:JITTer:PATTern] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:JITTer:PATTern?"`

JITTer:PJ?

Query   `:MEASure:JITTer:PJ?`

Returns the periodic jitter, PJ (δ-δ), value measured on the current source.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:PJ] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:JITTer:PJ?"`

JITTer:PJRMs?

Query   `:MEASure:JITTer:PJRMs?`

Returns the periodic jitter value, RJ (rms), measured on the current source.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:PJRMs] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASure:JITTer:PJRMs?"`

JITTer:RJ?

Query   `:MEASure:JITTer:RJ?`

Returns the random jitter value measured on the current source.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format	`[:MEASure:JITTer:RJ] <value><NL>`
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"`
	`20 OUTPUT 707;":MEASURE:JITTER:RJ?"`

JITTer:RJSTablize

| Command | `:MEASure:JITTer:RJSTablize {{OFF | 0} | {ON | 1}}` |

Turns RJ stabilization on or off. RJ Stabilization locks the value of the measured RJ. Use RJ stabilization to prevent any uncorrelated non-Gaussian, non-periodic jitter from falsely contributing to any measured RJ value. This requires a two-part measurement. First, remove any sources of uncorrelated non-Gaussian, non-periodic jitter (for example, crosstalk or non-periodic electromagnetic interference), set RJ stabilization off and measure the RJ. Then, turn RJ stabilization on and reapply the sources of uncorrelated non-Gaussian, non-periodic jitter. One use of RJ stabilization is to prevent crosstalk, from an adjacent channel, appearing as jitter. Use the MEASure:JITTer:RJSValue command to set or query the stabilization value.

Restrictions	Jitter mode. 86100D or 86100C (Software revision A.04.20 and above).
Query	`:MEASure:JITTer:RJSTablize?`
Returned Format	`[:MEASure:JITTer:RJSTablize] {{OFF	0}	{ON	1}}<NL>`
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"`
	`20 OUTPUT 707;":MEASURE:JITTER:RJSTABLIZE ON"`

JITTer:RJSValue

| Command | `:MEASure:JITTer:RJSValue <RJ_set_num>` |

Sets the RJ stabilization value. Use the MEASure:JITTer:RJSTablize command to turn RJ stabilization on or off.

Restrictions	Jitter mode. 86100D or 86100C (Software revision A.04.20 and above).
Query	`:MEASure:JITTer:RJSValue?`
Returned Format	`[:MEASure:JITTer:RJSValue] <RJ_set_num><NL>`
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"`
	`20 OUTPUT 707;":MEASURE:JITTER:RJSVALUE 6E-12"`

JITTer:SIGNal

| Command | `:MEASure:JITTer:SIGNal {CLOCk|DATA}` |

Specifies the type of signal being measured.

| Restrictions | Jitter mode. 86100D or 86100C (Software revision A.04.00 and above). When writing new code for software revision A.07.00 and above, use the recommended command "SINTegrity:SIGNal" on page 278. |
| Query | `:MEASure:JITTer:SIGNal?` |

This query returns the current setting for the signal type.

| Returned Format | `[:MEASure:JITTer:SIGNal] {CLOCk|DATA}<NL>` |
| Example | `:MEASURE:JITTER:SIGNAL DATA` |

JITTer:SIGNal:AUTodetect

Command	`:MEASure:JITTer:SIGNal:AUTodetect {ON|OFF}`

Turns automatic detection of the signal type (clock or data) on or off. The automatic detection occurs during an autoscale.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above). When writing new code for software revision A.07.00 and above, use the recommended command "SINTegrity:SIGNal:AUTodetect" on page 278.

Query   `:MEASure:JITTer:SIGNal:AUTodetect?`

This query returns the current setting for automatic signal detection.

Returned Format   `[:MEASure:JITTer:SIGNal:AUTodetect] {ON|OFF}<NL>`

Example   `:MEASURE:JITTER:SIGNAL:AUTODETECT ON`

JITTer:TJ?

Query   `:MEASure:JITTer:TJ?`

Returns the total jitter value measured on the current source.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Returned Format   `[:MEASure:JITTer:TJ] <value><NL>`

Example   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:JITTER:TJ?"`

JITTer:TJ:DEFine

Command   `:MEASure:JITTer:TJ:DEFine <level_value>`

Sets the Bit Error Ratio (BER) at which total jitter is measured. The default value is 10^{-12}.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.10 and above). Option 200, Enhanced Jitter Analysis Software.

Query   `:MEASure:JITTer:TJ:DEFine?`

Returned Format   [:MEASure:JITTer:TJ:DEFine] <level_value><NL>

Example   `10 OUTPUT 707;":MEASure:JITTer:TJ?"`

JITTer:UNITs

Command   `:MEASure:JITTer:UNITs {SECond|UINTerval}`

Sets the units used for jitter mode measurements, seconds or unit interval.

Restrictions   Jitter mode. 86100D or 86100C (Software revision A.04.00 and above).

Query   `:MEASure:JITTer:UNITs?`

This query returns the current setting for jitter mode measurement units.

Returned Format   `[:MEASure:JITTer:UNITs] {SEC|UINT}<NL>`

Example	`:MEASure:JITTer:UNITs SEC`

MATLab

Command	`:MEASure:MATLab<N>{?	[<source>]}`
	Installs and runs an assigned MATLAB measurement script or queries the results of a script. Specify the script in the command syntax by including 1, 2, 3, or 4 for <N>. To assing a script, refer to **"MATLab<N>:SCRipt"** on page 270.
Restrictions	86100D or 86100C (Software revision A.08.00 and above). Option 201, Advanced Waveform Analysis Software.
Returned Format	[:MEASure:MATLab<N>{?|[<source>]}<NL>
Example	`10 OUTPUT 707;":MEASure:MATLab3 <source>"`

MATLab<N>:SCRipt

Command	`:MEASure:MATLab<N>:SCRipt{?	"<filename>"}`
	Assigns a user-defined MATLAB script to one of four script measurements. Specify these locations in the command syntax by including 1, 2, 3, or 4 for <N>. From the front-panel, locate script measurement buttons on the MATLAB tab in Eye/Mask, Oscilloscope, or TDR/TDT modes. The query returns the name of an assigned script. If no script is assigned to the selected location (1, 2, 3, or 4) the query returns the string "none".
Restrictions	86100D or 86100C (Software revision A.08.00 and above). Option 201, Advanced Waveform Analysis Software.
Returned Format	[:MEASure:MATLab<N>:SCRipt] "<filename>"<NL>
Example	`10 OUTPUT 707;":MEASURE:MATLAB2:SCRIPT "my_TWDP.m""`

MATLab<N>:ETENable

Command	`:MEASure:MATLab<N>:ETENable{?	{ON	1	OFF	0 }}`
	Enables or disables the MATLAB text ouput for a measurement script. The query returns the text output setting for the specified script. The value <N> represents one of four scripts (1, 2, 3, or 4).
Restrictions	86100D or 86100C (Software revision A.08.00 and above). Option 201, Advanced Waveform Analysis Software.
Returned Format	[:MEASure:MATLab<N>:ETENable{?|{ON|1|OFF|0 }}<NL>
Example	`10 OUTPUT 707;":MEASURE:MATLAB2:ETENABLE "my_test.m""`

MATLab<N>:ETEXt?

Query	`:MEASure:MATLab<N>:ETEXt?`
	Queries the MATLAB text ouput for a measurement script. The value <N> represents one of four scripts (1, 2, 3, or 4).
Restrictions	86100D or 86100C (Software revision A.08.00 and above). Option 201, Advanced Waveform Analysis Software.

Returned Format	[:MEASure:MATLab<N>:ETEXt "<text>"<NL>
Example	10 OUTPUT 707;":MEASURE:MATLAB2:ETEXt?"

NWIDth

Command	:MEASure:NWIDth [{CHANnel<N> | FUNCtion<N> | WMEMory<N>}]

Measures the width of the first negative pulse on the screen using the mid-threshold levels of the waveform (50% levels with standard measurements selected). The source is specified with the MEASure:SOURce command or with the optional parameter following the NWIDth command. The algorithm is, if the first edge on screen is rising, then

nwidth = time at the second rising edge – time at the first falling edge

else,

nwidth = time at the first rising edge – time at the first falling edge.

<N> for channels is dependent on the type of plug-in and its location in the instrument. For functions: 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4.

Mode	Oscilloscope mode only
Query	:MEASure:NWIDth? [{CHANnel<N> | FUNCtion<N> | WMEMory<N>}]
Returned Format	[:MEASure:NWIDth] <width>[,<result_state>]<NL>

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example	10 OUTPUT 707;":SYSTEM:HEADER OFF" 20 OUTPUT 707;":MEASURE:NWIDTH?"

OMAMplitude

Command	:MEASure:OMAMplitude [{CHANnel<N> | FUNCtion<N> | WMEMory<N>}]

On NRZ optical signals, measures the difference (absolute value) between the optical power of a one pulse and the optical power of a zero pulse. Use this command on single-valued waveforms in Oscilloscope mode. Measurements can be made on square waves and any other PRBS pulse train as long as three edges are present on the display. If less than three edges are displayed, the message Edge? is shown on the display. To measure OMA, the average power level of the center 20% between the first two edges is determined as well as the average power level of the center 20% between the second two edges. If the measurement is performed on an electrical signal, the measurement units are reported in volts. <N> for channels, functions, and waveform memories is 1, 2, 3, or 4.

Restrictions	Oscilloscope mode. Software revision A.07.00 and above.
Query	:MEASure:OMAMplitude? [{CHANnel<N> | FUNCtion<N> | WMEMory<N>}]
Returned Format	[:MEASure:OMAMplitude] <ratio>[,<result_state>]<NL>

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example	10 OUTPUT 707;":SYSTEM:HEADER OFF" 20 OUTPUT 707;":MEASURE:OMAMPLITUDE?"

OVERshoot

Command	:MEASure:OVERshoot [{CHANnel<N> | FUNCtion<N> | WMEMory<N>}]

Measures the overshoot of the first edge on the screen. Sources are specified with the MEASure:SOURce command or with the optional parameter following the OVERshoot command. <N> for channels, functions, and waveform memories is 1, 2, 3, or 4.

The algorithm is:

If the first edge onscreen is rising, then

$$\text{overshoot} = \frac{\text{Local V}_{max} - V_{top}}{V_{amplitude}}$$

else

$$\text{overshoot} = \frac{\text{V}_{base} - \text{Local V}_{min}}{V_{amplitude}}$$

Mode	Oscilloscope mode only
Query	`:MEASure:OVERshoot? [{CHANnel<N>	FUNCtion<N>	WMEMory<N>}]`
Returned Format	`[:MEASure:OVERshoot] <ratio>[,<result_state>]<NL>`
	If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:OVERSHOOT?"`

PERiod

Command	`:MEASure:PERiod [{CHANnel<N>	FUNCtion<N>	WMEMory<N>}]`
	Measures the period of the first complete cycle on the screen using the mid-threshold levels of the waveform (50% levels with standard measurements selected). The source is specified with the MEASure:SOURce command or with the optional parameter following the PERiod command. <N> for channels, functions, and waveform memories is 1, 2, 3, or 4. The algorithm is:

If the first edge onscreen is rising then

 period = time at the second rising edge – time at the first rising edge

else

 period = time at the second falling edge – time at the first falling edge.

Mode	Oscilloscope mode only
Query	`:MEASure:PERiod? [{CHANnel<N>	FUNCtion<N>	WMEMory<N>}]`
Returned Format	`[:MEASure:PERiod] <period>[,<result_state>]<NL>`
	If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:PERIOD?"`

PWIDth

Command	`:MEASure:PWIDth [{CHANnel<N>	FUNCtion<N>	WMEMory<N>}]`

Measures the width of the first positive pulse on the screen using the mid-threshold levels of the waveform (50% levels with standard measurements selected). The source is specified with the MEASure:SOURce command or with the optional parameter following the PWIDth command. <N> for channels is dependent on the type of plug-in and its location in the instrument. For functions: 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4. The algorithm is:

If the first edge on screen is rising, then

pwidth = time at the first falling edge – time at the first rising edge

else,

pwidth = time at the second falling edge – time at the first rising edge

Mode	Oscilloscope mode only
Query	`:MEASure:PWIDth? [{CHANnel<N>	FUNCtion<N>	WMEMory<N>}]` Returns the measured pulse width in seconds.
Returned Format	`[:MEASure:PWIDth] <width>[,<result_state>]<NL>` If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:PWIDTH?"`

RESults?

Query	`:MEASure:RESults?` In Oscilloscope, Eye/Mask, and TDR/TDT modes, returns the current results of up to four measurements that are displayed in the results table. For each measurement, the result values, shown in Table 43, are returned in a comma separated string. The measurements are returned in the order displayed in the table from top to bottom. If SENDvalid is ON (refer to page 276), the <result_state> is also returned. If :MEASure:SENDvalid is off, any questionable restults are returned as 9.999E+37, except for the n-samples field. If :MEASure:SENDvalid is on, current values are returned for any questionable results.
Restrictions	Does not work with Jitter mode.
Returned Format	`[:MEASure:RESults] <result values><NL>`
Example	`20 OUTPUT 707;":MEASURE:RESULTS?"`

Table 42   Returned Results Values

Send valid OFF	Send valid ON
measurement name	measurement name
current result	current result
	result state (see Table 43)
minimum [*]	minimum [a]
maximum [a]	maximum [a]
mean [a]	mean [a]
standard deviation [a]	standard deviation [a]
n-samples [a]	n-samples [a]

Table 42   Returned Results Values (continued)

Send valid OFF	Send valid ON
Additional Fields with Limit Test On	
limit failures	limit failures
limit total tests	limit total tests
limit status	limit status

* This value is not returned in Jitter Mode. Instead, the measurement result
9.99999E+37 is returned.

Table 43   Result States (Sheet 1 of 2)

Code	Description
0	Result correct. No problem found.
1	Result questionable. Current questionable values are returned.
2	Result less than or equal to value returned.
3	Result greater than or equal to value returned.
4	Result returned is invalid.
5	Result invalid. Required edge not found.
6	Result invalid. Max not found.
7	Result invalid. Min not found.
8	Result invalid. Requested time not found.
9	Result invalid. Requested voltage not found.
10	Result invalid. Top and base are equal.
11	Result invalid. Measurement zone too small.
12	Result invalid. Lower threshold not on waveform.
13	Result invalid. Upper threshold not on waveform.
14	Result invalid. Upper and lower thresholds are too close.
15	Result invalid. Top not on waveform.
16	Result invalid. Base not on waveform.
17	Result invalid. Completion criteria not reached.
18	Result invalid. Measurement invalid for this type of signal.
19	Result invalid. Signal is not displayed.
20	Result invalid. Waveform is clipped high.
21	Result invalid. Waveform is clipped low.
22	Result invalid. Waveform is clipped high and low.
23	Result invalid. Data contains all holes.
24	Result invalid. No data on screen.
25	Result invalid. Cursor is not on screen.
26	Result invalid. Measurement aborted.
27	Result invalid. Measurement timed-out.
28	Result invalid. No measurement to track.
30	Result invalid. Eye pattern not found.
32	Result invalid. Dark level is invalid.
33	Result invalid. Color grade/gray scale database has more than one source.

Table 43   Result States (Sheet 2 of 2)

34	Result invalid. No RZ eye pattern found.
35	Result invalid. Excessive extinction ratio correction.
37	Result invalid. No TDR/TDT reference plane defined.

RISetime

Command	`:MEASure:RISetime [{CHANnel<N>	FUNCtion<N>	RESPonse<N>	WMEMory<N>	CGRade}]`

Measures the rise time of the first displayed edge by measuring the time at the lower threshold of the rising edge, measuring the time at the upper threshold of the rising edge, then calculating the rise time with the following algorithm:

Rise time = time at upper threshold point – time at lower threshold point.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the RISetime command. Where CHANnel<N>, FUNCtion<N>, RESPonse<N>, and WMEMory<N> apply in Oscilloscope and TDR modes only; CGRade in Eye mode only. <N> is for channels, functions, TDR responses and waveform memories: 1, 2, 3, or 4. With standard measurements selected, the lower threshold is at the 10% point and the upper threshold is at the 90% point on the rising edge.

Mode	All instrument modes.
Query	`:MEASure:RISetime? [{CHANnel<N>	FUNCtion<N>	RESPonse<N>	WMEMory<N>	CGRade}]`
Returned Format	`[:MEASure:RISetime] <rise_time>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in .

Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":MEASURE:RISETIME?"`

SCRatch

Command	`:MEASure:SCRatch`

Clears the measurement results from the screen.

Example	This example clears the current measurement results from the screen. `10 OUTPUT 707;":MEASURE:SCRATCH"`

SENDvalid

Command	`:MEASure:SENDvalid {ON	OFF	1	0}`

Enables the result state code to be returned with the :MEASure:RESults? query and with individual measurements. If SENDvalid is off, any questionable restults are returned as 9.999E+37. If SENDvalid is on, current values are returned for any questionable results and an additional result telling you if the data is valid or not. A zero "0" indicates valid data, a "1" indicates the data is questionable.

Query	`:MEASure:SENDvalid?`

The query returns the state of the SENDvalid control.

Returned Format	`[:MEASure:SENDvalid] {0	1}<NL>`

Examples   `10 OUTPUT 707;":MEASURE:CGR:JITT? RMS`
`! Your code to read invalid result: -9.99999E+37`
`30 OUTPUT 707;":MEASURE:SENDvalid ON`
`40 OUTPUT 707;":MEASURE:CGR:JITT? RMS`
`! Your code to read questionable result: -1.17E-012,1`
`50 OUTPUT 707;":MEASURE:CGR:JITT? RMS`
`! Your code to read valid result: -1.20E-012,0`

See Also   Refer to the MEASure:RESults query for information on the results returned and how they are affected by the SENDvalid command. Refer to the individual measurements for information on how the result state is returned.

SINTegrity:BERFloor?

Query   `MEASure:SINTegrity:BERFloor?`

Returns the bit error ratio (BER) floor measurement. This is the extrapolated BER at the center of the eye. If both amplitude and jitter analysis is active, it will take into account both the probability of timing errors (jitter) as well as the probability of amplitude errors (noise and interference).

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Returned Format   `[:MEASure:SINTegrity:BERFloor] <measurement><NL>`

Example   `10 OUTPUT 707;":MEASURE:SINTEGRITY:BERFLOOR?"`

SINTegrity:BERLimit?

Query   `MEASure:SINTegrity:BERLimit?`

Returns JITTer if the signal bit error ratio is primarily due to jitter, AMPLitude if the signal bit error ratio is primarily due to noise and interference, or BALanced if the two are evenly contributing to BER. The value 9.999E37 is returned, if both BER Floors are $\leq 1\text{x }10^{-18}$. This command is only available if both jitter analysis and amplitude analysis are turned on.

Restrictions   86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.

Returned Format   `[:MEASure:SINTegrity:BERLimit] {JITTer | AMPLitude | BALanced}<NL>`

Example   `10 OUTPUT 707;":MEASURE:SINTEGRITY:BERLIMIT?"`

SINTegrity:PATTern?

Query   `MEASure:SINTegrity:PATTern?`

Returns definite-length block data. The data block contains the pattern as determined by the instrument. Each value in the pattern is a single byte. Values in the pattern are the ASCII values for '0' and '1' (30 hex and 31 hex, respectively). The data block is followed by a terminator character, 0A hex (linefeed). This query will return an incomplete description of the pattern if all of the data needed to determine the pattern has not yet been acquired. This query produces an error if signal type is set to clock signal. This replaces the obsolete command "JITTer:PATTern?" on page 267

This query returns data in the LSB (Least Significant Byte) first format. This format can affect the ability of your programs to correctly interpret the returned data as explained in "Definite-Length Block Response Data" on page 17.

Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Returned Format	`[:MEASure:SINTegrity:PATTern] <value><NL>`
Example	`10 OUTPUT 707;":MEASURE:SINTEGRITY:PATTERN?"`

SINTegrity:SIGNal

Command	`MEASure:SINTegrity:SIGNal {CLOCk	DATA}`
	Specifies the type of signal being measured in Jitter and Signal Integrity mode. It replaces the obsolete command :MEASure:JITTer:SIGNal.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Query	`MEASure:SINTegrity:SIGNal?`
Returned Format	`[:MEASure:SINTegrity:SIGNal] {CLOCk	DATA}<NL>`
Example	`10 OUTPUT 707;":MEASURE:SINTEGRITY:SIGNAL CLOCK"`

SINTegrity:SIGNal:AUTodetect

Command	`MEASure:SINTegrity:SIGNal:AUTodetect {ON	1	OFF	0}`
	Turns automatic detection of the signal type (clock or data) on or off. The automatic detection happens during an autoscale. This command replaces the obsolete command :MEASure:JITTer:SIGNal:AUTodetect.
Restrictions	86100D or 86100C (software revision A.07.00 and above). Jitter Mode including Advanced Amplitude Analysis/RIN/Q-Factor application.
Query	`MEASure:SINTegrity:SIGNal:AUTodetect?`
Returned Format	`[:MEASure:SINTegrity:SIGNal:AUTodetect] {1	0}<NL>`
Example	`10 OUTPUT 707;":MEASURE:SINTEGRITY:SIGNAL:AUTODETECT ON"`

SOURce

Command	`:MEASure:SOURce <source>[,<source>]`
	Selects the source for measurements. You can specify one or two sources with this command. All measurements except MEASure: DEFine:DELTatime are made on the first specified source. The delta time measurement uses two sources if two are specified. If only one source is specified, the delta time measurement uses that source for both of its parameters. The source is always color grade/gray scale data in eye mode, except for average optical power and histogram measurements. This is a global definition. It is used for all subsequent remote measurements unless a different source is specified with the optional source parameter in the measure command. <source> is {CHANnel<N>	FUNCtion<N>	RESPonse<N>	WMEMory<N>}. <N>, for channels, functions, TDR responses and waveform memories, is 1, 2, 3, or 4.
Mode	Oscilloscope and TDR modes. Eye mode uses this for average optical power measurements.
Query	`:MEASure:SOURce?`
Returned Format	`[:MEASure:SOURce] <source>[,<source>]<NL>`

Example   `10 OUTPUT 707;":MEASURE:SOURCE CHANNEL1"`

TEDGe?

Query   `:MEASure:TEDGe? <meas_thres_txt>,<slope><occurrence> [,<source>]`

Returns the time interval between the trigger event and the specified edge (threshold level, slope, and transition) in oscilloscope mode. The query will return the time interval between the reference plane and the specified edge in TDR mode. <meas_thres_txt> is defined as UPPer, MIDDle, or LOWer to identify the threshold. <slope> is { – (minus) for falling | + (plus) for rising | <none> (the slope is optional; if no slope is specified, + (plus) is assumed) }. <occurrence> is a numeric value representing the edge of the occurrence. The desired edge must be present on the display. Edges are counted with 1 being the first edge from the left on the display. <source> is {CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>} with <N>, for channels, functions, TDR responses, and waveform memories, equal to 1, 2, 3, or 4.

NOTE   TEDGe is measured for a value less than or equal to 20. A value greater than 20 returns data out of range.

Mode   Oscilloscope and TDR modes.

Returned Format   `[:MEASure:TEDGe] <time>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   This example returns the time interval between the trigger event and the 90% threshold on the second rising edge of the source waveform to the numeric variable, Time.

```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":MEASURE:TEDGE? UPPER,+2"
30 ENTER 707;Time
```

NOTE   When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TDR:AVERage

Command   `:MEASure:TDR:AVERage {CHANnel<N> | RESPonse<N>}`

Measures the average TDR impedance (Y-axis value) for the selected channel or response. Because the measurement is taken from data across the entire screen, display only data that you want included in the measurement. For example, do *not* display data before (to the left of) the reference plane.

Restrictions   TDR mode. 86100D or 86100C (software revision A.05.00 and above).

Query   `:MEASure:TDR:AVERage? {CHANnel<N> | RESPonse<N>}`

Returned Format   `[:MEASure:TDR:AVERage] <voltage> [,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example   ```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:TDR:AVERAGE? RESP1"
```

## TDR:MAX

| | |
|---|---|
| **Command** | `:MEASure:TDR:MAX {CHANnel<N> | RESPonse<N>}` |

Measures the maximum TDR impedance (Y-axis value) for the selected channel or response. Because the measurement is taken from data across the entire screen, display only data that you want included in the measurement. For example, do *not* display data before (to the left of) the reference plane. When used as a query, the returned value uses the same units as the setting for the selected channel or response. For example, if the channel units are set to volts, this query returns a value in volts.

| | |
|---|---|
| **Restrictions** | TDR mode. 86100D or 86100C (software revision A.05.00 and above). |
| **Query** | `:MEASure:TDR:MAX? {CHANnel<N> | RESPonse<N>}` |
| **Returned Format** | `[:MEASure:TDR:MAX {CHANnel<N> | RESPonse<N>}] <value><NL>` |
| **Example** | `10 OUTPUT 707;":MEASure:TDR:MAX RESPONSE1"` |

## TDR:MIN

| | |
|---|---|
| **Command** | `:MEASure:TDR:MIN {CHANnel<N> | RESPonse<N>}` |

Measures the minimum TDR impedance (Y-axis value) for the selected channel or response. Because the measurement is taken from data across the entire screen, display only data that you want included in the measurement. For example, do *not* display data before (to the left of) the reference plane. When used as a query, the returned value uses the same units as the setting for the selected channel or response. For example, if the channel units are set to volts, this query returns a value in volts.

| | |
|---|---|
| **Restrictions** | TDR mode. 86100D or 86100C (software revision A.05.00 and above). |
| **Query** | `:MEASure:TDR:MIN? {CHANnel<N> | RESPonse<N>}` |
| **Returned Format** | `[:MEASure:TDR:MIN {CHANnel<N> | RESPonse<N>}] <value><NL>` |
| **Example** | `10 OUTPUT 707;":MEASure:TDR:MIN RESPONSE1"` |

## TMAX

| | |
|---|---|
| **Command** | `:MEASure:TMAX [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |

Measures the first time at which the first maximum voltage of the source waveform occurred. The source is specified with the MEASure:SOURce command or with the optional parameter following the TMAX command. In TDR mode, the time reported is measured with respect to the reference plane. <N> is an integer, from 1 through 4.

| | |
|---|---|
| **Mode** | Oscilloscope and TDR modes. |
| **Query** | `:MEASure:TMAX? [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |

The query returns the time at which the first maximum voltage occurred.

| | |
|---|---|
| **Returned Format** | `[:MEASure:TMAX] <time>[,<result_state>]<NL>` |

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275. When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:TMAX?"
```

## TMIN

Command    `:MEASure:TMIN [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]`

Measures the first time at which the first minimum voltage of the source waveform occurred. The source is specified with the MEASure:SOURce command or with the optional parameter following the TMIN command. In TDR mode, the time reported is measured with respect to the reference plane. <N> is an integer, from 1 through 4.

Mode    Oscilloscope and TDR modes.

Query    `:MEASure:TMIN? [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]`

The query returns the time at which the first minimum voltage occurred.

Returned Format    `[:MEASure:TMIN] <time>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275. When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:TMIN?"
```

## TVOLt?

Query    `:MEASure:TVOLt? <voltage>,<slope><occurrence>[,{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]`

Returns the time interval between the trigger event and the specified voltage level and transition (oscilloscope mode) or the time interval between the reference plane and the specified voltage level and transition (TDR mode). The source is specified with the MEASure:SOURce command or with the optional parameter following the TVOLt? query. <voltage> is the voltage level at which time will be measured. <slope> is the direction of the waveform change when the specified voltage is crossed, rising (+) or falling (–). <occurrence> is the number of the crossing to be reported. If one, the first crossing is reported; if two, the second crossing is reported, and so on. <N> is an integer, from 1 through 4.

Mode    Oscilloscope and TDR modes.

Returned Format    `[:MEASure:TVOLt] <time>[,<result_state>]<NL>`

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275. When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

Example    The following example returns the time interval between the trigger event and the transition through –.250 Volts on the third rising edge of the source waveform to the numeric variable, Time.
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":MEASURE:TVOLT? -.250,+3"
```

## VAMPlitude

Command    `:MEASure:VAMPlitude [{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>}]`

Calculates the difference between the top and base voltage of the specified source. Sources are specified with the MEASure:SOURce command or with the optional parameter following the VAMPlitude command. <N> is 1, 2, 3, or 4.

| | |
|---|---|
| **Mode** | Oscilloscope and TDR modes. |
| **Query** | `:MEASure:VAMPlitude? [{CHANnel<N> | FUNCtion<N> | RESPonse<N> | WMEMory<N>}]` |
| | The query returns the calculated difference between the top and base voltage of the specified source. |
| **Returned Format** | `[:MEASure:VAMPlitude] <amplitude>[,<result_state>]<NL>` |
| | If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275. |
| **Example** | `10 OUTPUT 707;":SYSTEM:HEADER OFF"`<br>`20 OUTPUT 707;":MEASURE:VAMPLITUDE?2"` |

## VAVerage

| | |
|---|---|
| **Command** | `:MEASure:VAVerage [{CYCLe | DISPlay} [,{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]]` |
| | Calculates the average voltage over the displayed waveform. The source is specified with the MEASure:SOURce command or with the optional parameter following the VAVerage command. The CYCLe parameter specifies to measure the average voltage across the first period of the display. This option is valid in oscilloscope mode only. The DISPlay parameter specifies to measure all the data on the display. This option is valid in both oscilloscope and TDR modes. <N> is an integer, from 1 through 4. |
| **Mode** | Oscilloscope and TDR (DISPlay option only) modes. |
| **Query** | `:MEASure:VAVerage? [{CYCLe | DISPlay}, [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]]` |
| | The query returns the calculated average voltage of the specified source. |
| **Returned Format** | `[:MEASure:VAVerage] <voltage> [,<result_state>]<NL>` |
| | If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275. |
| **Example** | `10 OUTPUT 707;":SYSTEM:HEADER OFF"`<br>`20 OUTPUT 707;":MEASURE:VAVERAGE? DISPLAY"` |

## VBASe

| | |
|---|---|
| **Command** | `:MEASure:VBASe [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |
| | Measures the statistical base of the waveform. The source is specified with the MEASure:SOURce command or with the optional parameter following the VBASe command. <N>, for channels, is dependent on the type of plug-in and its location in the instrument. For functions <N> is 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4. For TDR responses: 1, 2, 3, or 4. |
| **Mode** | Oscilloscope and TDR modes. |
| **Query** | `:MEASure:VBASe? [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |
| | The query returns the measured voltage value at the base of the specified source. |
| **Returned Format** | `[:MEASure:VBASe] <voltage>[,<result_state>]<NL>` |
| | If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275. |
| **Example** | `10 OUTPUT 707;":SYSTEM:HEADER OFF"`<br>`20 OUTPUT 707;":MEASURE:VBASE?"` |

## VMAX

| | |
|---|---|
| **Command** | `:MEASure:VMAX [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |

Measures the absolute maximum voltage present on the selected source waveform. The source is specified with the MEASure:SOURce command or with the optional parameter following the VMAX command. <N>, for channels, is dependent on the type of plug-in and its location in the instrument. For functions: 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4. For TDR responses: 1, 2, 3, or 4.

| | |
|---|---|
| **Mode** | Oscilloscope and TDR modes. |
| **Query** | `:MEASure:VMAX? [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |
| **Returned Format** | `[:MEASure:VMAX] <voltage>[,<result_state>]<NL>` |

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

| | |
|---|---|
| **Example** | `10 OUTPUT 707;":SYSTEM:HEADER OFF"`<br>`20 OUTPUT 707;":MEASURE:VMAX?"` |

## VMIN

| | |
|---|---|
| **Command** | `:MEASure:VMIN [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |

Measures the absolute minimum voltage present on the selected source waveform. The source is specified with the MEASure:SOURce command or with the optional parameter following the VMIN command. <N>, for channels, is dependent on the type of plug-in and its location in the instrument. For functions: 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4. For TDR responses: 1, 2, 3, or 4.

| | |
|---|---|
| **Mode** | Oscilloscope and TDR modes. |
| **Query** | `:MEASure:VMIN? [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |
| **Returned Format** | `[:MEASure:VMIN] <voltage>[,<result_state>]<NL>` |

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

| | |
|---|---|
| **Example** | `10 OUTPUT 707;":SYSTEM:HEADER OFF"`<br>`20 OUTPUT 707;":MEASURE:VMIN?"` |

## VPP

| | |
|---|---|
| **Command** | `:MEASure:VPP [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |

Measures the maximum and minimum voltages on the selected source, then calculates the peak-to-peak voltage as the difference between the two voltages. Sources are specified with the MEASure:SOURce command or with the optional parameter following the VPP command. <N> is an integer, from 1 through 4.

| | |
|---|---|
| **Mode** | Oscilloscope and TDR modes only |
| **Query** | `:MEASure:VPP? [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]` |
| **Returned Format** | `[:MEASure:VPP] <voltage>[,<result_state>]<NL>` |

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example        10 OUTPUT 707;":SYSTEM:HEADER OFF"
               20 OUTPUT 707;":MEASURE:VPP?"

## VRMS

Command        :MEASure:VRMS {CYCLe | DISPlay}, {AC | DC} [,{CHANnel<N> | FUNCtion<N> | WMEMory<N>}]

Measures the RMS voltage of the selected waveform by subtracting the average value of the waveform from each data point on the display. Sources are specified with the MEASure:SOURce command or with the optional parameter following the VRMS command. <N> is 1, 2, 3, or 4.

The CYCLe parameter instructs the RMS measurement to measure the RMS voltage across the first period of the display. The DISPLay parameter instructs the RMS measurement to measure all the data on the display. Generally, RMS voltage is measured across one waveform or cycle, however, measuring multiple cycles may be accomplished with the DISPLay option. The DISPLay parameter is also useful when measuring noise. The AC parameter is used to measure the RMS voltage subtracting out the DC component. The DC parameter is used to measure RMS voltage including the DC component. The AC RMS, DC RMS, and VAVG parameters are related as in the following formula:

$$\mathbf{DCVRMS}^2 = \mathbf{ACVRMS}^2 + \mathbf{VAVG}^2$$

Mode           Oscilloscope mode only.

Query          :MEASure:VRMS? {CYCLe | DISplay}, {AC | DC} [,{CHANnel<N> | FUNCtion<N> | WMEMory<N>}]

Returned Format   [:MEASure:VRMS] <voltage>[,<result_state>]<NL>

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example        10 OUTPUT 707;":SYSTEM:HEADER OFF"
               20 OUTPUT 707;":MEASURE:VRMS? CYCLE,AC"

## VTIMe?

Query          :MEASure:VTIMe? <time> [,{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]

Returns the measured voltage. <time> is the time interval between the trigger event and the specified edge (oscilloscope mode) or the time interval between the reference plane and the specified edge in TDR mode. <N> is an integer, from 1 to 4.

Mode           Oscilloscope and TDR modes.

Returned Format   [:MEASure:VTIMe] <voltage>[,<result_state>]<NL>

If SENDvalid is ON, <result_state> is also returned, as defined in Table 43 on page 275.

Example        10 OUTPUT 707;":SYSTEM:HEADER OFF"
               20 OUTPUT 707;":MEASURE:VTIME? 500E-3"

## VTOP

Command        :MEASure:VTOP [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]

Measures the statistical top of the selected source waveform. The source is specified with the MEASure:SOURce command or with the optional parameter following the VTOP command. <N>, for channels, is dependent on the type of plug-in and its location in the instrument. For functions: 1 or 2. For waveform memories (WMEMORY): 1, 2, 3, or 4. For TDR responses: 1, 2, 3, or 4.

Mode           Oscilloscope and TDR modes.

**Query**    `:MEASure:VTOP? [{CHANnel<N> | FUNCtion<N> | WMEMory<N> | RESPonse<N>}]`

**Returned Format**    `[:MEASure:VTOP] <voltage>[,<result_state>]<NL>`

If SENDvalid is ON, the <result_state> is also returned, as defined in Table 43 on page 275.

**Example**    `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":MEASURE:VTOP?"`

# 19 S-Parameter Commands (Rev. A.08.00 and Above)

This subsystem provides support for the S-parameter features provided with instrument revision A.08.00 and above. These features are part of Option 202, Enhanced Impedance and S-Parameter software. The S-parameter graphs display the S-parameters that have been transformed from the TDR/TDT time domain data to the frequency domain.

**KEYSIGHT**
TECHNOLOGIES

# Introduction

Software revision A.08.00 introduced two new S-parameter graphs (phase and group delay) in addition to the original S-parameter magnitude graph. If you are designing programs for instruments with software revision A.07.00 and below, refer to the commands documented in **Chapter 20**, "This subsystem provides support for the S-parameter features provided with instrument revision A.07.00 and below. If you are programming an instrument with software revision A.08.00 and above, refer to Chapter 19, "This subsystem provides support for the S-parameter features provided with instrument revision A.08.00 and above. These features are part of Option 202, Enhanced Impedance and S-Parameter software. The S-parameter graphs display the S-parameters that have been transformed from the TDR/TDT time domain data to the frequency domain.". S-parameter features are part of Option 202, Enhanced Impedance and S-Parameter software. The S-parameter graph displays the S-parameters that have been transformed from the TDR/TDT time domain data to the frequency domain.". On revision A.08.00 instruments, you can still use the commands documented in **Chapter 20**, however, the commands documented in this chapter are the preferred commands.

To turn S-parameter analysis on and off, use the TDRSparam command. To display the graphs, use the command **"SPARameter:GRAPh"** on page 184, **"SPARameter:LAYout"** on page 185, and **"SPARameter:SHADe"** on page 185. The original MAGGraph:HORizontal:STARt and MAGGraph:HORizontal:SPAN commands have been replaced with the GRAPh:HORizontal:STARt and GRAPh:HORizontal:SPAN commands; these new commands control the horizontal scaling of all three graphs. Use the :MAGGraph, :PGRaph, and :GDGRaph commands to control the vertical scaling of the magnitude, phase, and group-delay graphs, respectively. Use the :SPARameter:MARKer commands to place markers on the graphs. New queries have been added for the phase and group-delay graph markers (MARKer:PGRaph: and MARKer:GDGRaph:). S-parameter data (including phase information) can be saved to files using **"SPARameter:SAVE"** on page 171. The Fourier transform of the time-domain step response includes trace data starting from the reference plane. To save the S-parameter data in a touchstone file (without group-delay information), refer to the command **"TFILe?"** on page 174.

## Restrictions

The S-Parameter subsystem requires TDR mode with Option 202, Enhanced Impedance and S-Parameter software. Instrument software revision A.06.00 and above.

## Windowing

By adjusting the time span and reference plane position, you can use windowing as a time filtering technique to measure the frequency response at a specific location of a test device. Only the information in the window is transformed allowing you to isolate the physical interconnects of a device and view them individually in frequency domain. Adjusting the time scale (time-per-division) will impact the maximum frequency range and frequency resolution.

## Frequency Span

The maximum usable frequency span is always set for the current conditions when the graph is displayed. The frequency span is dependent upon the time span used and the points-per-waveform setting. The time span (acquisition interval) for the Fourier transform equals the time-per-division setting multiplied by the number of display graticules (divisions) that the trace occupies.

$$F_{maximum} = \frac{\text{points-per-waveform}}{2(\text{time/division})(\text{display divisions})}$$

Consider the situation where the reference plane is at or beyond the display's left edge. In this case, data from the entire ten display divisions is used. If the time scale is 10 ns/div and the points-per-waveform setting is 1024, the maximum frequency will be 5.1 GHz.

If you move the reference plane to the second display division to the right of the display's left edge, only data from eight display divisions is used. With the same 10 ns/div time scale and 1024 points-per-waveform setting, the maximum frequency will now be 6.4 GHz. As you can see from the equation, as the time span decreases, the frequency span increases.

## Frequency Span Between Points

The number of points displayed on the screen is a result of the Fast Fourier Transform. If the graph is drawn with too few points, you may want to increase the frequency resolution. Frequency resolution is defined by the following equation:

$$F_{resolution} = \frac{1}{(time/division)(display\ divisions)}$$

Select a time span (acquisition interval) that is appropriate for your frequency data. Because time and frequency are inversely related, decreased time spans result in increased frequency resolution (fewer frequency data points). For example, with a 200 ps-per-division time-per-division setting with data taken across the full 10 display divisions, the frequency resolution equals 500 MHz. For the most information about your test device, place the reference plane near the display's left edge and increase the time-per-division setting.

# Commands

## GDGRaph:VERTical:MAXimum

| | |
|---|---|
| **Command** | `:SPARameter:GDGRaph:VERTical:MAXimum <vertical_max>`<br>Sets the maximum group delay of the group delay graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Example** | `10 OUTPUT 707;":SPAR:GDGR:VERT:MAX 2.0E-7"` |
| **Query** | `:SPARameter:GDGRaph:VERTical:MAXimum?` |
| **Returned Format** | `[:SPARameter:GDGRaph:VERTical:MAXimum] <vertical_max><NL>` |

## GDGRaph:VERTical:MINimum

| | |
|---|---|
| **Command** | `:SPARameter:GDGRaph:VERTical:MINimum <vertical_min>`<br>Sets the minimum group delay of the group delay graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Example** | `10 OUTPUT 707;":SPAR:GDGR:VERT:MIN -2.0E-7"` |
| **Query** | `:SPARameter:GDGRaph:VERTical:MINimum?` |
| **Returned Format** | `[:SPARameter:GDGRaph:VERTical:MINimum] <vertical_min><NL>` |

## GDGRaph:MARKer:XDELta?

| | |
|---|---|
| **Query** | `:SPARameter:MARKer:GDGRaph:XDELta?`<br>Queries the frequency difference ($\Delta$) between the X1 and X2 markers on the group delay graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Returned Format** | `[:SPARameter:MARKer:GDGRaph:XDELta] <value><NL>` |

## GDGRaph:MARKer:Y1Position?

| | |
|---|---|
| **Query** | `:SPARameter:MARKer:GDGRaph:Y1Position?`<br>Queries the amplitude value (Y1) of the X1 marker on the group delay graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Returned Format** | `[:SPARameter:MARKer:GDGRaph:Y1Position] <value><NL>` |

## GDGRaph:MARKer:Y2Position?

| | |
|---|---|
| **Query** | `:SPARameter:MARKer:GDGRaph:Y2Position?`<br>Queries the amplitude value (Y2) of the X2 marker on the group delay graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |

| | |
|---|---|
| Returned Format | `[:SPARameter:MARKer:GDGRaph:Y2Position] <value><NL>` |

## GDGRaph:MARKer:YDELta?

| | |
|---|---|
| Query | `:SPARameter:MARKer:GDGRaph:YDELta?` |
| | Queries the amplitude difference (Δ) between the X1 and X2 markers (Y1 and Y2 positions) on the group delay graph. |
| Restrictions | 86100D or 86100C (software revision A.08.00 and above). |
| Returned Format | `[:SPARameter:MARKer:GDGRaph:YDELta] <value><NL>` |

## GRAPh:HORizontal:SPAN

| | |
|---|---|
| Command | `:SPARameter:GRAPh:HORizontal:SPAN <span_freq>` |
| | Sets the start frequency of the S-parameters magnitude and phase graphs. Depending on the span setting, the span may need to be reduced before the start frequency can be changed. |
| Restrictions | 86100D or 86100C (software revision A.08.00 and above). |
| Example | `10 OUTPUT 707;":SPAR:GRAP:HOR:SPAN 5.0E+9"` |
| Query | `:SPARameter:GRAPh:HORizontal:SPAN?` |
| Returned Format | `[:SPARameter:GRAPh:HORizontal:SPAN] <span_freq><NL>` |

## GRAPh:HORizontal:STARt

| | |
|---|---|
| Command | `:SPARameter:GRAPh:HORizontal:STARt <start_freq>` |
| | Sets the start frequency of the S-parameters magnitude and phase graphs. Depending on the span setting, the span may need to be reduced before the start frequency can be changed. |
| Restrictions | 86100D or 86100C (software revision A.08.00 and above). |
| Example | `10 OUTPUT 707;":SPAR:GRAP:HOR:STAR 10E+6"` |
| Query | `:SPARameter:GRAPh:HORizontal:START?` |
| Returned Format | `[:SPARameter:GRAPh:HORizontal:STARt] <start_freq><NL>` |

## MAGGraph:MARKer:XDELta?

| | |
|---|---|
| Query | `:SPARameter:MARKer:MAGGraph:XDELta?` |
| | Queries the frequency difference (Δ) between the X1 and X2 markers on the magnitude graph. This is the recommended replacement for the "MARKer:XDELta?" on page 301. |
| Restrictions | 86100D or 86100C (software revision A.08.00 and above). |
| Returned Format | `[:SPARameter:MARKer:MAGGraph:XDELta] <value><NL>` |

## MAGGraph:MARKer:Y1Position?

| | |
|---|---|
| Query | `:SPARameter:MARKer:MAGGraph:Y1Position?` |

Queries the amplitude value (Y1) of the X1 marker on the magnitude graph. This is the recommended replacement for the **"MARKer:Y1Position?"** on page 301.

**Restrictions**    86100D or 86100C (software revision A.08.00 and above).

**Returned Format**    `[:SPARameter:MARKer:MAGGraph:Y1Position] <value><NL>`

## MAGGraph:MARKer:Y2Position?

**Query**    `:SPARameter:MARKer:MAGGraph:Y2Position?`

Queries the amplitude value (Y2) of the X2 marker on the magnitude graph. This is the recommended replacement for the **"MARKer:Y2Position?"** on page 301.

**Restrictions**    86100D or 86100C (software revision A.08.00 and above).

**Returned Format**    `[:SPARameter:MARKer:MAGGraph:Y2Position] <value><NL>`

## MAGGraph:MARKer:YDELta?

**Query**    `:SPARameter:MARKer:MAGGraph:YDELta?`

Queries the amplitude difference ($\Delta$) between the X1 and X2 markers (Y1 and Y2 positions) on the magnitude graph. This is the recommended replacement for the **"MARKer:YDELta?"** on page 302.

**Restrictions**    86100D or 86100C (software revision A.08.00 and above).

**Returned Format**    `[:SPARameter:MARKer:MAGGraph:YDELta] <value><NL>`

## MAGGraph:VERTical:MAXimum

**Command**    `:SPARameter:MAGGraph:VERTical:MAXimum <vertical_max>`

Sets the maximum amplitude (dB) of the S-parameters graph.

**Example**    `10 OUTPUT 707;":SPAR:MAGG:VERT:MAX 5"`

**Query**    `:SPARameter:MAGGraph:VERTical:MAXimum?`

**Returned Format**    `[:SPARameter:MAGGraph:VERTical:MAXimum] <vertical_max><NL>`

## MAGGraph:VERTical:MINimum

**Command**    `:SPARameter:MAGGraph:VERTical:MINimum <vertical_min>`

Sets the minimum amplitude (dB) of the S-parameters graph.

**Example**    `10 OUTPUT 707;":SPAR:MAGG:VERT:MIN -30"`

**Query**    `:SPARameter:MAGGraph:VERTical:MINimum?`

**Returned Format**    `[:SPARameter:MAGGraph:VERTical:MINimum] <vertical_min><NL>`

## MARKer:X1Position

**Command**    `:SPARameter:MARKer:X1Position <X1_frequency>`

Sets the X1 marker position to data point that is nearest the specified frequency. After using this command, query the value to determine the actual frequency of the marker.

| Example | 10 OUTPUT 707;":SPAR:MARK:X1P 10E9" |
|---|---|
| Query | Reads the frequency position of the X1 marker.<br>:SPARameter:MARKer:X1Position? |
| Returned Format | [:SPARameter:MARKer:X1Position] <X1_frequency><NL> |

## MARKer:X2Position

| Command | :SPARameter:MARKer:X2Position <X2_frequency><br>Sets the X2 marker position to data point that is nearest the specified frequency. After using this command, query the value to determine the actual frequency of the marker. |
|---|---|
| Example | 10 OUTPUT 707;":SPAR:MARKer:X2Position 10E9" |
| Query | Reads the frequency position of the X2 marker.<br>:SPARameter:MARKer:X2Position? |
| Returned Format | [:SPARameter:MARKer:X2Position] <X2_frequency><NL> |

## MARKer:X1Source

| Command | :SPARameter:MARKer:X1Source {CHANnel<N> \| RESPonse<N> \| WMEMory<N> \| FUNCtion<N>}<br>Selects the source waveform of the X1 marker, if more than one waveform is displayed on the graph. |
|---|---|
| Example | 10 OUTPUT 707;":SPAR:MARK:X1S CHAN2" |
| Query | The query returns only the short form of the command. For example CHAN1, RESP1, WMEM1, or FUNC1.<br>The long form is not returned even if :SYSTem:LONGform is on.<br>:SPARameter:MARKer:X1S? |
| Returned Format | [:SPARameter:MARKer:X1Source] {CHAN<N> \| RESP<N> \| WMEM<N> \| FUNC<N>}<NL> |

## MARKer:X2Source

| Command | :SPARameter:MARKer:X2Source {CHANnel<N> \| RESPonse<N> \| WMEMory<N> \| FUNCtion<N>}<br>Selects the source waveform of the X2 marker, if more than one waveform is displayed on the graph. |
|---|---|
| Example | 10 OUTPUT 707;":SPAR:MARK:X2S CHAN1" |
| Query | The query returns only the short form of the command. For example CHAN1, RESP1, WMEM1, or FUNC1.<br>The long form is not returned even if :SYSTem:LONGform is on.<br>:SPARameter:MARKer:X2Source? |
| Returned Format | [:SPARameter:MARKer:X2Source] {CHAN<N> \| RESP<N> \| WMEM<N> \| FUNC<N>}<NL> |

## MARKer:X1STate

| Command | :SPARameter:MARKer:X1STate {ON \| 1 \| OFF \| 0}<br>Turn on and off the X1 marker. |
|---|---|
| Example | 10 OUTPUT 707;":SPAR:MARK:X1ST ON" |
| Query | :SPARameter:MARKer:X1STate? |
| Returned Format | [:SPARameter:MARKer:X1STate] {ON \| 1 \| OFF \| 0}<NL> |

## MARKer:X2STate

| | |
|---|---|
| **Command** | `:SPARameter:MARKer:X2STate {ON | 1 | OFF | 0}` |
| | Turn on and off the X2 marker. |
| **Example** | `10 OUTPUT 707;":SPAR:MARK:X2ST ON"` |
| **Query** | `:SPARameter:MARKer:X2STate?` |
| **Returned Format** | `[:SPARameter:MARKer:X2STate] {ON | 1 | OFF | 0}<NL>` |

## PGRaph:MARKer:XDELta?

| | |
|---|---|
| **Query** | `:SPARameter:MARKer:PGRaph:XDELta?` |
| | Queries the frequency difference ($\Delta$) between the X1 and X2 markers on the phase graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Returned Format** | `[:SPARameter:MARKer:PGRaph:XDELta] <value><NL>` |

## PGRaph:MARKer:Y1Position?

| | |
|---|---|
| **Query** | `:SPARameter:MARKer:PGRaph:Y1Position?` |
| | Queries the amplitude value (Y1) of the X1 marker on the phase graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Returned Format** | `[:SPARameter:MARKer:PGRaph:Y1Position] <value><NL>` |

## PGRaph:MARKer:Y2Position?

| | |
|---|---|
| **Query** | `:SPARameter:MARKer:PGRaph:Y2Position?` |
| | Queries the amplitude value (Y2) of the X2 marker on the phase graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Returned Format** | `[:SPARameter:MARKer:PGRaph:Y2Position] <value><NL>` |

## PGRaph:MARKer:YDELta?

| | |
|---|---|
| **Query** | `:SPARameter:MARKer:PGRaph:YDELta?` |
| | Queries the amplitude difference ($\Delta$) between the X1 and X2 markers (Y1 and Y2 positions) on the phase graph. |
| **Restrictions** | 86100D or 86100C (software revision A.08.00 and above). |
| **Returned Format** | `[:SPARameter:MARKer:PGRaph:YDELta] <value><NL>` |

## PGRaph:VERTical:MAXimum

| | |
|---|---|
| **Command** | `:SPARameter:PGRaph:VERTical:MAXimum <vertical_max>` |
| | Sets the maximum angle of the S-parameters phase graph. |

| | |
|---|---|
| Restrictions | 86100D or 86100C (software revision A.08.00 and above). |
| Example | 10 OUTPUT 707;":SPAR:PGR:VERT:MAX 10" |
| Query | :SPARameter:PGRaph:VERTical:MAXimum? |
| Returned Format | [:SPARameter:PGRaph:VERTical:MAXimum] <vertical_max><NL> |

## PGRaph:VERTical:MINimum

| | |
|---|---|
| Command | :SPARameter:PGRaph:VERTical:MINimum <vertical_min> |
| | Sets the minimum angle of the S-parameters phase graph. |
| Restrictions | 86100D or 86100C (software revision A.08.00 and above). |
| Example | 10 OUTPUT 707;":SPAR:PGR:VERT:MIN -10" |
| Query | :SPARameter:PGRaph:VERTical:MINimum? |
| Returned Format | [:SPARameter:PGRaph:VERTical:MINimum] <vertical_min><NL> |

## TDRSparam

| | |
|---|---|
| Command | :SPARameter:TDRSparam {ON | 1 | OFF | 0} |
| | Turns on and off the S-parameter measurements, which also displays or hides the S-parameter graph. Because the S-parameter calculations occure only when the graph shade is displayed, the graph must be displayed before S-parameter data can be saved to a file. Refer to "SPARameter:SAVE" on page 171. |
| Example | 10 OUTPUT 707;":SPAR:TDRS ON" |
| Query | :SPARameter:TDRSparam? |
| Returned Format | [:SPARameter:TDRSparam] {ON | 1 | OFF | 0}<NL> |

## VWINdow

| | |
|---|---|
| Command | :SPARameter:VWINdow {ON | 1 | OFF | 0} |
| | Turns on and off the display of the windowed time-domain region. This region highlights the the range of the TDR data that will be transformed to the frequency domain and displayed on the S-parameter graph. It is a visual aid for the user and does not alter the data range transformed. |
| Example | 10 OUTPUT 707;":SPAR:VWIN ON" |
| Query | :SPARameter:VWINdow? |
| Returned Format | [:SPARameter:VWINdow] {ON | 1 | OFF | 0}<NL> |

# 20 S-Parameter Commands (Rev. A.07.00 and Below)

This subsystem provides support for the S-parameter features provided with instrument revision A.07.00 and below. If you are programming an instrument with software revision A.08.00 and above, refer to Chapter 19, "This subsystem provides support for the S-parameter features provided with instrument revision A.08.00 and above. These features are part of Option 202, Enhanced Impedance and S-Parameter software. The S-parameter graphs display the S-parameters that have been transformed from the TDR/TDT time domain data to the frequency domain.". S-parameter features are part of Option 202, Enhanced Impedance and S-Parameter  software. The S-parameter graph displays the S-parameters that have been transformed from the TDR/TDT time domain data to the frequency domain.

To turn S-parameter analysis on and off, use "TDRSparam" on page 302. Use the :SPARameter:MAGGraph commands in this chapter to control the scaling of the S-parameters graph. Use the :SPARameter:MARKer commands to place markers on the graph. S-parameter data (including phase information) can be saved to files using "SPARameter:SAVE" on page 171. The Fourier transform of the time-domain step response includes trace data starting from the reference plane.

**KEYSIGHT**
TECHNOLOGIES

# Introduction

### Restrictions

The S-Parameter subsystem requires TDR mode with Option 202, Enhanced Impedance and S-Parameter  software. Instrument software revision A.06.00 and above.

### Windowing

By adjusting the time span and reference plane position, you can use windowing as a time filtering technique to measure the frequency response at a specific location of a test device. Only the information in the window is transformed allowing you to isolate the physical interconnects of a device and view them individually in frequency domain. Adjusting the time scale (time-per-division) will impact the maximum frequency range and frequency resolution.

### Frequency Span

The maximum usable frequency span is always set for the current conditions when the graph is displayed. The frequency span is dependent upon the time span used and the points-per-waveform setting. The time span (acquisition interval) for the Fourier transform equals the time-per-division setting multiplied by the number of display graticules (divisions) that the trace occupies.

$$F_{maximum} = \frac{\text{points-per-waveform}}{2(\text{time/division})(\text{display divisions})}$$

Consider the situation where the reference plane is at or beyond the display's left edge. In this case, data from the entire ten display divisions is used. If the time scale is 10 ns/div and the points-per-waveform setting is 1024, the maximum frequency will be 5.1 GHz.

If you move the reference plane to the second display division to the right of the display's left edge, only data from eight display divisions is used. With the same 10 ns/div time scale and 1024 points-per-waveform setting, the maximum frequency will now be 6.4 GHz. As you can see from the equation, as the time span decreases, the frequency span increases.

### Frequency Span Between Points

The number of points displayed on the screen is a result of the Fast Fourier Transform. If the graph is drawn with too few points, you may want to increase the frequency resolution. Frequency resolution is defined by the following equation:

$$F_{resolution} = \frac{1}{(\text{time/division})(\text{display divisions})}$$

Select a time span (acquisition interval) that is appropriate for your frequency data. Because time and frequency are inversely related, decreased time spans result in increased frequency resolution (fewer frequency data points). For example, with a 200 ps-per-division time-per-division setting with data taken across the full 10 display divisions, the frequency resolution equals 500 MHz. For the most information about your test device, place the reference plane near the display's left edge and increase the time-per-division setting.

# Commands

## MAGGraph:HORizontal:SPAN

**Command**  :SPARameter:MAGGraph:HORizontal:SPAN <span_freq>
Sets the frequency span of the S-parameters graph.

**Restrictions**  When writing new code for software revision A.08.00 and above, use the recommended command **"GRAPh:HORizontal:SPAN"** on page 291.

**Example**  10 OUTPUT 707;":SPAR:MAGG:HOR:SPAN 5.0E+9"

**Query**  :SPARameter:MAGGraph:HORizontal:SPAN?

**Returned Format**  [:SPARameter:MAGGraph:HORizontal:SPAN] <span_freq><NL>

## MAGGraph:HORizontal:STARt

**Command**  :SPARameter:MAGGraph:HORizontal:STARt <start_freq>
Sets the start frequency of the S-parameters graph. Depending on the span setting, the span may need to be reduced before the start frequency can be changed.

**Restrictions**  When writing new code for software revision A.08.00 and above, use the recommended command **"GRAPh:HORizontal:STARt"** on page 291.

**Example**  10 OUTPUT 707;":SPAR:MAGG:HOR:STAR 10E+6"

**Query**  :SPARameter:MAGGraph:HORizontal:START?

**Returned Format**  [:SPARameter:MAGGraph:HORizontal:STARt] <start_freq><NL>

## MAGGraph:VERTical:MAXimum

**Command**  :SPARameter:MAGGraph:VERTical:MAXimum <vertical_max>
Sets the maximum amplitude (dB) of the S-parameters graph.

**Example**  10 OUTPUT 707;":SPAR:MAGG:VERT:MAX 5"

**Query**  :SPARameter:MAGGraph:VERTical:MAXimum?

**Returned Format**  [:SPARameter:MAGGraph:VERTical:MAXimum] <vertical_max><NL>

## MAGGraph:VERTical:MINimum

**Command**  :SPARameter:MAGGraph:VERTical:MINimum <vertical_min>
Sets the minimum amplitude (dB) of the S-parameters graph.

**Example**  10 OUTPUT 707;":SPAR:MAGG:VERT:MIN -30"

**Query**  :SPARameter:MAGGraph:VERTical:MINimum?

**Returned Format**  [:SPARameter:MAGGraph:VERTical:MINimum] <vertical_min><NL>

## MARKer:X1STate

| | |
|---|---|
| **Command** | `:SPARameter:MARKer:X1STate {ON | 1 | OFF | 0}` |
| | Turn on and off the X1 marker. |
| **Example** | `10 OUTPUT 707;":SPAR:MARK:X1ST ON"` |
| **Query** | `:SPARameter:MARKer:X1STate?` |
| **Returned Format** | `[:SPARameter:MARKer:X1STate] {ON | 1 | OFF | 0}<NL>` |

## MARKer:X2STate

| | |
|---|---|
| **Command** | `:SPARameter:MARKer:X2STate {ON | 1 | OFF | 0}` |
| | Turn on and off the X2 marker. |
| **Example** | `10 OUTPUT 707;":SPAR:MARK:X2ST ON"` |
| **Query** | `:SPARameter:MARKer:X2STate?` |
| **Returned Format** | `[:SPARameter:MARKer:X2STate] {ON | 1 | OFF | 0}<NL>` |

## MARKer:X1Source

| | |
|---|---|
| **Command** | `:SPARameter:MARKer:X1Source {CHANnel<N> | RESPonse<N> | WMEMory<N> | FUNCtion<N>}` |
| | Selects the source waveform of the X1 marker, if more than one waveform is displayed on the graph. |
| **Example** | `10 OUTPUT 707;":SPAR:MARK:X1S CHAN2"` |
| **Query** | The query returns only the short form of the command. For example CHAN1, RESP1, WMEM1, or FUNC1. The long form is not returned even if :SYSTem:LONGform is on. `:SPARameter:MARKer:X1S?` |
| **Returned Format** | `[:SPARameter:MARKer:X1Source] {CHAN<N> | RESP<N> | WMEM<N> | FUNC<N>}<NL>` |

## MARKer:X2Source

| | |
|---|---|
| **Command** | `:SPARameter:MARKer:X2Source {CHANnel<N> | RESPonse<N> | WMEMory<N> | FUNCtion<N>}` |
| | Selects the source waveform of the X2 marker, if more than one waveform is displayed on the graph. |
| **Example** | `10 OUTPUT 707;":SPAR:MARK:X2S CHAN1"` |
| **Query** | The query returns only the short form of the command. For example CHAN1, RESP1, WMEM1, or FUNC1. The long form is not returned even if :SYSTem:LONGform is on. `:SPARameter:MARKer:X2Source?` |
| **Returned Format** | `[:SPARameter:MARKer:X2Source] {CHAN<N> | RESP<N> | WMEM<N> | FUNC<N>}<NL>` |

## MARKer:X1Position

| | |
|---|---|
| **Command** | `:SPARameter:MARKer:X1Position <X1_frequency>` |
| | Sets the X1 marker position to data point that is nearest the specified frequency. After using this command, query the value to determine the actual frequency of the marker. |
| **Example** | `10 OUTPUT 707;":SPAR:MARK:X1P 10E9"` |

| Query | Reads the frequency position of the X1 marker. |
|---|---|
| | `:SPARameter:MARKer:X1Position?` |

| Returned Format | `[:SPARameter:MARKer:X1Position] <X1_frequency><NL>` |
|---|---|

## MARKer:X2Position

| Command | `:SPARameter:MARKer:X2Position <X2_frequency>` |
|---|---|
| | Sets the X2 marker position to data point that is nearest the specified frequency. After using this command, query the value to determine the actual frequency of the marker. |

| Example | `10 OUTPUT 707;":SPAR:MARKer:X2Position 10E9"` |
|---|---|

| Query | Reads the frequency position of the X2 marker. |
|---|---|
| | `:SPARameter:MARKer:X2Position?` |

| Returned Format | `[:SPARameter:MARKer:X2Position] <X2_frequency><NL>` |
|---|---|

## MARKer:Y1Position?

| Command | `:SPARameter:MARKer:Y1Position?` |
|---|---|
| | Queries the amplitude value (Y1) of the X1 marker on an associated magnitude graph. |

| Restrictions | When writing new code for software revision A.08.00 and above, use the recommended command "MAGGraph:MARKer:Y1Position?" on page 291. |
|---|---|

| Query | `:SPARameter:MARKer:Y1Position?` |
|---|---|

| Returned Format | `[:SPARameter:MARKer:Y1Position] <value><NL>` |
|---|---|

## MARKer:Y2Position?

| Command | `:SPARameter:MARKer:Y2Position?` |
|---|---|
| | Queries the amplitude value (Y2) of the X2 marker on an associated magnitude graph. |

| Restrictions | When writing new code for software revision A.08.00 and above, use the recommended command "MAGGraph:MARKer:Y2Position?" on page 292. |
|---|---|

| Query | `:SPARameter:MARKer:Y2Position?` |
|---|---|

| Returned Format | `[:SPARameter:MARKer:Y2Position] <value><NL>` |
|---|---|

## MARKer:XDELta?

| Command | `:SPARameter:MARKer:XDELta?` |
|---|---|
| | Queries the frequency difference ($\Delta$) between the X1 and X2 markers on an associated magnitude graph. |

| Restrictions | When writing new code for software revision A.08.00 and above, use the recommended command "GDGRaph:MARKer:XDELta?" on page 290. |
|---|---|

| Query | `:SPARameter:MARKer:XDELta?` |
|---|---|

| Returned Format | `[:SPARameter:MARKer:XDELta] <value><NL>` |
|---|---|

## MARKer:YDELta?

| | |
|---|---|
| **Command** | `:SPARameter:MARKer:YDELta?` |

Queries the amplitude difference (Δ) between the X1 and X2 markers (Y1 and Y2 positions) on an associated magnitude graph.

**Restrictions**    When writing new code for software revision A.08.00 and above, use the recommended command "MAGGraph:MARKer:YDELta?" on page 292.

**Query**    `:SPARameter:MARKer:YDELta?`

**Returned Format**    `[:SPARameter:MARKer:YDELta] <value><NL>`

## TDRSparam

**Command**    `:SPARameter:TDRSparam {ON | 1 | OFF | 0}`

Turns on and off the S-parameter measurements, which also displays or hides the S-parameter graph. Because the S-parameter calculations occure only when the graph shade is displayed, the graph must be displayed before S-parameter data can be saved to a file. Refer to "SPARameter:SAVE" on page 171.

**Example**    `10 OUTPUT 707;":SPAR:TDRS ON"`

**Query**    `:SPARameter:TDRSparam?`

**Returned Format**    `[:SPARameter:TDRSparam] {ON | 1 | OFF | 0}<NL>`

## VWINdow

**Command**    `:SPARameter:VWINdow {ON | 1 | OFF | 0}`

Turns on and off the display of the windowed time-domain region. This region highlights the the range of the TDR data that will be transformed to the frequency domain and displayed on the S-parameter graph. It is a visual aid for the user and does not alter the data range transformed.

**Example**    `10 OUTPUT 707;":SPAR:VWIN ON"`

**Query**    `:SPARameter:VWINdow?`

**Returned Format**    `[:SPARameter:VWINdow] {ON | 1 | OFF | 0}<NL>`

# 21 Signal Processing Commands

The Signal Processing subsystem is used to control the signal processing applications. Refer to the instrument's online help for information on using these applications.

| NOTE | Instrument software revision A.04.10 and above (86100C/D instruments) with Option 201, Advanced Waveform Analysis Software, is required to run the Linear Feedforward Equalizer and MATLAB Filter applications. |
|------|------|

**KEYSIGHT**
TECHNOLOGIES

# Introduction

## General Application Commands

The following general commands are used for the active signal processing application.

```
SPRocessing:SOURce
SPRocessing:SOURce:DISPlay
SPRocessing:OUTPut
```

Linear Feedforward Equalizer Application Commands

The Linear Feedforward Equalizer application is controlled using the SPRocessing:LFEQualizer commands. Because the Linear Feedforward Equalizer uses single-valued waveforms, it requires pattern lock triggering in either Eye/Mask or Oscilloscope instrument modes. If you are modeling equalization to open a severely closed eye diagram, you may need to manually set pattern lock on the instrument.

MATLAB Filter Application Commands

The MATLAB Filter application is controlled using the SPRocessing:MATLab commands. MATLAB Filter works in Oscilloscope, Eye/Mask, or TDR/TDT modes. Use the SPRocessing:MATLab command to turn on and off this application. The MATLAB Filter application does not include MATLAB. So, you must purchase (www.mathworks.com) and install MATLAB separately on the instrument. If MATLAB is not already running on the instrument, when the MATLAB Filter application is started, MATLAB is automatically started and is minimized.

Because the MATLAB Filter uses single-valued waveforms, it requires pattern lock triggering in either Eye/Mask or Oscilloscope instrument modes. If you are creating a filter to open a severely closed eye diagram, you may need to manually set pattern lock on the instrument.

Commands

LFEQualizer

Command	`:SPRocessing:LFEQualizer {ON	1	OFF	0}`

Turns on and off the linear feedforward equalizer application. Pattern lock must be turned on prior to sending the LFEQualizer ON command.

Query	`:SPRocessing:LFEQualizer?`
Returned Format	`[:SPRocessing:LFEQualizer:] {0	1}<NL>`
Example	`10 OUTPUT 707;":SPROCESSING:LFEQUALIZER ON"`

LFEQualizer:BANDwidth

Command	`:SPRocessing:LFEQualizer:BANDwidth <bandwidth_setting>`

Sets or queries the bandwidth setting of the Linear Feedforward Equalizer application. Before sending this command, set the bandwidth mode to CUSTom using the LFEQualizer:BWMode command.

Query	`:SPRocessing:LFEQualizer:BANDwidth?`
Returned Format	`[:SPRocessing:LFEQualizer:BANDwidth] <bandwidth_setting><NL>`
Example	`10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:BWMODE CUSTOM"` `20 OUTPUT 707;":SPROCESSING:LFEQUALIZER:BANDWIDTH 12.5GHz"`

LFEQualizer:BWMode

Command	`:SPRocessing:LFEQualizer:BWMode {TSBandwidth	TTDelay	CUSTom}`

Sets or queries the bandwidth mode of the the Linear Feedforward Equalizer application. TSBandwidth selects tracking the source bandwidth. TTDelay selects tracking of the tap delay. CUSTom allows you to enter a bandwidth value using the LFEQualizer:BANDwidth command.

Query	`:SPRocessing:LFEQualizer:BWMode?`
Returned Format	`[:SPRocessing:LFEQualizer:BWMode] {TSBandwidth	TTDelay	CUSTom}<NL>`
Example	`10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:BWMODE "TTDelay""`

LFEQualizer:FDELay

Command	`:SPRocessing:LFEQualizer:FDELay <delay_setting>`

Sets or queries the filter delay setting of the Linear Feedforward Equalizer application. The filter delay sets the zero-time reference and is specified in tap delay increments. The delay value can be expressed to two significant digits between zero and one less than the number of taps. For example, if the design used three taps, the delay value can be between 0.00 and 2.00.

Restrictions	Software revision A.04.20 and above.
Query	`:SPRocessing:LFEQualizer:FDELay?`
Returned Format	`[:SPRocessing:LFEQualizer:FDELay] <delay_setting><NL>`

Example   `10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:FDELAY 1.50"`

LFEQualizer:NTAPs

Command   `:SPRocessing:LFEQualizer:NTAPs <number>`
Sets or queries the number of taps set for the Linear Feedforward Equalizer application.

Query   `:SPRocessing:LFEQualizer:NTAPs?`

Returned Format   `[:SPRocessing:LFEQualizer:NTAPs] <number><NL>`

Examples   `10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:NTAPS 4"`

LFEQualizer:TAP

Command   `:SPRocessing:LFEQualizer:TAP <tap_number>, <tap_value>`
Sets or queries the gain value for each tap for the Linear Feedforward Equalizer application. Use <tap_number> to specify tap. Use <tap_value> to specify the gain for the specified tap.

Query   `:SPRocessing:LFEQualizer:TAP? <tap_number>`

Returned Format   `[:SPRocessing:LFEQualizer:TAP] <tap_number>,<tap_value><NL>`

Example   `10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:TAP 3, 0.5"`

LFEQualizer:TAP:AUTomatic

Command   `:SPRocessing:LFEQualizer:TAP:AUTomatic`
Automatically open a closed eye diagram by determining the tap values for the displayed waveform. This function requires a PRBS pattern of length 2^5-1, 2^6-1, 2^7-1, 2^8-1, 2^9-1, 2^{10}-1, 2^{11}-1, 2^{12}-1, 2^{13}-1, 2^{14}-1, or 2^{15}-1. Inverted PRBS patterns are also supported.

Restrictions   Software revision A.04.20 and above.

Example   `10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:TAP:AUTOMATIC"`

LFEQualizer:TAP:NORMalize

Command   `:SPRocessing:LFEQualizer:TAP:NORMalize`
Normalizes the tap values for unity gain (0 dB) in the Linear Feedforward Equalizer application. The relative value of each tap is maintained.

Example   `10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:TAP:NORMALIZE"`

LFEQualizer:TDELay

Command   `:SPRocessing:LFEQualizer:TDELay <delay_value>`
Sets or queries the tap delay value of the the Linear Feedforward Equalizer application. The equalizer tap delay setting must first be set to CUSTom using the LFEQualizer:TDMode command.

Query   `:SPRocessing:LFEQualizer:TDELay?`

Returned Format   `[:SPRocessing:LFEQualizer:TDELay] <delay_value> <NL>`

Examples   `10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:TDMODE CUSTOM"`
            `20 OUTPUT 707;":SPROCESSING:LFEQUALIZER:TDELAY 1.607E-9"`

LFEQualizer:TDMode

Command   `:SPRocessing:LFEQualizer:TDMode {TBITrate | CUSTom}`

Sets or queries the tap delay mode. TBITrate specifies tracking of the bitrate. CUSTom allows you to enter a specific delay value using the LFEQualizer:TDELay command.

Query   `:SPRocessing:LFEQualizer:TDMode?`

Returned Format   `[:SPRocessing:LFEQualizer:TDMode] {TBITrate | CUSTom}<NL>`

Example   `10 OUTPUT 707;":SPROCESSING:LFEQUALIZER:TDMODE TBITRATE"`

MATLab

Command   `:SPRocessing:MATLab {ON | OFF | 1 | 0}`

Turns on and off the MATLAB Filter application. If MATLAB is not already running on the instrument, it is automatically started and is minimized.

Query   `:SPRocessing:MATLab?`

Returned Format   `[:SPRocessing:MATLab] {ON | OFF | 1 | 0}<NL>`

Example   `10 OUTPUT 707;":SPROCESSING:MATLAB ON"`

MATLab:ETENable

Command   `:SPRocessing:MATLab:ETENable {ON | OFF | 1 | 0}`

Enables or disables the capture of the text that is normally displayed in the MATLAB Command Window when a script is run. Use the MATlab:ETEXt command to retrieve the actual text.

Query   `:SPRocessing:MATLab:ETENable?`

Returned Format   `[:SPRocessing:MATLab:ETENable] {ON | OFF | 1 | 0}<NL>`

Example   `10 OUTPUT 707;":SPROCESSING:MATLAB:ETENABLE ON"`

MATLab:ETEXt

Command   `:SPRocessing:MATLab:ETEXt?`

Queries the MATLAB script engine text that is displayed in MATLAB's Command Window. This command is valid only when the MATLAB script engine's text capture is turned on as specified by the MATlab:ETENable command.

Returned Format   `[:SPRocessing:MATLab:ETEXt] <string><NL>`

Example   `10 OUTPUT 707;":SPROCESSING:MATLAB:ETEXT?"`

MATLab:SCRipt

Command   `:SPRocessing:MATLab:SCRipt <file_name>`

Selects the MATLAB script file for the MATLAB Filter application. Also, queries the selected script file name with path. <file_name> is the name of the file, with a maximum of 254 characters (including the path name, if used). If a path does not precede the file name, the file name assumes the default directory for scripts.

Query   `:SPRocessing:MATLab:SCRipt?`

Returned Format   `[:SPRocessing:MATLab:SCRipt] <file_name><NL>`

Example   `10 OUTPUT 707;":SPROCESSING:MATLAB:SCRIPT "d:\user files\matlab scripts\my script.m""`

OUTPut

Command   `:SPRocessing:OUTPut {FUNCtion<n>}`

Selects the math function (F1, F2, F3, or F4) for the output of the active signal processing application. <n> is the numeral 1, 2, 3, or 4 representing one of four math functions.

Query   `:SPRocessing:OUTPut?`

Returned Format   `[:SPRocessing:OUTPut] {FUNCtion<n>}<NL>`

Example   `10 OUTPUT 707;":SPROCESSING:OUTPUT FUNCTION2"`

SOURce

Command   `:SPRocessing:SOURce {CHANnel<n> | FUNCtion<n>}`

Selects an input channel (CH1 or CH2) or a math function (F1, F2, F3, or F4) for the input to the active signal processing application. <n> is the numeral 1, 2, 3, or 4 representing one of two input channels or one of four math functions.

Query   `:SPRocessing:SOURce?`

Returned Format   `[:SPRocessing:SOURce] {CHANnel<n> | FUNCtion<n>}<NL>`

Example   `10 OUTPUT 707;":SPROCESSING:SOURCE CHANNEL1"`

SOURce:DISPlay

Command   `:SPRocessing:SOURce:DISPlay {ON | OFF | 1 | 0}`

Turns on or off the display of the selected source for the active signal processing application.

Query   `:SPRocessing:SOURce:DISPlay?`

Returned Format   `[:SPRocessing:SOURce:DISPlay] {1 | 0}<NL>`

Example   `10 OUTPUT 707;":SPROCESSING:SOURCE:DISPLAY ON"`

22 TDR/TDT Commands (Rev. A.06.00 and Above)

With the introduction of software revision A.06.00, extensive changes were made to the TDR/TDT capability of the instrument. Consequently, changes were required to this command subsystem. Refer to the previous chapter for documentation on the command for software revision A.05.00 and below. If Option 202, Enhanced Impedance and S-Parameter Software, is installed, you can display and save S-parameters. Refer to Chapter 20, "S-Parameter Commands (Rev. A.07.00 and Below)".

Introduction

Use the STIMulus:MODe command to select single-ended, differential, or common mode measurements. Use STIMulus:STATe to turn on and off the stimulus. Always issue the the STIMulus:MODe command before the STIMulus:STATe command. Channel connections are established using the RESPonse:CONNect command. Refer to "CONNect" on page 312.

KEYSIGHT
TECHNOLOGIES

Table 44   TDR/TDT Commands

Commands (Revision A.06.00)	Retained Commands (Revision A.05.00 and Below)	Obsolete Commands
CONNect		DCALib
DUT:DIRection		HPOLarity
DUT:TYPE		NVALid?
RESPonse:CALibrate	RESPonse:CALibrate	PRESet
RESPonse:DISPlay		RATE
RESPonse:RISetime	RESPonse:RISetime	RESPonse
RESPonse:RPLane?		RESPonse:CALibrate:CANCel
RESPonse:TYPE		RESPonse:CALibrate:CONTinue
RESPonse:VAMPlitude		RESPonse:HORizontal
RESPonse:VERTical	RESPonse:VERTical	RESPonse:HORizontal:POSition
RESPonse:VERTical:OFFSet	RESPonse:VERTical:OFFSet	RESPonse:HORizontal:RANGe
RESPonse:VERTical:RANGe	RESPonse:VERTical:RANGe	RESPonse:TDRDest
RESPonse:VLOad		RESPonse:TDRTDT
STIMulus:EXTernal		RESPonse:TDTDest
STIMulus:EXTernal:POLarity		STIMulus
STIMulus:MODe		
STIMulus:RATE		
STIMulus:STATe		

Module Channel Identification

In previous software revisions, each TDR/TDT subsystem command identified the TDR module installation (left or right mainframe slot) with the form :TDR{2:4}:<command>. Starting with software revision A.06.00, the TDR/TDT subsystem no longer uses this identification scheme; the new syntax form is simply :TDR:<command>.

TDR/TDT Calibration

TDR/TDT calibration corrects for measurement system effects and locates the reference plane of the step response. The reference plane is the time (or distance) of the incident step and is the location that all subsequent impedance measurements are made relative to. Starting with software revision A.06.00 and above, TDR/TDT Calibration replaces the normalization and reference plane calibration. TDR/TDT Calibration allows marker time readouts relative to the reference plane but, in addition, adds the ability to change the time base setting, corrects for systematic errors, and enables a pulse

rise time filter to simulate real step responses. For best results, before starting the TDR/TDT calibration place the step response at the reference plane within the first graticule division as shown the following picture.

1st graticule
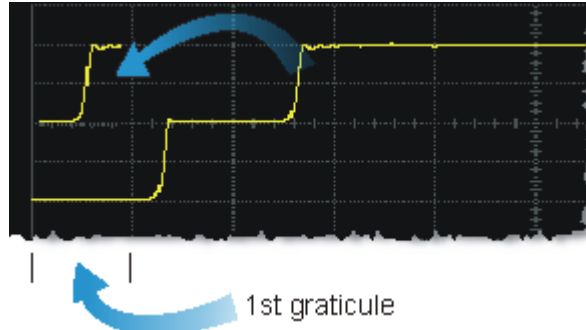
The calibration commands step through the TDR/TDT Calibration Wizard. Send "RESPonse:CALibrate" on page 313 followed by "SDONe?" on page 139 to begin the calibration. Use the returned string from the SDONe? query to determine when a calibration step has completed. If you set a time out value, make sure that the value is set long enough to allow the measurement to complete. SDONe? returns the prompt string for the next step. After making the test setup connections for a calibration step, send "CONTinue" on page 133 followed by SDONe?. At the end of the last step, SDONe? returns the string "Done".

NOTE   Once the module calibration procedure is started, all access to the instrument's front panel is blocked, including the use of the Local button. Pressing Local during a module calibration will not place the instrument in local mode. The calibration must either be cancelled (using "CANCel" on page 133) or finished before you can regain control to the instrument's front panel. Failure of a calibration step results in that step being repeated.

More Information

Option 202 TDR Peeling is implemented as a math function. Refer to "PEELing" on page 192. To perform the measurements that are listed on the measurement toolbar, refer to Chapter 18, "Measure Commands"

Commands

CONNect

Command	`:TDR:CONNect CHANnel<N>, {DUTPort<N>	NONE}`

Enters the measurement setup connections between the instrument channels and the test device ports. Use the NONE argument to delete a previously established connection. For example, to set up a return loss (s11) measurement on a single-ended device, you could send the following command
`10 OUTPUT 707;":TDR:CONN CHAN1, DUTP1"`
to connect channel 1 on the TDR module to port 1 on the test device.

For differential and common-mode connections, specify either channel of the pair to connect both paths, as both lines on a balanced connection are considered one port. For example, the above command would connect channels 1 and 2 to port 1 on the test device. Including the CHAN1 argument automatically selects channel 2 for the other side of the balanced line.

Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:CONN CHAN1, DUTP1"`
Query	The query returns only the short form of the command, DUTP1. The long form is not returned even if :SYSTem:LONGform is on. `:TDR:CONNect? CHANnel<N>`
Returned Format	`[:TDR:CONNect] CHANnel<N>, {DUTP<N>	NONE}<NL>`

DUT:DIRection

Command	`:TDR:DUT:DIRection {FORWard	REVerse}` Selects the direction of the stimulus through the test device: forward or reverse.
Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:DUT:DIR FORW"`
Query	`:TDR:DUT:DIR?`
Returned Format	`[:TDR:DUT:DIRection] {FORWard	REVerse}<NL>`

DUT:TYPE

Command	`:TDR:DUT:TYPE {D1Port	D2Port	D2PThru	D4Port}` Selects the type of device that you are measuring.

Table 45   Device Type Arguments

Argument	Device Type	Description
D1Port		One-port single-ended device
D2Port		Two-port single-ended device. Or, one port differential/common mode input.
D2PThru		Two-port device. Single-ended input, single-ended output.

Table 45   Device Type Arguments

Argument	Device Type	Description
D4Port		Four-port single-ended device. Or, two port differential/common mode input.

Restrictions	Software revision A.06.00 and above. TDR mode.
Example	10 OUTPUT 707;":TDR:DUT:TYPE D1PORT"
Query	The query returns only the short form of the command. For example D1P, D2P, D2PT, or D4P. The long form is not returned even if :SYSTem:LONGform is on. :TDR:DUT:TYPE?
Returned Format	[:TDR:DUT:TYPE] {D1P | D2P | D2PT | D4P}<NL>

RESPonse:CALibrate

Command	:TDR:RESPonse\<N\>:CALibrate

Initiates a TDR/TDT channel calibration. Setup the horizontal scale and position to view the test device on the display before starting a calibration. The argument <N> is an integer, 1 through 4, that identifies the *channel* to be calibrated. For TDR measurements, it is the channel that is the source of the TDR step pulse. For TDT measurements, it is the channel that receives the step pulse. For differential and common-mode measurements, you specify either channel of the pair to calibrate both paths. Refer to Table 46 on page 313 for several examples. Failure of a calibration step results in that step being repeated. Refer to "TDR/TDT Calibration" on page 310 for more information.

Send the query "SDONe?" on page 139 to determine when a calibration step has completed. If you set a time out value, make sure that the value is set long enough to allow the measurement to complete. SDONe? returns the prompt string for the next step. After making the test setup connections for a calibration step, send "CONTinue" on page 133 followed by SDONe?. At the end of the last step, SDONe? returns the string "Done".

Restrictions	Software revision A.06.00 and above. TDR mode.
Example	10 OUTPUT 707;":TDR:RESPONSE1:CALIBRATE"

Table 46   Examples of Command with Channel Identification

Calibration	Command
Single-ended TDR, Channel 1	TDR:RESPonse1:CALibrate
Single-ended TDT, Channel 2	TDR:RESPonse2:CALibrate
Differential TDR, Channel 1 and 2	TDR:RESPonse1:CALibrate *or* TDR:RESPonse2:CALibrate
Differential TDR, Channel 3 and 4	TDR:RESPonse3:CALibrate *or* TDR:RESPonse4:CALibrate
Differential TDT, Channel 3 and 4	TDR:RESPonse3:CALibrate *or* TDR:RESPonse4:CALibrate
Common mode TDR, Channel 1 and 2	TDR:RESPonse1:CALibrate *or* TDR:RESPonse2:CALibrate

RESPonse:DISPlay

Command	`:TDR:RESPonse<N>:DISPlay {ON	1	OFF	0 }`

Turns on or off the display of the indicated response waveform. The value <N> is an integer, 1 through 4, that identifies the response waveform.

Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:RESPONSE1:DISP ON"`
Query	`:TDR:RESPonse<N>:DISPlay?`
Returned Format	`[:TDR:RESPonse<N>:DISPlay] {ON	1	OFF	0 }<NL>`

RESPonse:RISetime

Command	`:TDR:RESPonse<N>:RISetime <risetime>`

Specifies the response risetime setting in seconds. Since there is only one risetime value shared by all calibrated responses, the value of <N> must be 1, 2, 3, or 4. Any of these four integers will have the same effect. You can select a risetime for TDR/TDT measurements that is close to the actual risetime used in your system. Valid risetime settings are bounded by the current timebase and record length settings. While the TDR step's rise time (which is applied to the device under test) is fixed, a set of mathematical operations is applied to the measured response to model the effect of the specified TDR step risetime. This risetime value applies to both TDR and TDT calibrated channels. All calibrated responses share the same risetime value.

Restrictions	Available in all software revisions. TDR mode.
Example	`10 OUTPUT 707;":TDR:RESPONSE1:RISETIME 100 PS"`
Query	`:TDR:RESPonse<N>:RISetime?`
Returned Format	`[:TDR:RESPonse<N>:RISetime] <risetime><NL>`

RESPonse:RPLane?

Query	`:TDR:RESPonse<N>:RPLane?`

Queries the reference plane position for TDR or TDT responses. The reference plane value is identical for and applies to all responses. A settings conflict error is reported if no stimulus channel is active. If the response is uncalibrated, a default value is returned. The value <N> is an integer, 1 through 4, that identifies the response waveform.

Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:RESPONSE1:RPLANE?"`
Returned Format	`[:TDR:RESPonse<N>:RPLane] <value><NL>`

RESPonse:TYPE

Command	`:TDR:RESPonse<N>:TYPE {CSINgle	CDIFf	CCOMmon	UDIFf	UCOMmon}`

Use with differential mode or common mode measurements to select the type of measurement for the indicated response. The value <N> is an integer, 1 through 4, that identifies the response waveform.

The command arguments are defined as follows:

- CSINgle selects a calibrated single-ended response
- CDIFf selects a calibrated differential mode response
- CCOMmon selects a calibrated common mode response
- UDIFf selects an uncalibrated differential mode response
- UCOMmon selects an uncalibrated common mode response

Restrictions   Software revision A.06.00 and above. TDR mode.

Example   `10 OUTPUT 707;":TDR:RESPONSE1:TYPE CDIFf"`

Query   The query returns only the short form of the command. For example CSIN, CDIF, CCOM, UDIF, or UCOM. The long form is not returned even if :SYSTem:LONGform is on.
`:TDR:RESPonse<N>:TYPE?`

Returned Format   `[:TDR:RESPonse<N>:TYPE] {CSINgle | CDIFf | CCOMmon | UDIF | UCOM}<NL>`

RESPonse:VAMPlitude?

Query   `:TDR:RESPonse<N>:VAMPlitude?`

Returns the TDR incident step amplitude (top – base) value that was measured by the instrument during a TDR calibration. The default value of 200 mV is returned if the normalization has not yet been done. The V amplitude value (V_{ampl}) can be used to calculate the impedance for TDR or TDT responses. The DCA marker system does this automatically. Use **"RESPonse:VLOad?"** on page 316 to return the value of V_{load}. Use the following equation for the calculation:

$$\text{Impedance (V)} = \frac{Z_0 \left((V_{ampl} - V_{load}) + V \right)}{\left((V_{ampl} + V_{load}) - V \right)}$$

where Z_0 equals 50 ohms in the instrument.

A settings conflict error is reported if no stimulus channel is active. If the response is uncalibrated, a default value is returned (200 mV). The value <N> is an integer, 1 through 4, that identifies the response waveform.

Restrictions   Software revision A.06.00 and above. TDR mode.

Example   `10 OUTPUT 707;":TDR:RESPONSE1:VAMPlitude?"`

Returned Format   `[:TDR:RESPonse<N>:VAMPlitude] <value><NL>`

RESPonse:VERTical

Command   `:TDR:RESPonse<N>:VERTical {AUTO | MANual}`

This command specifies whether the TDR/TDT response should automatically track the source channel's vertical scale (AUTO), or use a user-defined scale specified with the VERTical:OFFSet and VERTical:RANGe commands (MANual). AUTO is the usual setting. The keyword TSOurce may also be used. This command is compatible with the Agilent83480/54750 and is equivalent to AUTO. The value <N> is an integer, 1 through 4, that identifies the response waveform.

Restrictions   Available in all software revisions. TDR mode.

Example   `10 OUTPUT 707;":TDR:RESPONSE1:VERTICAL MANUAL"`

Query	`:TDR:RESPonse<N>:VERTical?`
Returned Format	`[:TDR:RESPonse<N>:VERTical] {AUTO	MANual}<NL>`

RESPonse:VERTical:OFFSet

Command   `:TDR:RESPonse<N>: VERTical:OFFSet <offset_value>`

This command sets the vertical position of the specified response and changes the vertical tracking setting to MANual if it is in AUTO. Refer to "RESPonse:VERTical" on page 315. The position is always referenced to center screen. The value <N> is an integer, 1 through 4, that identifies the response waveform. The <offset_value> argument is the offset value in the current channel UNITs. Suffix UNITs are ignored; only the scalar part is used (m in mv).

Restrictions   Available in all software revisions. TDR mode.

Example
```
10 OUTPUT 707;":TDR:RESPONSE1:VERTICAL MANUAL"
20 OUTPUT 707;":TDR:RESPONSE1:VERTICAL:OFFSET 50 MV"
```

Query   The information reterned from the query is only valid when the vertical tracking mode is set to manual.
`:TDR:RESPonse<N>:VERTical:OFFSet?`

Returned Format   `[:TDR:RESPonse<N>:VERTical:OFFSet] <volts><NL>`

RESPonse:VERTical:RANGe

Command   `:TDR:RESPonse<N>:VERTical:RANGe <range_value>`

This command specifies the vertical range of the TDR/TDT response and changes the vertical tracking setting to MANual if it is in AUTO. Refer to "RESPonse:VERTical" on page 315. The value <N> is an integer, 1 through 4, that identifies the response waveform. The <range_value> argument is in the current UNITs setting and suffix supplied. (The suffix does not set the UNITs; it is ignored.)

Restrictions   Available in all software revisions. TDR mode.

Example
```
10 OUTPUT 707;":TDR:RESPONSE1:VERTICAL MANUAL"
20 OUTPUT 707;":TDR:RESPONSE1:VERTICAL:RANGE 5 V"
```

Query   The information reterned from the query is only valid when the vertical tracking mode is set to manual.
`:TDR:RESPonse<N>:VERTical:RANGe?`

Returned Format   `[:TDR:RESPonse<N>:VERTical:RANGe] <volts><NL>`

RESPonse:VLOad?

Query   `:TDR:RESPonse<N>:VLOad?`

Returns the TDR incident step voltage into a 50 ohm impedance load that was measured by the instrument during a TDR calibration. This query returns the default value (200 mV), if the normalization has not yet been done. The V_{load} value for calculating the impedance for TDR or TDT responses. The DCA marker system does this automatically. Use "RESPonse:VAMPlitude?" on page 315 to return the value of $V_{amplitude}$. Use the equation listed under "RESPonse:VAMPlitude?" on page 315 to calculate the impedance. A settings conflict error is reported if no stimulus channel is active or if the query is sent for a TDT response. If the response is uncalibrated, a default value is returned (200 mV). The value <N> is an integer, 1 through 4, that identifies the response waveform.

Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:RESPONSE1:VLOAD?"`
Returned Format	`[:TDR:RESPonse<N>:VLOad] <value><NL>`

STIMulus:EXTernal

Command	`:TDR:STIMulus:EXTernal {ON	1	OFF	0 }`
	Specifies that an external pulse accelerator is being used in the test setup.
Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:STIM:EXT ON"`
Query	`:TDR:STIMulus:EXTernal?`
Returned Format	`[:TDR:STIMulus:EXTernal] {ON	1	OFF	0 }<NL>`

STIMulus:EXTernal:POLarity

Command	`:TDR:STIMulus:EXTernal:POLarity {POSitive	NEGative}[, {POSitive	NEGative}]`
	When using an external step accelerator, sets the polarity of the channels to match the polarity of the TDR remote head. For single-ended measurements, the first argument is required and defines the polarity of the external step. For differential or common mode measurements, both arguments are used with the second argument defining the second external step polarity.
Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:STIM:EXT:POL POS, NEG"`
Query	The query always returns both polarity values regardless of stimulus mode.
	`:TDR:STIMulus:EXTernal:POLarity?`
Returned Format	`[:TDR:STIMulus:EXTernal:POLarity] {POSitive	NEGative}, {POSitive	NEGative}<NL>`

STIMulus:MODE

Command	`:TDR:STIMulus:MODE {SINGle	DIFFerential	COMMon}`
	Sets the measurement stimulus to single-ended, differential, or common mode.
Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:STIM:MOD SING"`
Query	If :SYSTem:LONGform is ON, this query returns the following strings: SINGLEENDED, COMMONMODE, or DIFFERENTIAL. Note that, with the exception of DIFFERENTIAL, these strings do not match the long form argument strings for the command.
	`:TDR:STIMulus:MODE?`
Returned Format	`[:TDR:STIMulus:MODE] {SINGleended	DIFFerential	COMMonmode}<NL>`

STIMulus:RATE

Command	`:TDR:STIMulus:RATE { AUTO	<rate>}`

This command sets the period of the TDR pulse generator. You should usually leave this set to AUTO unless you need to define a specific rate. In AUTO, the instrument will attempt to keep subsequent periods off screen when the timebase is changed. <rate> is the period to which you want to set the generator, in Hertz. You can add a suffix to indicate that the rate is in Hertz (HZ, KHZ, and so on).

The query returns the current period of the pulse generator, even when the control is set to AUTO. The query is allowed in all modes.

Restrictions	Software revision A.06.00 and above. TDR mode.
Query	`:TDR:STIMulus:RATE?`
Returned Format	`[:TDR:STIMulus:RATE] <rate><NL>`

STIMulus:STATe

Command	`:TDR:STIMulus:STATe {CHANnel<N>	LMODule	RMODule}, {ON	1	OFF	0 }`
	Turns on and off the selected stimulus. Use the CHANnel argument for single-ended stimulus and the LMODule (left module) and RMODule (right modules) arguments for differential mode or common mode measurements.
Restrictions	Software revision A.06.00 and above. TDR mode.
Example	`10 OUTPUT 707;":TDR:STIM:STAT CHAN2, ON"`
Query	`:TDR:STIMulus:STATe? {CHANnel<N>	LMODule	RMODule}`
Returned Format	`[:TDR:STIMulus:STATe] {CHANnel<N>	LMODule	RMODule}, {ON	1	OFF	0 }<NL>`

23 TDR/TDT Commands (Rev. A.05.00 and Below)

The TDR/TDT command subsystem documents the commands used to set up TDR/TDT measurements in instruments with revision A.05.00 and below. If you are programming an instrument with software revision above A.05.00, refer to Chapter 22.

All of the TDR/TDT subsystem commands are of the form :TDR{2 | 4}:<command>. The {2 | 4} option is used to identify the slot in which you have installed the TDR/TDT plug-in module. Select 2 if the module is in slots 1 and 2; 4 if the module is in slots 3 and 4. For example, if the module is in slots 3 and 4, and you want to issue the TDR subsystem PRESet command, you use the command string :TDR4:PRESET.

DCALib

Command   `:TDR{2 | 4}:DCALib {RPCalib | NORMal | QNORmal}`

This command allows you to select the type of differential normalization (or calibration) to be performed. In TDT mode, the NORMal and QNORmal procedures are equivalent; only the NORMal parameter is recognized. RPCalib selects reference plane calibration. This option is provided for

backward compatibility. NORMal sets the calibration procedure to differential normalization. This version of the differential normalization procedure models the coupling between the test fixture channels, and compensates for its effects. QNORmal sets the calibration procedure to differential normalization. This version of the differential normalization procedure, also known as "Quick Normalization", assumes that the coupling between the test fixture channels is negligible.

Restrictions	Software revision A.05.00 and below. TDR mode.
Query	`:TDR{2 | 4}:DCALib?`
	The query returns the select calibration mode.
Returned Format	`[:TDR{2 | 4}:DCAL] {RPCalib | NORMal | QNORmal}<NL>`
Example	`10 OUTPUT 707;":TDR2:DCAL QNOR"`

HPOLarity

Command	`:TDR{2 | 4}:HPOLarity {POSitive | NEGative}`
	Use this command when performing differential measurements with an external step generator. In the test setup, you can connect either a positive or a negative TDR remote head on the second channel. This command sets the polarity of the second channel to match that of the TDR remote head thus ensuring the proper display of the response.
Restrictions	Software revision A.04.20 and A.05.00. TDR mode.
Example	`10 OUTPUT 707;":TDR2:HPOLARITY NEGATIVE"`
Query	`:TDR{2 | 4}:HPOLarity?`
Returned Format	`[:TDR{2 | 4}:HPOLarity] {POSitive | NEGative}<NL>`

NVALid?

Query	`:TDR{2 | 4}:NVALid?`
	Queries the specified TDR module to determine if valid normalization data exists. A 1 is returned, if a valid normalization exists. Otherwise, a 0 is returned.
Restrictions	Software revision A.04.20 and A.05.00. TDR mode.
Example	`10 OUTPUT 707;":TDR2:NVALid?"`
Returned Format	`[:TDR{2 | 4}:NVALid] {1 | 0}<NL>`
Returned Format	`[:TDR{2 | 4}:RESPonse<N>] {1 | 0}<NL>`

PRESet

Command	`:TDR{2 | 4}:PRESet`
	This command performs an automatic set up of the instrument for TDR or TDT measurements, based on the stimulus. This command does the following:

- Turn on TDR channels.
- If the stimulus is set to EXTernal ( ), turn off channel 1 or 3 and turn on channel 2 or 4.

- If the TDT destinations are not shown, turn on the TDT destination channels. (**"RESPonse:TDTDest"** on page 325).
- Set the timebase to 500 ps/div and positions the incident edge on screen.
- Turn on averaging and set best flatness (**"AVERage"** on page 123).
- For all channels that are on:
- Set the attenuation units to ratio.
- Set the attenuation to 1:1.
- Set the bandwidth to low (12.4 GHz). (Set high for external stimulus.)
- Set the units to volts.
- Set the channel scale to 100 mV/div.
- Set the channel offset to 200 mV or –200 mV for differential stimulus.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   The following example presets the instrument for TDR/TDT operations.
`10 OUTPUT 707;":TDR2:PRESET"`

RATE

Command   `:TDR{2 | 4}:RATE {AUTO | <rate>}`

This command sets the period of the TDR pulse generator. You should usually leave this set to AUTO unless you need to define a specific rate. In AUTO, the instrument will attempt to keep subsequent periods off screen when the timebase is changed. <rate> is the period to which you want to set the generator, in Hertz. You can add a suffix to indicate that the rate is in Hertz (HZ, KHZ, and so on).

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   `10 OUTPUT 707;":TDR2:RATE 120 KHZ"`

Query   `:TDR{2 | 4}:RATE?`

The query returns the current period of the pulse generator, even when the control is set to AUTO. The query is allowed in all modes.

Returned Format   `[:TDR{2 | 4}:RATE] {AUTO | <rate>}<NL>`

Example   `10 OUTPUT 707;":TDR2:RATE?"`

RESPonse

Command   `:TDR{2 | 4}:RESPonse<N> {ON | 1 | OFF | 0 | DIFFerential | COMMonmode | INDividual}`

This command turns on or off a TDR or TDT normalized response. <N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands. OFF turns off the response for the specified stimulus. ON turns on the normalized response of the channel.

The keyword NORMalize may also be used. This command is compatible with the Agilent83480/54750 and is equivalent to ON.

The DIFFerential argument turns on the differential response. COMMonmode turns on the common mode response. INDividual turns on the response for the corresponding channel. This option is valid for responses computed by the differential normalization procedure, as set by commands :TDR {2 | 4}:DCALib:NORMal or :TDR {2 | 4}:DCALib:QNORmal.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   The following example turns on common mode response on response 1.
```
10 OUTPUT 707;":TDR2:RESPONSE1 COMMONMODE"
```

Query   :TDR{2 | 4}:RESPonse<N>?

The query returns the current response setting for the specified stimulus. The query is allowed in all modes.

Returned Format   [:TDR{2 | 4}:RESPonse<N>] {OFF | DIFFerential | COMMonmode | INDividual | ON}<NL>

RESPonse:CALibrate

Command   :TDR{2 | 4}:RESPonse<N>:CALibrate

This command begins a TDR or TDT normalization and reference plane calibration. Which calibration is done (TDR or TDT) depends on the setting of the TDRTDT control. <N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands.

If the module needs calibration, this command automatically triggers a module calibration before the TDR or TDT normalization and reference plane calibration begins.

Once the module calibration procedure is started, all access to the instrument's front panel is blocked, including the use of the Local button. Pressing Local during a module calibration will not place the instrument in local mode. The calibration must either be cancelled or finished before you can regain control to the instrument's front panel.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   The following example begins a TDR or TDT calibration.
```
10 OUTPUT 707;":TDR2:RESPONSE1:CALIBRATE"
```

RESPonse:CALibrate:CANCel

Command   :TDR{2 | 4}:RESPonse<N>:CALibrate:CANCel

This command activates the cancel softkey during a TDR or TDT normalization and reference plane calibration. This command is retained for backward compatibility with the 83480/54750. The preferred command is :CALibrate:CANCel.

<N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   The following example cancels the current calibration operation.
```
10 OUTPUT 707;":TDR2:RESPONSE1:CALIBRATE:CANCEL"
```

RESPonse:CALibrate:CONTinue

Command   `:TDR{2 | 4}:RESPonse<N>:CALibrate:CONTinue`

This command activates the continue softkey during a TDR or TDT normalization and reference plane calibration. This command is retained for backward compatibility with the 83480/54750. The preferred command is :CALibrate:CONTinue.

<N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   The following example continues a paused calibration operation.
`10 OUTPUT 707;":TDR2:RESPONSE1:CALIBRATE:CONTINUE"`

RESPonse:HORizontal

Command   `:TDR{2 | 4}:RESPonse<N>:HORizontal {AUTO | MANual}`

This command specifies whether the TDR/TDT response should automatically track the source channel's horizontal scale (AUTO), or a user-defined scale specified with the HORizontal:POSItion and HORizontal:RANGe commands (MANual). AUTO is the usual setting. The keyword TSOurce may also be used. The value <N> is an integer, 1 through 4, that identifies the stimulus channel used to produce a response waveform. Because response waveforms are numbered based on the destination channel, for TDR commands, <N> and the response waveform number refer to the same waveforms. This is not the case for TDT related commands.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   `10 OUTPUT 707;":TDR2:RESPONSE1:HORIZONTAL AUTO"`

Query   `:TDR{2 | 4}:RESPonse<N>:HORizontal?`

Returned Format   `[:TDR{2 | 4}:RESPonse<N>:HORizontal] {AUTO | MANual}<NL>`

RESPonse:HORizontal:POSition

Command   `:TDR{2 | 4}:RESPonse<N>:HORizontal:POSition <position>`

This command specifies the horizontal position of the TDR/TDT response when horizontal tracking is set to manual. The position is always referenced to center screen. The value <N> is an integer, 1 through 4, that identifies the stimulus channel used to produce a response waveform. Because response waveforms are numbered based on the destination channel, for TDR commands, <N> and the response waveform number refer to the same waveforms. This is not the case for TDT related commands. The <position> argument is the offset from the center of the screen, in seconds.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   `10 OUTPUT 707;":TDR2:RESPONSE1:HORIZONTAL MANUAL"`
`20 OUTPUT 707;":TDR2:RESPONSE1:HORIZONTAL:POSITION 20E9"`

Query   The information reterned from the query is only valid when the horizontal tracking mode is set to manual.
`:TDR{2 | 4}:RESPonse<N>:HORizontal:POSition?`

Returned Format   `[:TDR{2 | 4}:RESPonse<N>:HORizontal:POSition] <position><NL>`

RESPonse:HORizontal:RANGe

Command   `:TDR{2 | 4}:RESPonse<N>:HORizontal:RANGe <range>`

This command specifies the range of the TDR/TDT response when the horizontal tracking is set to manual. The value <N> is an integer, 1 through 4, that identifies the stimulus channel used to produce a response waveform. Because response waveforms are numbered based on the destination channel, for TDR commands, <N> and the response waveform number refer to the same waveforms. This is not the case for TDT related commands. The <range> argument is the horizontal range in seconds.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   `10 OUTPUT 707;":TDR2:RESPONSE1:HORIZONTAL MANUAL"`
`20 OUTPUT 707;":TDR2:RESPONSE1:HORIZONTAL:RANGE 120 MS"`

Query   The information reterned from the query is only valid when the horizontal tracking mode is set to manual.
`:TDR{2 | 4}:RESPonse<N>:HORizontal:RANGe?`

Returned Format   `[:TDR{2 | 4}:RESPonse<N>:HORizontal:RANGe] <range><NL>`

RESPonse:RISetime

Command   `:TDR{2 | 4}:RESPonse<N>:RISetime <risetime>`

This command sets the risetime for the normalized response. The risetime setting is limited by the timebase settings and the record length. The normalize response function allows you to change the risetime of the normalized step. <N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands.

The <risetime> value specifies the risetime setting in seconds. The Risetime function allows you to change the normalized step's risetime within a range of values, with bounds established by the current timebase and record length settings. While the TDR step's risetime applied to the system under test is fixed, the measured response has a set of mathematical operations applied to it. These mathematical operations effectively change the displayed response to the system just as if a different TDR step risetime had actually been applied. This allows you to select a risetime for TDR/TDT measurements that is close to the actual risetime used in your system. This risetime value applies to both TDR and TDT normalized channels.

Restrictions   Software revision A.04.20 and A.05.00. TDR mode.

Example   `10 OUTPUT 707;"TDR2:RESPONSE1:RISETIME 100 PS"`

Query   `:TDR{2 | 4}:RESPonse<N>:RISetime?`

Returned Format   `[:TDR{2 | 4}:RESPonse<N>:RISetime] <risetime><NL>`

RESPonse:TDRDest

Command   `:TDR{2 | 4}:RESPonse{1 | 3}:TDRDest CHANnel<N>`

This command selects a TDR destination channel for an external stimulus. When you use an external stimulus, you must use this command to specify where the TDR channel is coming into the instrument. An external stimulus may be generated from channels 1 or 3 only.

A channel is valid as a TDR destination if it meets the following criteria:

· Must be an electrical channel.

- Must not have an active TDR stimulus.
- Must not be the destination of a TDT measurement.

<N> is an integer, 1 through 4.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   The following example sets channel 2 as the TDR destination channel for response 1:
```
10 OUTPUT 707;":TDR2:RESPONSE1:TDRDEST CHANNEL2"
```

Query   `:TDR{2 | 4}:RESPonse{1 | 3}:TDRDest?`
The query returns the current TDR destination channel for the selected response.

Returned Format   `[:TDR{2 | 4}:RESPonse{1 | 3}:TDRDest] <channel><NL>`

RESPonse:TDRTDT

Command   `:TDR{2 | 4}:RESPonse{1| 2| 3 | 4}:TDRTDT {TDR | TDT}`

This command controls the behavior of other :TDR{2| 4}:RESPonse commands and queries. A response waveform is fully specified by the TDRTDT setting, as well as by the stimulus value that is part of a "TDR{2 | 4}:RESPonse" command.

<N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   To turn on Response 1 waveform as TDR with stimulus = Chan1:
```
:TDR2:RESPonse1:TDRTDT to TDR
:TDR2:RESPonse1 to NORM
```
To turn on Response 2 waveform as TDT with stimulus = Chan1:
```
:TDR2:RESPonse1:TDTDest to Chan2
:TDR2:RESPonse1:TDRTDT to TDT
:TDR2:RESPonse1 to ON
```

RESPonse:TDTDest

Command   `:TDR{2 | 4}:RESPonse<N>:TDTDest {NONE | CHANnel<N>}`

This command selects a destination channel for a normalization measurement.

<N> is an integer, 1 through 4. This RESPonse<N> value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands.

For differential and common mode stimuli, the TDT destination is implied as follows:

- The TDT destination for channel 1 is channel 3.
- The TDT destination for channel 2 is channel 4.
- The TDT destination for channel 3 is channel 1.
- The TDT destination for channel 4 is channel 2.

A channel is valid as a TDT destination if it meets the following criteria:

- Must be an electrical channel.
- Must not have an active TDR stimulus.

- •   Must not be the destination of another TDT measurement.
- •   Must not be the destination of a TDR measurement (external stimulus only).

You must select a valid TDT destination before setting the TDRTDT control to TDT.

NONE   Deselects a channel as a TDT destination. This frees the channel to be the TDT destination of another TDR source.

<N>   For CHANnel<N>, this value is an integer, 1 through 4, indicating the slot in which the channel resides, followed by an optional A or B identifying which of two possible channels in the slot is being referenced.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   The following example selects channel 3 as the TDT destination channel for response 4.
`10 OUTPUT 707;":TDR2:RESPONSE4:TDTDEST CHANNEL3"`

Query   `:TDR{2 | 4}:RESPonse<N>:TDTDest?`
The query returns the current TDT destination channel for the specified response.

Returned Format   `[:TDR{2 | 4}:RESPonse<N>:TDTDest] {NONE | <channel>}<NL>`

RESPonse:VERTical

Command   `:TDR{2 | 4}:RESPonse<N>:VERTical {AUTO | MANual}`
This command specifies whether the TDR/TDT response should automatically track the source channel's vertical scale (AUTO), or use a user-defined scale specified with the VERTical:OFFSet and VERTical:RANGe commands (MANual). AUTO is the usual setting. The keyword TSOurce may also be used. This command is compatible with the Agilent83480/54750 and is equivalent to AUTO.

<N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands.

Restrictions   Software revision A.05.00 and below. TDR mode.

Example   `10 OUTPUT 707;":TDR2:RESPONSE1:VERTICAL MANUAL"`

Query   `:TDR{2 | 4}:RESPonse<N>:VERTical?`

Returned Format   `[:TDR{2 | 4}:RESPonse<N>:VERTical] {AUTO | MANual}<NL>`

RESPonse:VERTical:OFFSet

Command   `:TDR{2 | 4}:RESPonse<N>: VERTical:OFFSet <offset_value>`
This command sets the vertical position of the specified response when vertical tracking is set to MANual. The position is always referenced to center screen. <N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands. <offset_value> is the offset value in the current channel UNITs. Suffix UNITs are ignored; only the scalar part is used (m in mv).

Restrictions   Software revision A.05.00 and below. TDR mode.

Example	10 OUTPUT 707;":TDR2:RESPONSE1:VERTICAL MANUAL" 20 OUTPUT 707;":TDR2:RESPONSE1:VERTICAL:OFFSET 50 MV"

Query	The information reterned from the query is only valid when the vertical tracking mode is set to manual. `:TDR{2 | 4}:RESPonse<N>:VERTical:OFFSet?`

Returned Format	`[:TDR{2 | 4}:RESPonse<N>:VERTical:OFFSet] <volts><NL>`

RESPonse:VERTical:RANGe

Command	`:TDR{2 | 4}:RESPonse<N>:VERTical:RANGe <range_value>`

This command specifies the vertical range of the TDR/TDT response when the vertical tracking mode is set to MANual. <N> is an integer, 1 through 4. This value refers to the stimulus channel used to produce a response waveform, while the response waveforms are numbered based on the destination channel. For TDR commands, the response waveform numbers and RESPonse<N> refer to the same waveforms. This is not the case for TDT related commands. <range_value> is in the current UNITs setting and suffix supplied. (The suffix does not set the UNITs; it is ignored.)

Restrictions	Software revision A.05.00 and below. TDR mode.

Example	10 OUTPUT 707;":TDR2:RESPONSE1:VERTICAL MANUAL" 20 OUTPUT 707;":TDR2:RESPONSE1:VERTICAL:RANGE 5 V"

Query	The information reterned from the query is only valid when the vertical tracking mode is set to manual. `:TDR{2 | 4}:RESPonse<N>:VERTical:RANGe?`

Returned Format	`[:TDR{2 | 4}:RESPonse<N>:VERTical:RANGe] <volts><NL>`

STIMulus

Command	`:TDR{2 | 4}:STIMulus {OFF | ON | ON1 | ON2 | ON1AND2 | ON3 | ON4 | ON3AND4 |` `COMMonmode | DIFFerential | ECOMmon | EDIFferential | EXTernal}`

This command turns the TDR/TDT stimulus on or off. This command is set before starting normalization to specify type of normalization or reference plane calibration to perform. For the differential stimulus setting, a reference plane calibration is executed *unless* you specify which normalization procedure is to be executed using the :TDR {2 | 4}:DCALib command.

- The stimulus may be OFF, ON, or EXTernal.
- In slots 1 and 2, the stimulus may be OFF, ON1, ON2, ON1AND2, DIFFerential, COMMonmode, EDIFferential, or ECOMmon.
- In slots 3 and 4, the stimulus may be OFF, ON3, ON4, ON3AND4, DIFFerential, COMMonmode, EDIFferential, or ECOMmon.

After specifying the TDR/TDT stimulus, use the command :TDR<N>:PRESET. This command will set up the instrument for TDR or TDT measurements based on the selected stimulus.

The argument, OFF, turns off the pulse generator, using the channel as a regular analyzer channel. ON, ON1, ON3, and External turn on the channel 1 or channel 3 pulse generator for single-ended TDR or TDT measurements. ON2 and ON4 turn on the channel 2 or channel 4 pulse generator for single-ended TDR or TDT measurements. ON1AND2 and ON3AND4 turn on the pulse generator for channels 1 and 2 or channels 3 and 4 for simultaneous single-ended TDR or TDT measurements. DIFFerential turns on the pulse generator for channels 1 and 2 or channels 3 and 4 for differential TDR or TDT measurements. COMMonmode turn on the pulse generator for channels 1 and 2 or channels 3 and 4 for common-mode TDR or TDT measurements. EDIFferential and ECOMmon turn

on the pulse generator for channels 1 and 2 (or channels 3 and 4) in either differential or common mode. The pulses are sent to an external pulse generator and the second pair of channels (3 and 4 or 1 and 2 respectively) are used as either TDR or TDT destinations.

Restrictions   Software revision A.04.20 and A.05.00. TDR mode.

Example   The following example turns on pulse generators for channels 3 and 4 for single-ended TDR measurements.
```
10 OUTPUT 707;":TDR4:STIMULUS ON3AND4"
```

Query   `:TDR{2 | 4}:STIMulus?`

The query returns the current settings for the TDR pulse generators.

Returned Format   `[:TDR{2 | 4}:STIMulus] {OFF | ON | ON1 | ON2 | ON1AND2 | DIFFerential | COMMonmode | EXTernal | ON3 | ON4 | ON3AND4}<NL>`

24 Timebase Commands

The TIMebase subsystem commands control the horizontal (X axis) analyzer functions.

BRATe

Command   `:TIMebase:BRATe <bit_rate>`

Sets the bit rate used when the time base units are bit period. <bit_rate> is the bit rate (in bits-per-second).

Query   `:TIMebase:BRATe?`

The query returns the bit rate setting.

Returned Format   `[:TIMebase:BRATe] <bit_rate><NL>`

Examples   The following example sets the bit rate to 155.520 MHz.
`10 OUTPUT 707;":TIMEBASE:BRATe 155.520E6"`

MPOSition

Command   `:TIMebase:MPOSition <trigger_delay>`

Reduces the trigger's minimum timebase position. Use in Jitter Mode when making measurements on devices that employ a modulated clock. Jitter measurements on devices with a modulated clock can result in artificially higher measured jitter levels, due to the instrument's trigger delay. Reducing the minimum timebase position reduces the amount of observed jitter. The default value is 40.1 ns. Reduce to its minimum value of 24.1 ns. The minimum timebase position setting is ignored when the 86108A/B Precision Waveform Analyzer module is in use.

To learn more about the trigger position and modulated clocks, refer Agilent Product Note 86100-5, *Triggering Wide-Bandwidth Sampling Oscilloscopes For Accurate Displays of High-Speed Digital Communications Waveforms*. You can download this product note from the 86100C/D product page on Agilent's web site.

Restrictions   Software revision A.06.00 and above.

Query   `:TIMebase:MPOSition? <trigger_delay>`

The query returns the current delay value in seconds.

Returned Format   `[:TIMebase:MPOSition] <trigger_delay><NL>`

Examples   `10 OUTPUT 707;":TIMEBASE:MPOSITION 20E-9"`

POSition

Command   `:TIMebase:POSition <position_value>`

Sets the time interval between the trigger event and the delay reference point. The delay reference point is set with the TIMebase:REFerence command. The <position_value> argument's maximum value depends on the time-per-division setting. The value can optionally have units of bits or seconds, refer to Table 5 on page 16 to view the suffix units. If no units are specified, <position_value> has the units of the current units setting.

Restrictions   In Jitter Mode, scale and position controls are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error. In TDR/TDT mode, the delay reference point is set to coincide with the reference plane position.

Query   `:TIMebase:POSition? [{BITS | TIME}]`

Returns the current delay value in seconds. BITS specifies the bits-per-screen at bit rate. TIME specifies seconds-per-division. If you have zoomed the display to show a portion of the waveform, the query returns the value corresponding to the zoomed waveform rather than the entire waveform.

Returned Format   `[:TIMebase:POSition] <position_value><NL>`

Examples   `10 OUTPUT 707;":TIMEBASE:POSITION 2E-3"`

PRECision

Command   `:TIMebase:PRECision {ON | 1 | OFF | 0}`

Enables and disables the precision timebase. Enabling the precision timebase will also set the time reference. Disabling the precision timebase invalidates the time reference.

Restrictions   Requires Agilent86107A Precision Timebase Module or the 86108A/B Precision Waveform Analyzer (with firmware revision A.08.00).

Query   `:TIMebase:PRECision?`

Returns the state of the precision timebase.

Returned Format   `[:TIMebase:PRECision?] {0 | 1}<NL>`

Examples   `10 OUTPUT 707;":TIMEBASE:PRECISION ON"`

PRECision:REFSource

Command   `:TIMebase:PRECision:REFSource {INTernal | EXTernal}`

Selects the internal or external reference source used for the precision timebase in the 86108A/B.

Restrictions   Requires an 86108A/B Precision Waveform Analyzer (with firmware revision A.08.00).

Query   `:TIMebase:PRECision:REFSource?`

Returned Format   `[:TIMebase:PRECision:REFSource?] {INTernal | EXTernal}<NL>`

Examples   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":TIMEBASE:PRECISION:REFSource EXT"`

PRECision:RFRequency

Command   `:TIMebase:PRECision:RFRequency <frequency>`

Specifies the frequency of the reference clock at the input of the 86107A. The <frequency> argument is dependent upon the 86107A option number (9.0 GHz to 12.6 GHz and 18.0 GHz to 25.0 GHz for option 020 or, additionally, 38.0 GHz to 43.0 GHz for option 040).

Restrictions   Requires Agilent86107A Precision Timebase Module or the 86108A/B Precision Waveform Analyzer (with firmware revision A.08.00).

Query   `:TIMebase:PRECision:RFRequency?`

Returns the user specified frequency of the reference clock.

Returned Format   `[:TIMebase:PRECision:RFRequency?] <frequency><NL>`

Examples   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":TIMEBASE:PRECISION:RFREQUENCY?"`

PRECision:RFRequency:AUTodetect

Command   `:TIMebase:PRECision:RFRequency:AUTodetect {ON | 1 | OFF | 0}`

Enables and disables automatic detection of the external precision-timebase reference frequency during autoscale. When using the 86108A/B, the instrument may be unable to correctly detect an external clock source during an Autoscale resulting in errors. If this happens, use this command to turn off autodetection of the external clock source. This is equivalent to clearing the Auto Detect checkbox. This check box is visible in the Precision Timebase dialog box when an External reference clock source is selected while pattern lock is turned on.

Restrictions   Requires Agilent86107A Precision Timebase Module or the 86108A/B Precision Waveform Analyzer (with firmware revision A.08.00).

Query   `:TIMebase:PRECision:RFRequency:AUTodetect?`

Returned Format   `[:TIMebase:PRECision:RFRequency:AUTodetect?] {ON | 1 | OFF | 0}<NL>`

Examples   `10 OUTPUT 707;":SYSTEM:HEADER OFF"`
`20 OUTPUT 707;":TIMEBASE:PRECISION:RFREQUENCY:AUTODETECT ON"`

PRECision:TREFerence

Command   `:TIMebase:PRECision:TREFerence`

Sets the time reference. If the time reference fails to set, an error is produced.

Restrictions

Requires Agilent86107A Precision Timebase Module or the 86108A/B Precision Waveform Analyzer (with firmware revision A.08.00).

Query	`:TIMebase:PRECision:TREFerence?`

Returns whether the time reference has been successfully set. It does not indicate whether the time reference is still valid. A return value of 1 indicates the time reference was successfully set the last time the :TIMebase:PRECision:TREFerence command was sent (or the "Reset Time Reference" button was selected). A return value of 0 indicates the time reference was not successfully set either by the :TIMebase:PRECision:TREFerence command or by the front-panel "Reset Time Reference" button. The usual causes for not being able to set the time reference include missing, too small, or too large signal or the frequency is not in the specified ranges.

This query does not indicate whether the time reference is invalid due to a change in either frequency or amplitude of the time reference signal. Use "PTER?" on page 112 to query the Precision Timebase Event Register to identify whether the timebase reference is still valid.

Returned Format	`[:TIMebase:PRECision:TREFerence] {0	1}`
Example	`10 OUTPUT 707;":TIMEBASE:PRECISION:TREFERENCE?"`

RANGe

Command	`:TIMebase:RANGe <full_scale_range>`

Sets the full-scale horizontal time in seconds. The range value is ten times the time-per-division value. Range is always set in units of time (seconds), not in bits. <full_scale_range> is the full-scale horizontal time in seconds.

NOTE In Jitter Mode, scale and position controls are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Query	`:TIMebase:RANGe?`

Returns the current full-scale horizontal time. If you have zoomed the display to show a portion of the waveform, the query returns the value corresponding to the zoomed waveform rather than the entire waveform.

Returned Format	`[:TIMebase:RANGe] <full_scale_range><NL>`
Examples	`10 OUTPUT 707;":TIMEBASE:RANGE 10E-3"`

REFerence

Command	`:TIMebase:REFerence {LEFT	CENTer}`

Sets the delay reference to the left or center side of the display.

Query	`:TIMebase:REFerence?`

Returns the current delay reference position.

Returned Format	`[:TIMebase:REFerence] {LEFT	CENTer}<NL>`
Example	`10 OUTPUT 707;":TIMEBASE:REFERENCE?"`

SCALe

Command	`:TIMebase:SCALe <value>`

Sets the time base scale. This corresponds to the horizontal scale value displayed as time-per-div on the instrument's screen. The <value> argument can optionally have units of bits or seconds, refer to Table 5 on page 16 to view the suffix units. If no units are specified <value> has units of the current units setting, seconds for time-per-division and bits for bits on screen at bit rate setting.

NOTE	In Jitter Mode, scale and position controls are disabled. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Query	`:TIMebase:SCALe? [{BITS	TIME}]`

Returns the current scale time setting. BITS specifies bits-per-screen at bit rate. TIME specifies seconds-per-division. If the optional parameter is omitted, the scale value returned is in the units of the current units setting (bits or time). If you have zoomed the display to show a portion of the waveform, the query returns the value corresponding to the zoomed waveform rather than the entire waveform.

Returned Format	`[:TIMebase:SCALe] <time><NL>`
Examples	`10 OUTPUT 707;":TIMEBASE:SCALE 10E-3"`

UNITs

Command	`:TIMebase:UNITs {TIME	BITS}`

Sets the time base units.

Query	`:TIMebase:UNITs?`

Returns the time base units.

Returned Format	`[:TIMebase:UNITs] {TIME	BITS}<NL>`
Example	`10 OUTPUT 707;":TIMEBASE:UNITs?"`

25 Trigger Commands

The TRIGger subsystem commands define the conditions for triggering and have been defined to closely represent the front-panel trigger selections. Edge triggering identifies a trigger condition by looking for the slope (rising or falling) and voltage level (trigger level) on the source you select. Any input channel, auxiliary input trigger (4-channel scopes only), line, or external trigger (2-channel scopes only) inputs can be used as the trigger source.

ATTenuation

Command   `:TRIGger:ATTenuation <attenuation factor>[,{RATio | DECibel}]`

Controls the attenuation factor and units. The default attenuation factor value is 1:1. The default attenuation units is ratio.

Query   :TRIGger:ATTenuation?

Returns the current attenuation factor and units.

Returned Format   `[:TRIGger:ATTenuation] <attenuation factor>[,{RATio | DECibel}]<NL>`

BRATe

Command   `:TRIGger:BRATe <bit_rate>`

Sets the bit rate when the trigger is in pattern lock mode.

KEYSIGHT
TECHNOLOGIES

Restrictions	86100D or 86100C (software revision A.04.00 and above.
Query	`:TRIGger:BRATe?`
	Returns the current setting of the bit rate.
Returned Format	`[:TRIGger:BRATe] <bit_rate><NL>`
Example	`10 OUTPUT 707; ":TRIGger:BRATe 1E9"`

BRATe:AUTodetect

Command	`:TRIGger:BRATe:AUTodetect {{ON	1}	{OFF	0}}`
	Enables or disables automatic detection of the bit rate. When disabled, use the :TRIGger:BRATe command to set the bit rate. When enabled, use the :TRIGger:PLOCk:AUTodetect command to initiate automatic detection.
Restrictions	86100D or 86100C (software revision A.04.00 and above.
Query	`:TRIGger:BRATe:AUTodetect?`
Returned Format	[:TRIGger:BRATe:AUTodetect] {1	0}<NL>
Example	`10 OUTPUT 707; ":TRIGger:BRATe:AUTodetect ON"`

BWLimit

Command	`:TRIGger:BWLimit {DIVided	HIGH	LOW}`
	Controls an internal lowpass filter and a divider in the trigger. The bandwidth of the trigger is limited to approximately 100 MHz. DIVided mode is unaffected by the level, hysteresis, and slope settings. The DIVided parameter is only valid if the mainframe has option 001.
Query	`:TRIGger:BWLimit?`
	Returns the current setting for the specified trigger input.
Returned Format	`[:TRIGger:BWLimit] {HIGH	LOW	DIV}<NL>`
Example	`10 OUTPUT 707;":TRIGGER:BWLIMIT LOW"`

DCDRatio

Command	`:TRIGger:DCDRatio <data_to_clock_divide_ratio>`
	Sets the data-to-clock divide ratio used by pattern lock trigger mode. <data_to_clock_divide_ratio> must be one of the following integers: 1, 2, 4, 5, 8, 10, 15, 16, 20, 25, 30, 32, 35, 40, 45, 50, 64, 66, 100, 128.
Restrictions	86100D or 86100C (software revision A.04.00 and above.
Query	`:TRIGger:DCDRatio?`
	Returns the current setting of data-to-clock divide ratio.
Returned Format	`[:TRIGger:DCDRatio] <data_to_clock_divide_ratio><NL>`
Example	`10 OUTPUT 707; ":TRIGger:DCDRatio 16"`

DCDRatio:AUTodetect

Command   `:TRIGger:DCDRatio:AUTodetect {{ON | 1} | {OFF | 0}}`

Enables or disables automatic detection of the data-to-clock divide ratio. When disabled, use the :TRIGger:DCDRatio command to set the data-to-clock divide ratio. When enabled, use the :TRIGger:PLOCk:AUTodetect command to initiate automatic detection.

Restrictions   86100D or 86100C (software revision A.04.00 and above.

Query   `:TRIGger:DCDRatio:AUTodetect?`

Returned Format   [:TRIGger:DCDRatio:AUTodetect] {1 | 0}<NL>

Example   `10 OUTPUT 707; ":TRIGger:DCDRatio:AUTodetect ON"`

GATed

Command   `:TRIGger:GATed {ON | 1 | OFF | 0}`

Enables or disables the ability of the instrument to respond to trigger inputs.

Query   `:TRIGger:GATed?`

Returns the current gated setting.

Returned Format   `[:TRIGger:GATed] {1 | 0}<NL>`

HYSTeresis

Command   `:TRIGger:HYSTeresis {NORMal | HSENsitivity}`

Specifies the trigger hysteresis . NORMal is the typical hysteresis selection. HSENsitivity gives minimum hysteresis and the highest bandwidth.

Query   `:TRIGger:HYSTeresis?`

Returns the current hysteresis setting.

Returned Format   `[:TRIGger:HYSTeresis] {NORMal | HSENSitivity}<NL>`

LEVel

Command   `:TRIGger:LEVel <level>`

Specifies the trigger level. Only one trigger level is stored in the instrument. <level> is the trigger level on all trigger inputs.

Query   `:TRIGger:LEVel?`

Returns the trigger level.

Returned Format   `[:TRIGger:LEVel] <level> <NL>`

PLENgth

Command   `:TRIGger:PLENgth <pattern_length>`

Sets the length of the pattern used in pattern lock trigger mode. <pattern_length> is an integer value in the range of 1 to 2^{15} in jitter mode and 1 to 2^{23} in the other instrument modes.

Restrictions	86100D or 86100C (software revision A.04.00 and above.
Query	`:TRIGger:PLENgth?`
	Returns the current setting of pattern length.
Returned Format	`[:TRIGger:PLENgth] <pattern_length><NL>`
Example	`10 OUTPUT 707; ":TRIGger:PLENgth 127"`

PLENgth:AUTodetect

Command	`:TRIGger:PLENgth:AUTodetect {{ON	1}	{OFF	0}}`
	Enables or disables automatic detection of the pattern length. When disabled, use the :TRIGger:PLENgth command to set the pattern length. When enabled, use the :TRIGger:PLOCk:AUTodetect command to initiate automatic detection.
Restrictions	86100D or 86100C (software revision A.04.00 and above.
Query	`:TRIGger:PLENgth:AUTodetect?`
Returned Format	`[:TRIGger:PLENgth:AUTodetect] {1	0}<NL>`
Example	`10 OUTPUT 707; ":TRIGger:PLENgth:AUTodetect OFF"`

PLOCk

Command	`TRIGger:PLOCk {{ON	1}	{OFF	0}}`
	Enables or disables pattern lock. When pattern lock is turned on, the 86100C internally generates a trigger synchronous with the user's pattern. Pattern lock is only available on an 86100C mainframe with Option 001 installed.
Restrictions	86100D or 86100C (software revision A.04.00 and above.
Query	`TRIGger:PLOCk?`
Returned Format	`[:TRIGger:PLOCk] {1	0}<NL>`
Example	`10 OUTPUT 707; ":TRIGger:PLOCk ON"`

PLOCk:AUTodetect

Command	`:TRIGger:PLOCk:AUTodetect`
	Executes autodetecting of pattern lock parameters.
Restrictions	86100D or 86100C (software revision A.04.00 and above.
Query	`:TRIGger:PLOCk:AUTodetect?`
	Returns a string explaining the results of the last autodetect. The string is empty if the last autodetect completed successfully. The returned string stays the same until the next autodetect is executed.
Returned Format	The following are examples of strings returned by this query. (The blank spaces are filled in with the appropriate numeric values.)
	`Detected trigger rate ___ is less than the minimum trigger rate of ___` `Unable to determine the pattern length`

                          Unable to determine the bit rate and trigger divide ratio
                          User supplied data rate ___ is not a multiple of detected trigger rate ___

        Example   10 OUTPUT 707; ":TRIGger:PLOCk:AUTodetect"

RBIT

        Command   :TRIGger:RBIT <relative_bit>
                     Sets the relative bit number used by pattern lock trigger mode. <relative_bit> is an integer with a
                     minimum value of 0 and a maximum value equal to the current pattern length setting minus one.

    Restrictions   86100D or 86100C (software revision A.04.00 and above.

        Query   :TRIGger:RBIT?
                     Returns the current setting of relative bit.

Returned Format   [:TRIGger:RBIT] <relative_bit><NL>

        Example   10 OUTPUT 707; ":TRIGger:RBIT 1023"

SLOPe

        Command   :TRIGger:SLOPe {POSitive | NEGative}
                     Specifies the slope of the edge on which to trigger.

        Query   :TRIGger:SLOPe?
                     Returns the slope for the trigger.

Returned Format   [:TRIGger:SLOPe] {POSitive | NEGative}<NL>

        Example   10 OUTPUT 707; ":TRIGger:SLOPe POSitive"

SOURce

        Command   :TRIGger:SOURce {FPANel | FRUN | LMODule | RMODule}
                     Selects the trigger input. Front panel (FPANel), left module (LMODule), and right module (RMODule)
                     are front panel inputs. Free run (FRUN) is internally generated, and is not affected by the settings of
                     gates, level, slope, bandwidth, or hysteresis.

        Query   :TRIGger:SOURce?
                     Returns the current trigger source of the current mode.

Returned Format   [:TRIGger:SOURce] <trigger><NL>

SOURce:AUTodetect

        Command   :TRIGger:SOURce:AUTodetect {ON | OFF}
                     Turns on or off automatic selection of the trigger source when using 86108A/B modules.

    Restrictions   86108A/B modules. Software revision A.08.00 and above. Requires a mainframe with the enhanced
                     trigger option (86100D-ETR or 86100C-001).

        Query   :TRIGger:SOURce:AUTodetect?

Returned Format   `[:TRIGger:SOURce] <trigger><NL>`

26 Waveform Commands

Use the WAVeform subsystem to transfer waveform data between a computer and the instrument.

KEYSIGHT
TECHNOLOGIES

Introduction

Data Acquisition

When the data is acquired using the DIGitize command, the data is placed in the channel or function memory of the specified source. After the DIGitize command, the analyzer is stopped. If the analyzer is restarted over GPIB or the front panel, the data acquired with the DIGitize command is overwritten. You can query the preamble, elements of the preamble, or waveform data while the analyzer is running, but the data will reflect only the current acquisition, and subsequent queries will not reflect consistent data. For example, if the analyzer is running and you query the X origin, the data is queried in a separate GPIB command, and it is likely that the first point in the data will have a different time than that of the X origin. This is due to data acquisitions that may have occurred between the queries. For this reason, Agilent does not recommend this mode of operation. Instead, you should use the DIGitize command to stop the analyzer so that all subsequent queries will be consistent. Function data is volatile and must be read following a DIGitize command or the data will be lost when the analyzer is turned off.

Waveform Data and Preamble

The waveform record consists of two parts: the preamble and the waveform data. The waveform data is the actual sampled data acquired for the specified source. The preamble contains the information for interpreting the waveform data, including the number of points acquired, the format of the acquired data, and the type of acquired data. The preamble also contains the X and Y increments, origins, and references for the acquired data. The values in the preamble are set when you execute the DIGitize command. The preamble values are based on the settings of controls in the ACQuire subsystem. Although you can change preamble values with a GPIB computer, you cannot change the way the data is acquired. Changing the preamble values cannot change the type of data that was actually acquired, the number of points actually acquired, etc.

NOTE	The waveform data and preamble must be read or sent using two separate commands: WAVeform:DATA and WAVeform:PREamble. When changing any waveform preamble values, be sure to set the points in the preamble to the same value as the actual number of points in the waveform. Otherwise, inaccurate data will result.

Data Conversion

Data sent from the analyzer must be scaled for useful interpretation. The values used to interpret the data are the X and Y origins, X and Y increments, and X and Y references. These values can be read from the waveform preamble.

Conversion from Data Value to Units

To convert the waveform data values (essentially A/D counts) to real-world units, such as volts, use the following scaling formulas:

$$\text{Y-axis units} \ = \ Y_{\text{increment}}(\text{data value} - Y_{\text{reference}}) + Y_{\text{origin}}$$

$$\text{X-axis units} \ = \ X_{\text{increment}}(\text{data value} - X_{\text{reference}}) + X_{\text{origin}}$$

where the data index starts at zero: 0, 1, 2, ..., n-1.

The first data point for the time (X-axis units) must be zero so the time of the first data point is the X origin.

| NOTE | This conversion is not required for waveform data values returned in ASCII format. |

Data Format for GPIB Transfer

There are four types of data formats that you can select with the WAVeform:FORMat command: ASCii, BYTE, WORD, and LONG. Refer to the FORMat command in this chapter for more information on data format.

Commands

BANDpass?

Query   :WAVeform:BANDpass?

Returns an estimate of the maximum and minimum bandwidth limits of the source signal. Bandwidth limits are computed as a function of the coupling and the selected filter mode. Cutoff frequencies are derived from the acquisition path and software filtering.

Returned Format   [:WAVeform:BANDpass]<upper_cutoff>,<lower_cutoff><NL>

<upper_cutoff> is the maximum frequency passed by the acquisition system. <lower_cutoff> minimum frequency passed by the acquisition system.

Example
```
10 DIM Bandwidth$[50]!Dimension variable
20 OUTPUT 707;":WAVEFORM:BANDPASS?"
30 ENTER 707;Bandwidth$
```

BYTeorder

Command   :WAVeform:BYTeorder {MSBFirst | LSBFirst}

Selects the order in which bytes are transferred to and from the analyzer using WORD and LONG formats. If MSBFirst is selected, the most significant byte is transferred first. Otherwise, the least significant byte is transferred first. The default setting is MSBFirst. MSBFirst is for microprocessors, like Motorola's, where the most significant byte resides at the lower address. LSBFirst is for microprocessors, like Intel's, where the least significant byte resides at the lower address.

Example   This example sets up the instrument to send the most significant byte first during data transmission.
```
10 OUTPUT 707;":WAVEFORM:BYTEORDER MSBFIRST"
```

Query   :WAVeform:BYTeorder?

The query returns the current setting for the byte order.

Returned Format   [:WAVeform:BYTeorder] {MSBFirst | LSBFirst}<NL>

Example
```
10 DIM Setting$[10]!Dimension variable
20 OUTPUT 707;":WAVEFORM:BYTEORDER?"
30 ENTER 707;Setting$
```

COUNt?

Query   :WAVeform:COUNt?

Returns the fewest number of hits in all of the time buckets for the currently selected waveform. For the AVERAGE waveform type, the count value is the fewest number of hits for all time buckets. This value may be less than or equal to the value specified with the ACQuire:COUNt command. For the NORMAL, RAW, INTERPOLATE, and VERSUS waveform types, the count value returned is one, unless the data contains holes (sample points where no data is acquired). If the data contains holes, zero is returned.

Returned Format   [:WAVeform:COUNt] <N><NL>

<N> is an integer. Values range from 1 to 262144 for NORMal, RAW, or INTerpolate types and from 1 to 32768 for VERSus type.

Example
```
10 DIM Count$[50]!Dimension variable
20 OUTPUT 707;":WAVEFORM:COUNT?"
30 ENTER 707;Count$
```

DATA

Command	`:WAVeform:DATA <block_data>[,<block_data>]`

Transfers waveform data to the instrument over GPIB and stores the data in a previously specified waveform memory. The waveform memory is specified with the WAVeform:SOURce command. Only waveform memories may have waveform data sent to them. The format of the data being sent must match the format previously specified by the waveform preamble for the destination memory.

VERSus data is transferred as two arrays. The first array contains the data on the X axis (from left to right side of the graticule), and the second array contains the data on the Y axis (from bottom to top of the graticule). The two arrays are transferred one at a time over GPIB in a linear format. There are n data points sent in each array, where n is the number in the points portion of the preamble.

CGRade data is transferred as a two dimensional array, 321 words high and 451 words wide. The array corresponds to the graticule display, where each word is a sample hit count. The array is transferred column by column, starting with the upper left corner of the graticule.

The full-scale vertical range of the A/D converter will be returned with the data query. Use the Y-increment, Y-origin, and Y-reference values to convert the full-scale vertical ranges to voltage values. Use the Y-range and Y-display values to plot the voltage values. All of these reference values are available from the waveform preamble. Refer to "Conversion from Data Value to Units" earlier in this chapter.

To return the histogram's source data, use the :WAVeform:SOURce command to select the HISTogram as the source and then use the :WAVeform:DATA? query to retrieve the histogram data.

NOTE This command operates on waveform data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Signal or trigger source selection is not available" error.

‹block_data›	Binary block data in the # format.
Example	This example sends 1000 bytes of previously saved data to the instrument from the array, Set.

```
10 OUTPUT 707 USING "#,K";:WAVEFORM:DATA #800001000"
20 OUTPUT 707 USING "W";Set(*)
```

NOTE BASIC Image Specifiers. # is an BASIC image specifier that suppresses the automatic output of the EOL sequence following the last output item. K is an BASIC image specifier that outputs a number or string in standard form with no leading or trailing blanks. W is an BASIC image specifier that outputs 16-bit words with the most significant byte first.

Query	`:WAVeform:DATA?`

The query outputs waveform data to the computer over the GPIB interface. The data is copied from a waveform memory, function, channel buffer, or histogram previously specified with the WAVeform:SOURce command. The returned data is described by the waveform preamble.

NOTE CGRade as Waveform Source. If the waveform source is CGRade, then the waveform fromat must be set to WORD. WORD is the only format that works with color grade data.

Returned Format	`[:WAVeform:DATA] <block_data>[,<block_data>]<NL>`
Example	This example places the current waveform data from channel 1 of the array Wdata in the word format.

```
10 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
20 OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1!Select source
30 OUTPUT 707;":WAVEFORM:FORMAT WORD"!Select word format
```

```
40 OUTPUT 707;":WAVEFORM:DATA?"
50 ENTER 707 USING "#,1A";Pound_sign$
53 ENTER 707 USING "#,1D";Header_length
55 ENTER 707 USING "#,"&VAL$(Header_length)&"D";Length
60 Length = Length/2!Length in words
70 ALLOCATE INTEGER Wdata(1:Length)
80 ENTER 707 USING "#,W";Wdata(*)
90 ENTER 707 USING "-K,B";End$
100 END
```

NOTE   BASIC Image Specifiers. # is an BASIC image specifier that terminates the statement when the last ENTER item is terminated. EOI and line feed are the item terminators. 1A is an BASIC image specifier that places the next character received in a string variable. 1D is an BASIC image specifier that places the next character in a numeric variable. W is an BASIC image specifier that places the data in the array in word format with the first byte entered as the most significant byte. -K is an BASIC image specifier that places the block data in a string, including carriage returns and line feeds until EOI is true or when the dimensioned length of the string is reached. B is an BASIC specifier that enters the next byte in a variable.

The format of the waveform data must match the format previously specified by the WAVeform:FORMat, WAVeform:BYTeorder, and WAVeform:PREamble commands.

FORMat

Command   `:WAVeform:FORMat {ASCii | BYTE | LONG | WORD}`

Sets the data transmission mode for waveform data output. This command controls how the data is formatted when the data is sent from the analyzer and pertains to all waveforms. The default format is ASCii.

ASCii   ASCII formatted data consists of ASCII digits with each data value separated by a comma. Data values can be converted to real values on the Y axis (for example, volts) and transmitted in floating point engineering notation. In ASCII:

- The value "99.999E+36" represents a hole level (a hole in the acquisition data).
- The value "99.999E+33" represents a clipped-high level.
- The value "99.999E+30" represents a clipped-low level.

BYTE   BYTE formatted data is formatted as signed 8-bit integers. If you use BASIC, you need to create a function to convert these signed bits to signed integers. In byte format:

- The value 125 represents a hole level (a hole in the acquisition data).
- The value 127 represents a clipped-high level.
- The value 126 represents a clipped-low level.

Data is rounded when converted from a larger size to a smaller size. For waveform transfer into the analyzer:

- The maximum valid qlevel is 124.
- The minimum valid qlevel is –128.

LONG   LONG formatted data is transferred as signed 32-bit integers in four bytes. If WAVeform:BYTeorder is set to MSBFirst, the most significant byte of each word is sent first. If the BYTeorder is LSBFirst, the least significant byte of each word is sent first. Long format is only applicable to histogram data sources. In long format:

- The value 2046820352 represents a hole level (no sample data at the current data point).
- Long format is only valid with histogram data sources.

WORD WORD formatted data is transferred as signed 16-bit integers in two bytes. If WAVeform:BYTeorder is set to MSBFirst, the most significant byte of each word is sent first. If the BYTeorder is LSBFirst, the least significant byte of each word is sent first. In word format:

· The value 31232 represents a hole level (no sample data at the current waveform data point).

· The value 32256 represents a clipped-high level.

· The value 31744 represents a clipped-low level.

For waveform transfer into the analyzer:

· The maximum valid qlevel is 30720.

· The minimum valid qlevel is –32736.

Example This example selects the WORD format for waveform data transmission.
```
10 OUTPUT 707;":WAVEFORM:FORMAT WORD"
```

Query `:WAVeform:FORMat?`

The query returns the current output format for transferring waveform data.

Returned Format `[:WAVeform:FORMat] {ASCii | BYTE | LONG | WORD}<NL>`

Example This example places the current output format for data transmission in the string variable, Mode$.
```
10 DIM Mode$[50]!Dimension variable
20 OUTPUT 707;":WAVEFORM:FORMAT?"
30 ENTER 707;Mode$
```

POINts?

Query `:WAVeform:POINts?`

Returns the points value in the current waveform preamble. The points value is the number of time buckets contained in the waveform selected with the WAVeform:SOURce command.

Returned Format `[:WAVeform:POINts] <points><NL>`

<points> An integer. Values range from 1 to 262144. See the ACQuire:POINts command for more information.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":WAVEFORM:POINTS?"
```

NOTE When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

See Also The ACQuire:POINts command in Chapter 6, "Acquire Commands".

PREamble

Command `:WAVeform:PREamble <preamble_data>`

Sends a waveform preamble to the previously selected waveform memory in the analyzer. The preamble contains the scaling and other values used to describe the data. The waveform memory is specified with the WAVeform:SOURce command. Only waveform memories may have waveform data sent to them. The preamble can be used to translate raw data into time and voltage values.

The following lists the elements in the preamble.

<preamble_data>	`<format>, <type>, <points>,<count>, <X increment>,<X origin>,< X reference>, <Y increment>, <Y origin>,<Y reference>, <coupling>, <X display range>, <X display origin>, <Y display range>, <Y display origin>, <date, string>, <time, string>, <frame model #, string>, <module #, string>, <acquisition mode>, <completion>, <X units>, <Y units>, <max bandwidth limit>, <min bandwidth limit>`
<date>	A string containing the data in the format DD MMM YYYY, where DD is the day, 1 to 31; MMM is the month; and YYYY is the year.
<time>	A string containing the time in the format HH:MM:SS:TT, where HH is the hour, 0 to 23, MM is the minute, 0 to 59, SS is the second, 0 to 59, and TT is the hundreds of seconds, 0 to 99.
<frame model #>	A string containing the model number and serial number of the frame in the format MODEL#:SERIAL#.
<format>	0 for ASCII format. 1 for BYTE format. 2 for WORD format.
<type>	1 for RAW type. 2 for AVERAGE type. 3 not used. 4 not used. 5 for VERSUS type. 6 not used. 7 for NORMAL type. 8 for DATABASE type. 9 for OHM units. 10 for REFLECT units.
<acquisition mode>	2 for SEQUENTIAL mode.
<coupling>	0 for AC coupling.
<x units> **<y units>**	0 for UNKNOWN units. 1 for VOLT units. 2 for SECOND units. 3 for CONSTANT units. 4 for AMP units. 5 for DECIBEL units. 6 for HIT units. 7 for PERCENT units. 8 for WATT units.
	See Table 47 on page 349 for descriptions of all the waveform preamble elements.
BASIC Image Specifiers	# is an BASIC image specifier that suppresses the automatic output of the EOL sequence following the last output item. K is an BASIC image specifier that outputs a number or string in standard form with no leading or trailing blanks.
Query	`:WAVeform:PREamble?`
	The query outputs a waveform preamble to the computer from the waveform source, which can be a waveform memory or channel buffer.
Returned Format	`[:WAVeform:PREamble] <preamble_data><NL>`
Example	This example outputs the current waveform preamble for the selected source to the string variable, Preamble$.

```
10 DIM Preamble$[250]!Dimension variable
20 OUTPUT 707;":SYSTEM:HEADER OFF"!Response headers off
30 OUTPUT 707;":WAVEFORM:PREAMBLE?"
40 ENTER 707 USING "-K";Preamble$
50 END
```

-K is an BASIC image specifier that places the block data in a string, including carriage returns and line feeds, until EOI is true, or when the dimensioned length of the string is reached.

See Also	`WAVeform:DATA`

Table 47   Waveform Preamble Elements (Sheet 1 of 2)

Element	Description
Format	The format value describes the data transmission mode for waveform data output. This command controls how the data is formatted when it is sent from the analyzer. (See WAVeform:FORMat.)
Type	This value describes how the waveform was acquired. (See also WAVeform:TYPE.)
Points	The number of data points or data pairs contained in the waveform data. (See ACQuire:POINts.)
Count	For the AVERAGE waveform type, the count value is the minimum count or fewest number of hits for all time buckets. This value may be less than or equal to the value requested with the ACQuire:COUNt command. For NORMAL, RAW, INTERPOLATE, and VERSUS waveform types, this value is 0 or 1. The count value is ignored when it is sent to the analyzer in the preamble. (See WAVeform:TYPE and ACQuire:COUNt.)
X increment	The X increment is the duration between data points on the X axis. For time domain signals, this is the time between points. (See WAVeform:XINCrement.)
X Origin	The X origin is the X-axis value of the first data point in the data record. For time domain signals, it is the time of the first point. This value is treated as a double precision 64-bit floating point number. (See WAVeform:XORigin.)
X Reference	The X reference is the data point associated with the X origin. It is at this data point that the X origin is defined. In this analyzer, the value is always zero. (See WAVeform:XREFerence.)
Y Increment	The Y increment is the duration between Y-axis levels. For voltage waveforms, it is the voltage corresponding to one level. (See WAVeform:YINCrement.)
Y Origin	The Y origin is the Y-axis value at level zero. For voltage signals, it is the voltage at level zero. (See WAVeform:YORigin.)
Y Reference	The Y reference is the level associated with the Y origin. It is at this level that the Y origin is defined. In this analyzer, this value is always zero. (See WAVeform:YREFerence.)
Coupling	The input coupling of the waveform. The coupling value is ignored when sent to the analyzer in the preamble.
X Display Range	The X display range is the X-axis duration of the waveform that is displayed. For time domain signals, it is the duration of time across the display. (See WAVeform:XRANge.)
X Display Origin	The X display origin is the X-axis value at the left edge of the display. For time domain signals, it is the time at the start of the display. This value is treated as a double precision 64-bit floating point number. (See WAVeform:XDISplay.)
Y Display Range	The Y display range is the Y-axis duration of the waveform which is displayed. For voltage waveforms, it is the amount of voltage across the display. (See WAVeform:YRANge.)
Y Display Origin	(See WAVeform:YDISplay.)
Date	The date that the waveform was acquired or created.
Time	The time that the waveform was acquired or created.
Frame Model #	The model number of the frame that acquired or created this waveform. The frame model number is ignored when it is sent to an analyzer in the preamble.
Acquisition Mode	The acquisition sampling mode of the waveform.
Complete	The complete value is the percent of time buckets that are complete. The complete value is ignored when it is sent to the analyzer in the preamble. (See WAVeform:COMPlete.)
X Units	The X-axis units of the waveform. (See WAVeform:XUNits.)
Y Units	The Y-axis units of the waveform. (See WAVeform:YUNits.)

Table 47   Waveform Preamble Elements (Sheet 2 of 2)

Element	Description
Band Pass	The band pass consists of two values that are an estimation of the maximum and minimum bandwidth limits of the source signal. The bandwidth limit is computed as a function of the selected coupling and filter mode. (See the WAVeform:BANDpass query.)

SOURce

Command   `:WAVeform:SOURce {WMEMory<N> | FUNCtion<N> | CHANnel<N> | HISTogram | RESPonse<N> | CGRade}`

Selects a channel, function, TDR response, waveform memory, histogram, or color grade/gray scale as the waveform source. If the waveform source is set to CGRade, the default source is the first database signal displayed. To set the CGRade source you must use the :WAVeform:SOURce:CGRade command. TDR responses are valid sources for waveform queries only if the current settings for channel bandwidth, record length, and timebase match the settings valid during the TDR normalization procedure. In the case of a mismatch, the TDR response is not displayed and queries such as :WAV:POINTS? will return an error message indicating that the "source is not valid". Histogram data sources require long format.

<N>   An integer, 1 through 4.

Example   This example selects channel 1 as the waveform source.
`10 OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"`

Query   `:WAVeform:SOURce?`

The query returns the currently selected waveform source.

Returned Format   `[:WAVeform:SOURce] {WMEMory<N> | FUNCtion<N> | RESPonse<N> | CHANnel<N> | HISTogram | CGRade}<NL>`

Example   This example places the current selection for the waveform source in the string variable, Selection$.
`10 DIM Selection$[50]!Dimension variable`
`20 OUTPUT 707;":WAVEFORM:SOURCE?"`
`30 ENTER 707;Selection$`

SOURce:CGRade

Command   `:WAVeform:SOURce:CGRade {CHANnel<N> | FUNCtion<N> | CGMemory}`

Sets the color grade source for waveform commands. The default is the first displayed database signal.

CHANnel<N>   Corresponds to the channel databases.

FUNCtion<N>   Corresponds to the function databases.

<N>   An integer, 1 through 4.

Example   The following example sets the channel 1 database as the CGRade source.
`:WAVeform:SOURce:CGRade CHAN1`
`:WAVeform:SOURce CGRade`

The CGRade parameter in the second command corresponds to the channel 1 database.

Query   `:WAVeform:SOURce:CGRade?`

The query returns the current color grade source.

Returned Format	`[:WAVeform:SOURce:CGRade] {CHANnel<N>	FUNCtion<N>	CGMemory}<NL>`
Example	The following example gets the current color grade source and store the value in the string array, setting.

```
write_IO (":WAVeform:SOURce:CGRade?");
read_IO (Setting, SETTING_SIZE);
```

TYPE?

Query	`:WAVeform:TYPE?`

Returns the current acquisition data type for the currently selected source. The type returned describes how the waveform was acquired. The waveform type may be NORMAL, RAW, INTERPOLATE, AVERAGE, or VERSUS.

NORMAL data consists of the last data point in each time bucket. RAW data consists of one data point in each time bucket with no interpolation. In the INTERPOLATE acquisition type, the last data point in each time bucket is stored, and additional data points are filled in between the acquired data points by interpolation. AVERAGE data consists of the average of the first *n* hits in a time bucket, where *n* is the value in the count portion of the preamble. Time buckets that have fewer than *n* hits return the average of the data they contain. VERSUS data consists of two arrays of data: one containing the X-axis values, and the other containing the Y-axis values. Versus waveforms can be generated using the FUNCtion subsystem commands.

Returned Format	`[:WAVeform:TYPE] {NORMal	RAW	INTerpolate	AVERage	VERSus}<NL>`
Example	`10 OUTPUT 707;":WAVEFORM:TYPE?"`

XDISplay?

Query	`:WAVeform:XDISplay?`

Returns the X-axis value at the left edge of the display. For time domain signals, it is the time at the start of the display. For VERSus type waveforms, it is the value at the center of the X-axis of the display. This value is treated as a double precision 64-bit floating point number. If you have zoomed the display to show a portion of the waveform, the query returns the value corresponding to the zoomed waveform rather than the entire waveform.

Returned Format	`[:WAVeform:XDISplay] <value><NL>`
<value>	A real number representing the X-axis value at the left edge of the display.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":WAVEFORM:XDISPLAY?"`

XINCrement?

Query	`:WAVeform:XINCrement?`

Returns the duration between data points on the X axis. For time domain signals, this is the time difference between consecutive data points for the currently specified waveform source. For VERSus type waveforms, this is the duration between levels on the X axis. For voltage waveforms, this is the voltage corresponding to one level.

Returned Format	`[:WAVeform:XINCrement] <value><NL>`
<value>	A real number representing the duration between data points on the X axis.
Example	`10 OUTPUT 707;":SYSTEM:HEADER OFF"` `20 OUTPUT 707;":WAVEFORM:XINCREMENT?"`

See Also   You can obtain the Xincrement value through the WAVeform:PREamble query.

XORigin?

Query   :WAVeform:XORigin?

Returns the X-axis value of the first data point in the data record for the currently specified source .
For time domain signals, it is the time of the first point. For VERSus type waveforms, it is the X-axis
value at level zero. For voltage waveforms, it is the voltage at level zero. The value returned by this
query is treated as a double precision 64-bit floating point number.

Returned Format   [:WAVeform:XORigin] <value><NL>

<value> is a real number representing the X-axis value of the first data point in the data record.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
               20 OUTPUT 707;":WAVEFORM:XORIGIN?"

See Also   You can obtain the Xorigin value through the WAVeform:PREamble query.

XRANge?

Query   :WAVeform:XRANge?

Returns the X-axis duration of the displayed waveform. For time domain signals, it is the duration of
the time across the display. For VERSus type waveforms, it is the duration of the waveform that is
displayed on the X axis. If you have zoomed the display to show a portion of the waveform, the query
returns the value corresponding to the zoomed waveform rather than the entire waveform.

Returned Format   [:WAVeform:XRANge] <value><NL>

<value>   A real number representing the X-axis duration of the displayed waveform.

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
               20 OUTPUT 707;":WAVEFORM:XRANGE?"

XREFerence?

Query   :WAVeform:XREFerence?

Returns the data point or level associated with the Xorigin data value for the currently specified
source. It is at this data point or level that the X origin is defined. In this analyzer, the value is always
zero.

Returned Format   [:WAVeform:XREFerence] 0<NL>

Example   10 OUTPUT 707;":SYSTEM:HEADER OFF"
               20 OUTPUT 707;":WAVEFORM:XREFERENCE?"

See Also   You can obtain the Xreference value through the WAVeform:PREamble query.

XUNits?

Query   :WAVeform:XUNits?

Returns the X-axis units of the currently selected waveform source. The currently selected source
may be a channel, function, or waveform memory.

Returned Format   [:WAVeform:XUNits] {UNKNown | VOLT | SECond | CONStant | AMP | DECibels}<NL>

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":WAVEFORM:XUNITS?"
```

YDISplay?

Query   `:WAVeform:YDISplay?`

Rreturns the Y-axis value at the center of the display, in the units of the current waveform source.

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

<value>   A real number representing the Y-axis value at the center of the display.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":WAVEFORM:YDISPLAY?"
```

YINCrement?

Query   `:WAVeform:YINCrement?`

Returns the duration between the Y-axis levels for the currently specified source.

- For BYTE and WORD data, it is the value corresponding to one level increment in terms of waveform units.
- For ASCII data format, the YINCrement is the full range covered by the A/D converter.

Returned Format   `[:WAVeform:YINCrement] <real_value><NL>`

<real_value>   A real number in exponential (NR3) format.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":WAVEFORM:YINCREMENT?"
```

See Also   You can obtain the Yincrement value through the WAVeform:PREamble query.

YORigin?

Query   `:WAVeform:YORigin?`

Returns the Y-axis value at level zero.

- For BYTE and WORD data, and voltage signals, it is the voltage at level zero.
- For ASCII data format, the YORigin is the Y-axis value at the center of the data range. Data range is returned in the Y increment.

Returned Format   `[:WAVeform:YORigin] <real_value><NL>`

<real_value>   A real number in exponential (NR3) format.

Example
```
10 OUTPUT 707;":SYSTEM:HEADER OFF"
20 OUTPUT 707;":WAVEFORM:YORIGIN?"
```

See Also   You can obtain the YORigin value through the WAVeform:PREamble query.

YRANge?

Query   `:WAVeform:YRANge?`

Returns the range of Y values (in terms of waveform units) across the entire display.

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

‹value›	A real number representing the Y-axis duration of the displayed waveform.
Example	10 OUTPUT 707;":SYSTEM:HEADER OFF" 20 OUTPUT 707;":WAVEFORM:YRANGE?"

YREFerence?

Query	:WAVeform:YREFerence?
	Returns the level associated with the Y origin for the currently specified source. It is at this level that the Y origin is defined. In this analyzer, the value is always zero.
Returned Format	[:WAVeform:YREFerence] <integer_value><NL>
‹integer_value›	Always 0.
Example	10 OUTPUT 707;":SYSTEM:HEADER OFF" 20 OUTPUT 707;":WAVEFORM:YREFERENCE?"
See Also	You can obtain the YReference value through the WAVeform:PREamble query.

YUNits?

Query	:WAVeform:YUNits?
	Returns the Y-axis units of the currently selected waveform source which can be a channel, function, waveform memory, TDR response, or color grade/gray scale data.
Returned Format	[:WAVeform:YUNits] {UNKNown	VOLT	OHM	SECond	REFLect	CONStant	AMP	WATT}<NL>
Example	10 DIM Unit$[50]!Dimension variable 20 OUTPUT 707;":WAVEFORM:YUNITS?" 30 ENTER 707;Unit$

27 Waveform Memory Commands

The Waveform Memory Subsystem commands allow you to save and display waveforms, memories, and functions. In Waveform Memory commands, the <N> in WMEMory<N> represents the waveform memory number (1-4).

DISPlay

Command   :WMEMory<N>:DISPlay {{ON|1}|{OFF|0}}

Enables or disables the viewing of the selected waveform memory. <N> is the memory number is an integer from 1 to 4.

NOTE   This command operates on waveform data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Query   :WMEMory<N>:DISPlay?

Returns the state of the selected waveform memory.

Returned Format   [:WMEMory<N>:DISPlay] {1 | 0}<NL>

Example   10 OUTPUT 707;":WMEMORY1:DISPLAY ON"

LOAD

Command   :WMEMory<N>:LOAD <file_name>

Loads an 86100C/D waveform memory location with a waveform from a file which has an internal waveform format (extension .wfm) or a verbose/yvalues waveform format (extension .txt). You can load the file either from the D:\ drive (C drive on 86100A/B instruments) or A:\ drive. See the

examples below. The scope assumes the default path for waveforms is D:\User Files\Waveforms. To use a different path, please specify the path and file name completely. <N> is the memory number is an integer from 1 to 4. <file_name> specifies the file to load, and has either a .wfm or .txt extension.

NOTE   This command operates on waveform data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Examples   This example loads waveform memory 4 with a file that has the internal waveform format.
`10 OUTPUT 707;":WMEMORY4:LOAD ""D:\User Files\Waveforms\waveform.wfm"""`
This example loads waveform memory 3 with a file on the floppy drive that has the internal waveform format.
`10 OUTPUT 707;":WMEMORY3:LOAD ""a:\waveform.wfm"""`

Related Commands   `DISK:LOAD, DISK:STORe`

SAVE

Command   `:WMEMory<N>:SAVE {CHANnel<N> | WMEMory<N> | FUNCtion<N> | RESPonse<N>}`
Stores the specified channel, waveform memory, TDR response, or function to the waveform memory. The channel or function must be displayed (DISPlay set to ON) or an error status is returned. You can save waveforms to waveform memories whether the waveform memory is displayed or not. <N> is the memory number is an integer from 1 to 4.

NOTE   This command operates on waveform data which is not compatible with Jitter Mode. Do not use this command in Jitter Mode. It generates a "Settings conflict" error.

Example   This example saves channel 1 to waveform memory 4.
`10 OUTPUT 707;":WMEMORY4:SAVE chan1"`

XOFFset

Command   `:WMEMory<N>:XOFFset <offset_value>`
Sets the x-axis, horizontal position for the selected waveform memory's display scale. Position is referenced to center screen. <N> is the memory number is an integer from 1 to 4. <offset_value> is the horizontal offset (position) value.

Query   `:WMEMory<N>:XOFFset?`
Returns the current x-axis, horizontal position for the selected waveform memory.

Returned Format   `[:WMEMory<N>:XOFFset] <offset_value><NL>`

Example   This example sets the x-axis, horizontal position for waveform memory 3 to 0.1 seconds (100 ms).
`10 OUTPUT 707;":WMEMORY3:XOFFSET 0.1"`

XRANge

Command   `:WMEMory<N>:XRANge <range_value>`
Sets the x-axis, horizontal range for the selected waveform memory's display scale. The horizontal scale is the horizontal range divided by 10. <N> is the memory number is an integer from 1 to 4. <range_value> is the horizontal range value.

Query	`:WMEMory<N>:XRANge?`
	The query returns the current x-axis, horizontal range for the selected waveform memory.
Returned Format	`[:WMEMory<N>:XRANge] <range_value><NL>`
Example	This example sets the x-axis, horizontal range of waveform memory 2 to 435 ms.
	`10 OUTPUT 707;":WMEMORY2:XRANGE 435E-6"`

YOFFset

Command	`:WMEMory<N>:YOFFset <offset_value>`
	Sets the y-axis (vertical axis) offset for the selected waveform memory. <N> is the memory number is an integer from 1 to 4. <offset_value> is the vertical offset value.
Query	`:WMEMory<N>:YOFFset?`
	Returns the current y-axis (vertical) offset for the selected waveform memory.
Returned Format	`[:WMEMory<N>:YOFFset] <offset_value><NL>`
Example	This example sets the y-axis (vertical) offset of waveform memory 2 to 0.2V.
	`10 OUTPUT 707;":WMEMORY2:YOFFSET 0.2"`

YRANge

Command	`:WMEMory<N>:YRANge <range_value>`
	Sets the y-axis, vertical range for the selected memory. The vertical scale is the vertical range divided by 8. <N> is the memory number is an integer from 1 to 4. <range_value> is the vertical range value.
Query	`:WMEMory<N>:YRANge?`
	Returns the Y-axis, vertical range for the selected memory.
Returned Format	`[:WMEMory<N>:YRANge] <range_value><NL>`
Example	This example sets the y-axis (vertical) range of waveform memory 3 to 0.2 volts.
	`10 OUTPUT 707;":WMEMORY3:YRANGE 0.2"`

Index

Symbols

*CLS, 91
*ESE, 91
*ESR, 92
*IDN, 93
*LRN, 93
*OPC, 94
*OPT?, 94
*RCL, 95
*RST, 95
*SAV, 99
*SRE, 99
*STB?, 100
*TRG, 101
*TST, 101
*WAI, 101

Numerics

86115D, 142
86115D Option 004, 142

A

aborting a digitize operation, 20, 38
acquire commands
  AREA, 128
  AVERage, 123
  BEST, 124
  COUNt, 124
  EYELine, 124
  IMAGe, 128
  LTESt, 124
  POINts, 125
  RESet, 129
  RUNTil, 126
  SSCReen, 127
  SWAVeform, 128
acquired data
  distribution, 199
  flow, 11
acquisition
  points, 125
  record length, 125
acquisition event register, 34
acquisition limits event enable
  register), 104
acquisition limits event register, 104
ADD, 188
adding parameters, 15

address, instrument default, 37
advisory line, reading and writing to, 117
AEEN, 104
AER, 34
ALERT?, 104
ALIGn, 220
AMARgin, 220, 221
AMEThod, 221
AMPLitude, 241, 249
ANALysis, 241, 265
analyzer, default address, 37
ANNotation, 248
AOPTimize, 221
APOWer, 248
  CORRection, 248
AREA, 128, 195, 208, 230
ARELock, 154
Arm Event Register, ARM bit, 100
arming the trigger, 38
ASCII
  and FORMat, 346
  linefeed, 14
ATTenuation, 148, 335
attenuation factor, probe, 145
AUTO, 139, 159
auto skew, 22
AUTodetect, 269, 278, 331, 336, 337,
  338, 339
AUTomatic, 159, 306
AUToscale, 105
AVERage, 123, 279
AXIS, 200

B

BANDpass?, 344
BANDwidth, 141, 142, 305
bandwidth limit, 344
BASIC image specifier, 345
BATHtub, 178, 182
BER, 220
BERFloor?, 277
BERLimit?, 277
BEST, 124
BFILe?, 165
bit definitions, status reporting, 32
BITRate, 249
BITS?, 242, 263
BLANk, 106
block data, 17
BORDer, 201
BRATe, 329, 335
buffer, output, 17

bus
  activity, halting, 38
  commands, 38
  management issues, 37
BWLimit, 336
BWMode, 305
BYTE and FORMat, 346
byte order, 17
BYTeorder, 344
  and DATA, 346

C

CALCulate, 221
CALibrate, 145, 313, 322
calibration
  mainframe, 131
  module, 132
  probe, 133
  procedure, 132
  status, 140
calibration commands
  AUTO, 139
  CANCel, 133
  CONTinue, 133
  DLEVel?, 133
  ERATio, 133
  FRAMe, 134
  LABel, 134
  LRESistance, 135
  MODule, 135
  OCONversion?, 135
  OPOWer, 135
  OPTical, 136
  OUTPut, 137
  OWAVelength, 136
  PROBe, 138
  RECommend?, 138
  SAMPlers, 139
  SDONe?, 139
  SKEW, 139
  STARt, 134, 135
  STATus?, 134, 136, 140
  TIME?, 135, 137
  VERTical, 137
CANCel, 133, 154, 322
CDIRectory, 165
CDISplay, 106
center screen voltage, 144
CFRequency?, 155
CGRade, 175, 249, 350
CGRade as Waveform Source, 345

histogram commands
  AXIS, 200
  BORDer, 201
  DEFault, 201
  MODE, 200
  SCALe, 200
  SIZE, 200
  SOURce, 201
  source data, 199, 345
  WINDow, 201
  X1Position, 201
  X2Position, 202
  Y1Position, 202
  Y2Position, 202
HITS?, 223, 259
HORizontal, 189, 291, 299, 323
  POSition, 189
horizontal
  functions, controlling, 329
  offset, and XOFFset, 356
  range, and XRANge, 356
  scaling and functions, 188
HPOLarity, 320
HRATio, 221
hue, 182
HYSTeresis, 337

I

*IDN? (Identification Number), 93
IEEE
  definitions for interface, 37
  standard, 8
  standard status data structure
    model, 26
IMAGe, 128, 196, 209, 230
image specifiers
  and DATA, 345
  and PREamble, 348
infinity, 16
infinity representation, 16
initialization, 19
  event status, 26
INPut, 156
input buffer, clearing, 38
instrument
  address, 37
  default address, 37
  status, 37
integer definition, 15
intensity, 177
interface
  clear (IFC), 38
  functions, 37
  initializing, 19
  select code, 37
interrupted query, 17
INVert, 190
inverting functions, 190
ISI?, 241, 266
ISIVsbit?, 242

J

JEE, 108
JER?, 109
JITTer, 204, 253
jitter event enable register, 108
jitter event register, 109
Jitter mode
  unavailable commands, 39

L

LABel, 134, 180
LAGGing, 243
LATest?, 263
LAYout, 179, 183, 185
LBANdwidth, 156
LBWMode, 157
LCL, 34
LEADing, 243
LER?, 109
LEVel, 184, 337
LEVel, in TRIGger, 337
LEVel?, 266
LEVels?, 175
LFEQualizer, 305
limit test commands
  AREA, 208
  FAIL, 203
  IMAGe, 209
  JITTer, 204
  LLIMit, 204
  MNFound, 204
  RESet, 210
  RUNTil, 205
  SELect, 204, 205
  SINTegrity, 205
  SOURce, 206
  SSCReen, 206
  SSUMmary, 209
  SWAVeform, 209
  TEST, 210
  ULIMit, 211
limit test event enable register, 110
limit test event register, 35, 110
linear feedforward equalizer, 304
linefeed, 14
list of error messages, 42
listener
  code and capability, 38
  unaddressing all, 38
LLIMit, 204
LOAD, 167, 168, 224, 355
load resistance, 135
local event register, 34, 109
LOCation, 244
locked status, querying, 151
LOCKed?, 157
long form commands, 14
LONGform, 120

lowercase letters, 14
LOWest?, 243
LRESistance, 135
*LRN (Learn), 93
LSBFirst, and BYTeorder, 344
LSELect, 158
LTEE, 110
LTER, 35
LTER?, 110
LTESt, 124

M

M1S?, 259
M2S?, 260
M3S?, 260
MAGGraph, 292, 299
MAGNify, 190
making measurements, 239
managing bus issues, 37
MARKer, 290, 300
marker commands
  PROPagation, 213
  REACtance?, 214
  REFerence, 214
  RPANnotation, 214
  STATe, 214
  X1Position, 215
  X1Y1source, 215
  X2Position, 215
  X2Y2source, 216
  XDELta?, 216
  XUNits?, 216
  Y1Position, 216
  YDELta?, 217
  YUNits, 217
MASK, 224

Y

Z

X