Programmer's Guide

HP 83480A and HP 54750A

HP part number: 83480-90015 and 54750-97014
Printed in USA December 1995

Notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material.

1400 Fountaingrove Parkway, Santa Rosa, CA 95403-1799, USA

Safety Notes

The following safety notes are used throughout this manual. Familiarize yourself with each of the notes and its meaning before operating this instrument.

WARNING

Warning denotes a hazard. It calls attention to a procedure which, if not correctly performed or adhered to, could result in injury or loss of life. Do not proceed beyond a warning note until the indicated conditions are fully understood and met.

CAUTION

Caution denotes a hazard. It calls attention to a procedure that, if not correctly performed or adhered to, would result in damage to or destruction of the instrument. Do not proceed beyond a caution sign until the indicated conditions are fully understood and met.

General Safety Considerations

WARNING

Before this instrument is switched on, make sure it has been properly grounded through the protective conductor of the ac power cable to a socket outlet provided with protective earth contact.

Any interruption of the protective (grounding) conductor, inside or outside the instrument, or disconnection of the protective earth terminal can result in personal injury.

WARNING

There are many points in the instrument which can, if contacted, cause personal injury. Be extremely careful.

Any adjustments or service procedures that require operation of the instrument with protective covers removed should be performed only by trained service personnel.

CAUTION

Before this instrument is switched on, make sure its primary power circuitry has been adapted to the voltage of the ac power source.

Failure to set the ac power input to the correct voltage could cause damage to the instrument when the ac power cable is plugged in.

Typeface conventions

(Front-Panel Key)	This represents a key physically located on the instrument.
`Softkey`	This indicates a "softkey," a key whose label is determined by the firmware of the instrument.
`Screen Text`	This indicates text displayed on the instrument's screen.

WARNING This symbol will appear along with bold print to highlight a warning.

CAUTION This symbol will appear when special care is required.

NOTE

This symbol will appear to call attention to an important point in the text.

Contents

16. Mask Test Commands

Tables

Contents

1

Programming

Programming

This chapter contains information on how to program the instrument. You can perform the following tasks by programming the instrument:

- Set up the instrument.
- Make measurements.
- Get data (waveform, measurements, configuration) from instrument.
- Send information (pixel image, configurations) to instrument.

You'll find a list of error messages and their definitions at the end of this chapter.

Default instrument address

The instrument is configured at the factory for an HP-IB address of 7. You can change this address from the front panel using the following key presses:

1. Press the (Utility) key.
2. Press `HP-IB Setup . . .` and then `Address`.
3. Use the numeric keypad to enter the new HP-IB address.
4. Press `Exit`.

IEEE 488.2

The programming instructions in this manual conform to the IEEE 488.2 Standard Digital Interface for Programmable Instrumentation. You can find additional detailed information about the IEEE 488.2 Standard in ANSI/IEEE Std 488.2-1987, *"IEEE Standard Codes, Formats, Protocols, and Common Commands."*

NOTE

The programming examples for individual commands in this manual are written in HP BASIC 5.0 for an HP 9000 Series 200/300 Controller.

Command are grouped in subsystems

In accordance with IEEE 488.2, the instrument's commands are grouped into "subsystems." Commands in each subsystem perform similar tasks. The following subsystems are provided:

Common Commands
Root Level Commands
System Commands
Acquire Commands
Calibration Commands
Channel Commands
Disk Commands
Display Commands
FFT Commands
Function Commands
Hardcopy Commands
Histogram Commands
Limit Test Commands
Marker Commands
Mask Test Commands
Measure Commands
Pixel Memory Commands
Service Commands
Timebase Commands
Trigger Commands
TriggerN Commands
Waveform Commands
Waveform Memory Commands

Interface capability

On the instrument's rear panel, next to the HP-IB connector, is a list of HP-IB interface capabilities supported by this instrument. These capabilities are defined in the following table.

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	Controller	No Capability
E2	Driver Electronics	Tri State (1 MB/SEC MAX)

Introduction

Computers communicate with the instrument by sending and receiving messages over the HP-IB bus. Programming commands are normally sent as ASCII character strings embedded inside output statements of your programming language. Input statements are used to read in responses from the instrument.

For example, HP 9000 Series 200/300 BASIC uses the **OUTPUT** statement for sending commands and queries. After a query is sent, the response is usually read using the **ENTER** statement.

Your language's output and enter statements pass the instrument address, program message, and terminator. Passing the instrument address ensures that the program message is sent to the correct interface and instrument. The default address of the instrument is 7.

The following HP BASIC OUTPUT statement sends a command that turns on the bandwidth limit of channel 1:

```
OUTPUT 707;":CHANNEL1:BWLIMIT ON"
```

Notice that the message is sent to the instrument at address 707. This indicates address 7 on an interface with select code 7 (700 / 100 = 7). The location where the device address must be specified is dependent on the programming language you are using. In some languages, this may be specified outside the output command. In HP BASIC, this is always specified after the keyword **OUTPUT**. The examples in this manual assume the instrument is at device address 707. When writing programs, the address varies according to how the bus is configured.

Also notice that in HP BASIC, the string sent to the instrument is enclosed in quotes.

When the instrument is in the remote mode, the **Remote** message is displayed on the instrument screen.

Program Message Terminator

The instructions within the program message are executed after the instruction terminator is received. The terminator may be either a New Line (NL) character, an End-Or-Identify (EOI) asserted, or a combination of the two. All three ways are equivalent. 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.

Always initialize the instrument

It is good practice to initialize the instrument at the start of every program. This ensures that the bus and all appropriate interfaces are in a known state. For example, HP BASIC provides a `CLEAR` command which clears the interface buffer:

```
CLEAR 707
```

When you are using HP-IB, `CLEAR` also resets the instrument's parser. The parser is the program that reads in the instructions that you send.

After clearing the interface, use the *RST command to initialize the instrument to a preset state:

```
OUTPUT 707;"*RST"
```

Refer to the "Common Commands" chapter for the actual commands and syntax for initializing the instrument.

Sending Commands to the Instrument

It's easy to send a command to the instrument. Simply create a command string, and place the string in your program language's output statement. For commands other than Common Commands, include a semicolon (;) before the subsystem name. For example, the following string sends the DELAY command in the Timebase subsystem.

```
OUTPUT 707;":TIMEBASE:DELAY 1US"
```

Use either 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 the long form with the short form indicated using lowercase letters.

The following is a long form of a command:

```
OUTPUT 707;":TIMEBASE:DELAY 1US"
```

And this is the short form of the same command:

```
OUTPUT 707;":TIM:DEL 1US"
```

Programs written in long form are easily read and are almost self-documenting. The short form syntax conserves the amount of controller memory needed for program storage and reduces the amount of I/O activity.

Combine commands in the same subsystem

To execute more than one function within the same subsystem, separate the commands with a semicolon (;). For example, the following two lines,

```
OUTPUT 707;":TIMEBASE:REFERENCE CENTER"
OUTPUT 707;":TIMEBASE:POSITION 0.00001"
```

can be combined into one line:

```
OUTPUT 707;":TIMEBASE:REFERENCE CENTER;POSITION 0.00001"
```

The semicolon between the REFERENCE command and the POSITION command separates the two functions. The POSITION command does not need TIMEBASE preceding it since the TIMEBASE:REFERENCE command selects the subsystem.

Combine commands from different subsystems

You can send commands and program queries from different subsystems on the same line. Simply precede the new subsytem by a semicolon (;) followed by a colon (:). In the following example, the colon before CHANNEL1 allows you to send a command from another subsystem.

```
OUTPUT 707;":TIMEBASE:REFERENCE CENTER;:CHANNEL1:OFFSET 0"
```

Sending common commands

If a subsystem has been selected and a Common Command is received by the instrument, the instrument remains in the selected subsystem. For example, if the program message:

```
":ACQUIRE:TYPE AVERAGE;*CLS;COUNT 1024"
```

is received by the instrument, the instrument sets the acquire type and count, then clears the status information without leaving the selected subsystem.

If some other type of command is received within a program message, you must reenter the original subsystem after the command. For example, the program message:

```
"ACQUIRE:TYPE AVERAGE;:AUTOSCALE;ACQUIRE:COUNT 1024"
```

sets the acquire type, completes the autoscale operation, then sets the acquire count. In this example, the ACQUIRE command must be sent again after the AUTOSCALE command in order to reenter the Acquire subsystem and set count.

Adding parameters to a command

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

```
OUTPUT 707;":HARDCOPY:DEVICE LASERJET"
```

Separate multiple parameters with a comma (,). Spaces can be added around the commas to improve readability.

```
OUTPUT 707;":FUNCTION1:MULTIPLY CHANNEL1 , WMEMORY1"
```

Duplicate command names

Identical function mnemonics can be used for more than one subsystem. For example, the command RANGE can be used to change the vertical range or to change the horizontal range:

```
OUTPUT 707;":CHANNEL1:RANGE .4"
```

sets the vertical range of channel 1 to 0.4 volts full scale.

```
OUTPUT 707;":TIMEBASE:RANGE 1"
```

sets the horizontal time base to 1 second full scale.

CHANNEL1 and TIMEBASE select the subsystem and determine which range is being modified.

You can use upper or lowercase letters

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

White space

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

Embedded strings

Embedded strings contain groups of alphanumeric characters which are treated as a unit of data by the instrument. For example, the line of text written to the advisory line of the instrument with the :SYSTEM:DSP command:

```
:SYSTEM:DSP "This is a message."
```

Embedded strings may be delimited with either single (') or double (") quotes. These strings are case-sensitive and spaces act as legal characters just like any other character.

Numbers

All numbers are expected to be strings of ASCII characters. Thus, when sending the number 9, you would send a byte representing the ASCII code for the character "9" (which is 57). A three-digit number like 102 would take up three bytes (ASCII codes 49, 48, and 50). This is taken care of automatically when you include the entire instruction in a string. The following numbers are all equal:

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

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

Suffix Multipliers

Multiplier	Mnemonic	Multiplier	Mnemonic
1E18	EX	1E-3	M
1E15	PE	1E-6	U
1E12	T	1E-9	N
1E9	G	1E-12	P
1E6	MA	1E-15	F
1E3	K	1E-18	A

Suffix Units

Unit	Referenced Unit
V	Volt
S	Second

Returning Data to the Computer

Data is requested from the instrument using a query. Queries can be used to find out how the instrument is currently configured. They are also used to get results of measurements made by the instrument, with the query actually activating the measurement. Responses are returned as uppercase letters. You can select either the long or short form responses with the SYSTEM:LONGFORM command.

Queries usually take the form of a command followed by a question mark (?). After receiving a query, the instrument places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. For example, the query:

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

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

```
ENTER 707;Range
```

passes the value across the bus to the controller and places it in the variable `Range`.

Sending another command or query, before reading the result of a query, causes the output queue to be cleared and the current response to be lost. This also generates an error in the error queue.

The output of the instrument may be numeric or character data, depending on what is queried. Refer to the specific commands for the formats and types of data returned from queries. The following example shows the data being returned to a string variable:

```
10 DIM Rang$[30]
20 OUTPUT 707;":CHANNEL1:RANGE?"
30 ENTER 707;Rang$
40 PRINT Rang$
50 END
```

After running this program, the controller displays:

```
+8.00000E-01
```

The following example shows the data being returned to a numeric variable:

```
10 OUTPUT 707;":CHANNEL1:RANGE?"
20 ENTER 707;Rang
30 PRINT Rang
40 END
```

After running this program, the computer displays:

```
.8
```

Multiple queries

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 a string variable. When you read the result of multiple queries into string variables, each response is separated by a semicolon. For example, the response of the preceding query would be:

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

Use the following program message to read the query into multiple numeric variables:

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

Sending and Receiving Binary Data

You will often need to send or receive binary data. An example is receiving binary data via the :WAVEFORM:DATA? command. This is done using the definite-length block data format. This format allows any type data to be transmitted as a series of 8-bit binary bytes. This is particularly useful for sending large quantities of data or 8-bit extended ASCII codes.

With definite-length block data format, the binary data is preceded by several bytes which describe the length of the data.

- The first byte is a pound sign (#).
- The second byte is a digit indicating how many of the next bytes describe the data length.
- The following "n" bytes give the length of the binary data.
- The remaining bytes are data.

For example, when transmitting 1000 bytes of data, send the following bytes to the instrument:

```
#41000 <1000 bytes of data>
```

The "4" indicates that the next four bytes give the number that indicates the number of data bytes, in this case 1000, being transmitted.

Refer to the examples of receiving binary data in this chapter.

Making a Measurement

Making a measurement involves changing the instrument's settings, capturing the data, and then returning the data to the computer.

Changing instrument settings

Use the AUTOSCALE command on unknown waveforms to automatically set up the vertical channel, time base, and trigger level of the instrument.

```
OUTPUT 707;":AUTOSCALE"
```

The following lines show how to manually set up the instrument for an input signal:

```
OUTPUT 707;":CHANNEL1:PROBE 10;RANGE 8E-3;OFFSET 1E-3"
OUTPUT 707;":TIMEBASE:MODE NORMAL;RANGE 1E-3;DELAY 100E-6"
```

This example sets the instrument to the following settings:

probe attenuation 10
vertical scale ..... 8 mV full-scale (100 mV/div)
center of screen ..... 100 mV
timebase ..... 1 ms full-scale (100 μs/div)
delay ..... 100 μs

This program demonstrates preparing the instrument for measurements:

```
10    CLEAR 707
20    OUTPUT 707;"*RST"
30    OUTPUT 707;":TIMEBASE:RANGE 5E-4"
40    OUTPUT 707;":TIMEBASE:DELAY 22E-9"
50    OUTPUT 707;":TIMEBASE:REFERENCE LEFT"
60    OUTPUT 707;":CHANNEL1:PROBE 10"
70    OUTPUT 707;":CHANNEL1:RANGE .8"
80    OUTPUT 707;":CHANNEL1:OFFSET -.4"
90    OUTPUT 707;":TRIGGER:LEVEL TRIG2,-.4"
100   OUTPUT 707;":TRIGGER:SLOPE POSITIVE"
110   OUTPUT 707;":ACQUIRE:TYPE NORMAL"
120   OUTPUT 707;":DISPLAY:GRATICULE FRAME"
130   END
```

Capture data with the DIGITIZE command

When the DIGITIZE command is sent to an instrument, the specified channel signal is digitized with the current ACQUIRE parameters. To obtain waveform data, you must specify the WAVEFORM parameters for the waveform data prior to sending the :WAVEFORM:DATA? query.

When the DIGITIZE process is complete, the acquisition is stopped, and 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 waveform data record and the preamble.

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. The DIGITIZE command can be sent without parameters for a higher throughput.

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;":ACQUIRE:TYPE AVERAGE"
OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"
OUTPUT 707;":WAVEFORM:FORMAT BYTE"
OUTPUT 707;":ACQUIRE:COUNT 8"
OUTPUT 707;":ACQUIRE:POINTS 500"
OUTPUT 707;":DIGITIZE CHANNEL1"
OUTPUT 707;":WAVEFORM:DATA?"
```

This setup places the instrument into the averaged mode with eight averages. This means that when the DIGITIZE command is received, the command will execute until the signal has been averaged at least eight times.

When using HP-IB, a DIGITIZE operation may be aborted by sending a Device Clear over the bus (`CLEAR 707`).

Returning the data to the computer

After receiving the :WAVEFORM:DATA? query, the instrument passes the waveform information when addressed to talk. 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.

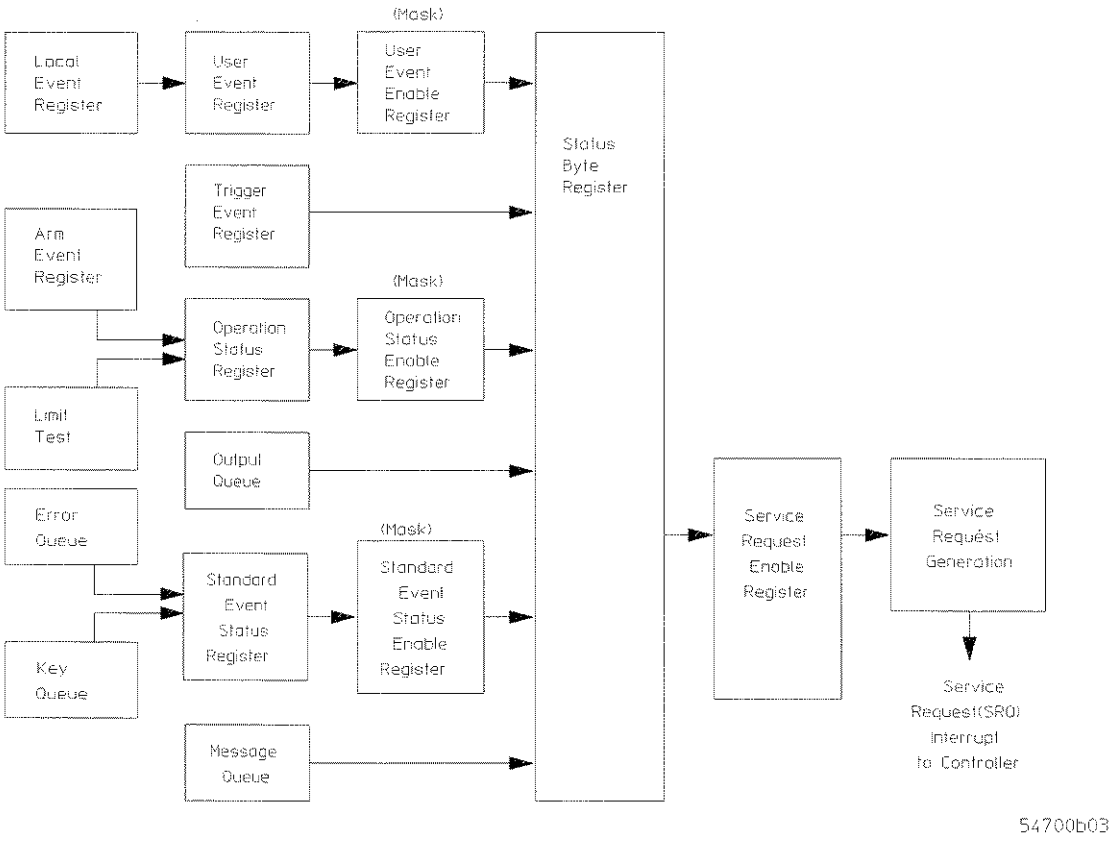
The easiest method of entering a digitized waveform depends on data structures, available formatting, and I/O capabilities. You must scale the integers to determine the voltage value of each point. These integers are passed starting with the leftmost point on the instrument's display. For more information, refer to the chapter on the Waveform subsystem.

Monitoring the Instrument

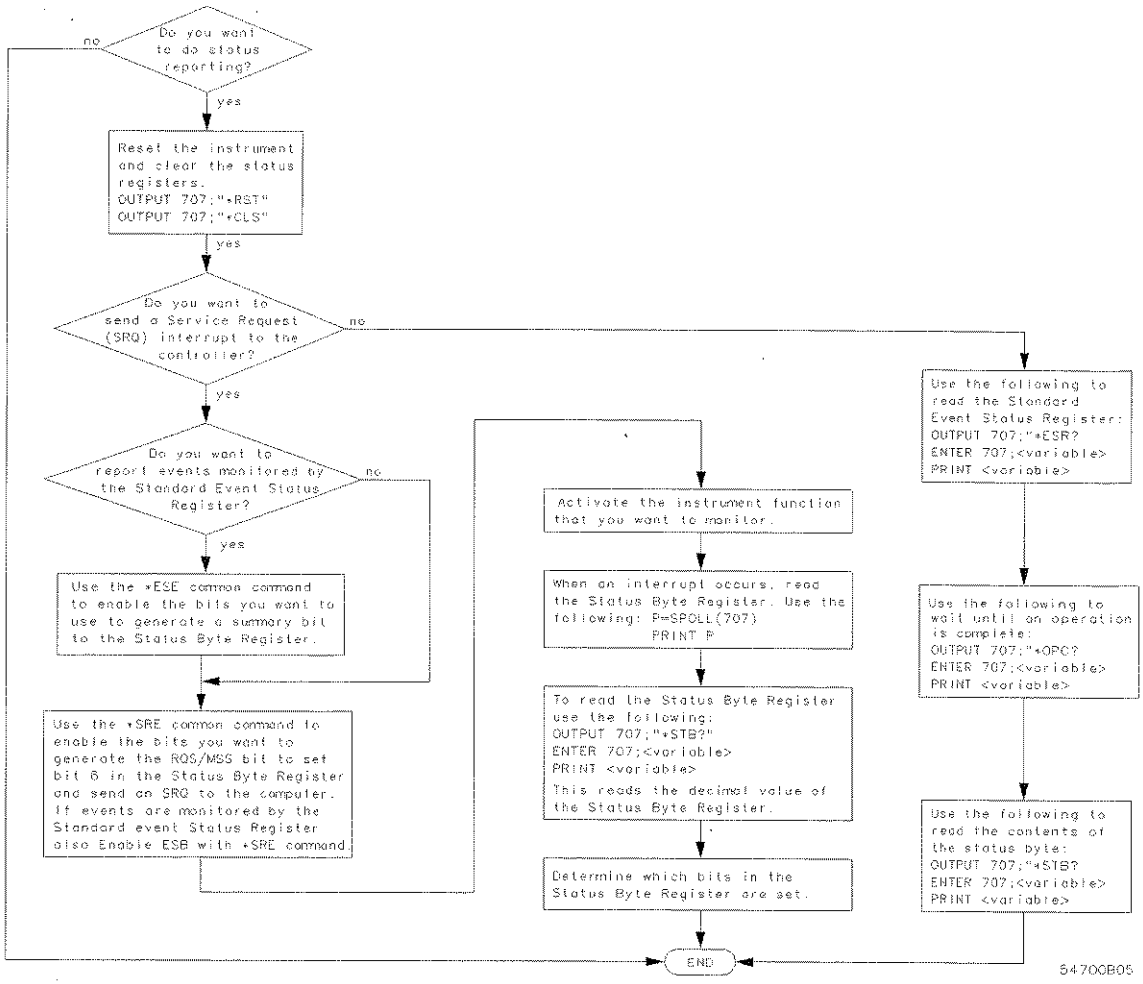
The following figure shows the instrument's status reporting structure. The status reporting structure monitors and reports events including the following:

- Status of an operation
- Availability of the measured data
- Reliability of the measured data

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. In addition, the request for the *OPC bit is also cleared.



Status Reporting Overview Block Diagram

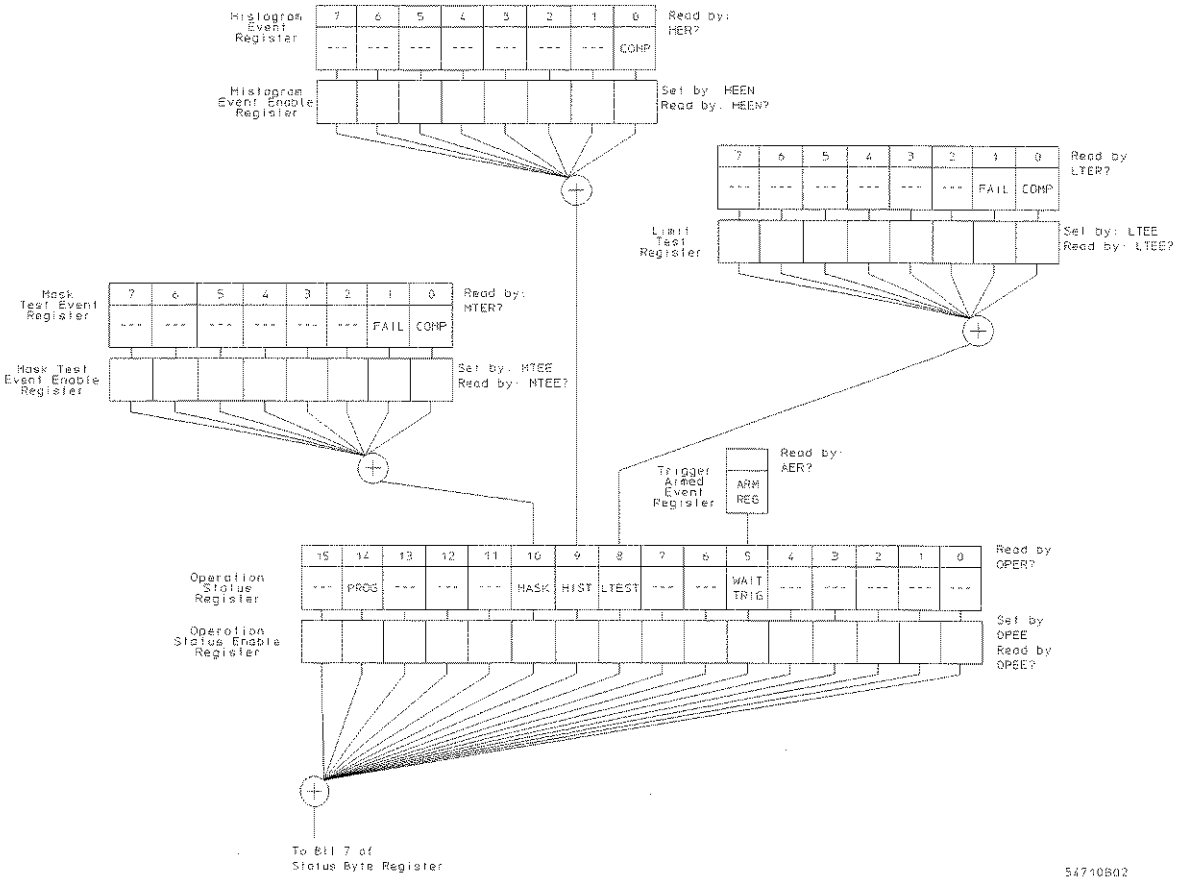


Status Reporting Decision Chart

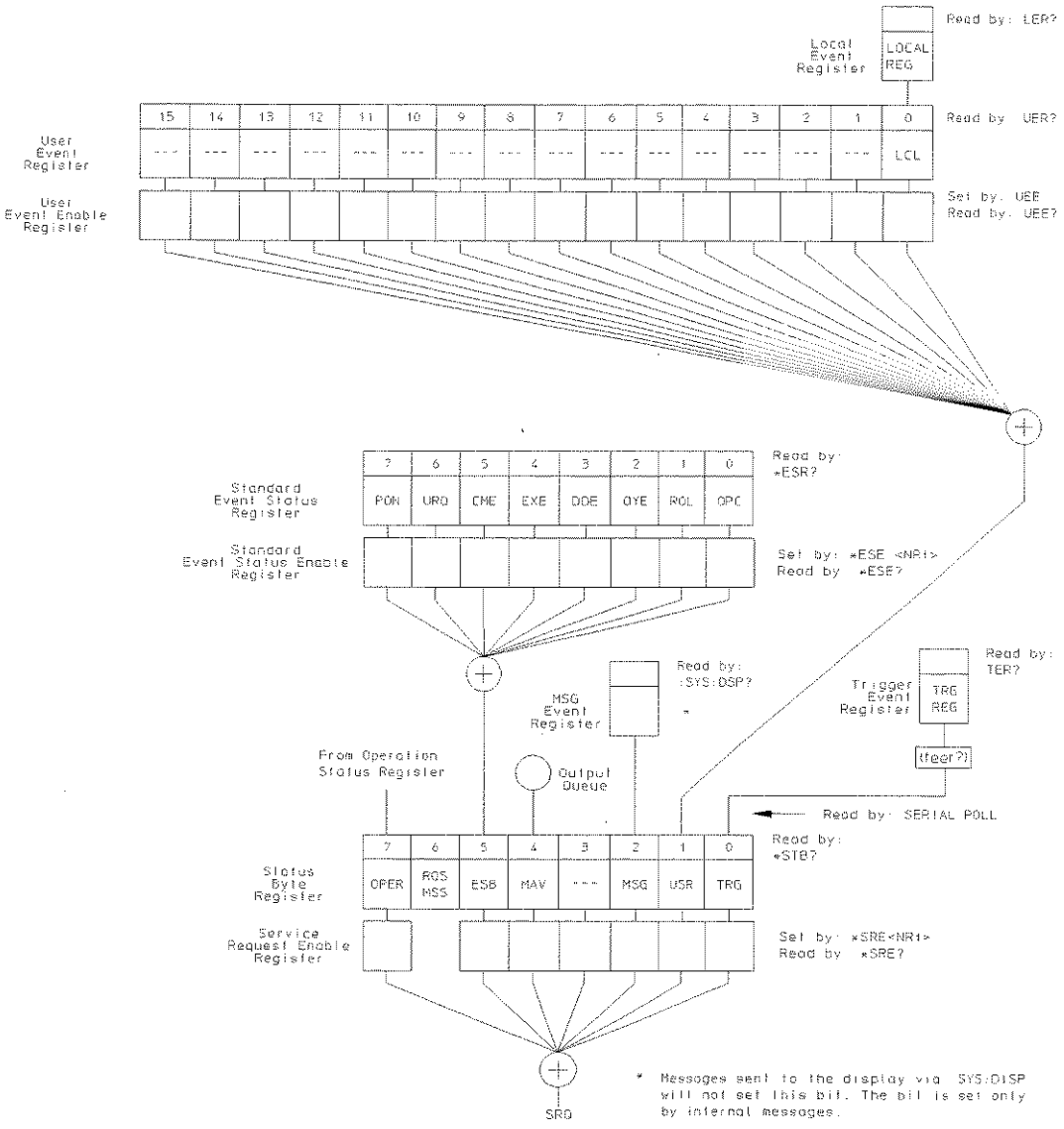
Status Reporting Registers

The following figures show the different status reporting data structures and how they work together. To make it possible 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.

To generate a service request (SRQ) interrupt to an external controller, 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.



Status Reporting Data Structures



Status Reporting Data Structures (continued)

Status Byte Register (STB)

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 or the SPOLL (HP-IB 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 SPOLL (serial poll) command reads bit 6 as the Request Service (RQS) bit and clears the bit which clears the SRQ interrupt. The *STB? command 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.

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.

No other bits in the Status Byte Register are cleared by either the *STB? query or the SPOLL (serial poll) command, except 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? command.

If bit 4 (weight = 16) and bit 5 (weight = 32) are set, the program prints the sum of the two weights. Since these bits were not enabled to generate an SRQ, bit 6 (weight = 64) is not set.

The following 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 112 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.

The following example uses the HP 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
```

NOTE

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 (SRE)

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.

The following 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 enable 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 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 program or controller operation when the trigger bit is set, then you must clear the Trigger Event Register after each time it has been set.

Standard Event Status Register (SESR)

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

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

When one of these events occur, the event sets the corresponding bit in the register. If the bits are enabled in the Standard Event Status Enable Register, the bits set in this register generate a summary bit to set bit 5 (ESB) in the Status Byte Register.

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 that are set at the present time.

The following 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
```

If bit 4 (weight = 16) and bit 5 (weight = 32) are set, the program prints the sum of the two weights.

Standard Event Status Enable Register (SESE)

To make it possible for any of the Standard Event Status Register bits to be able to generate a summary bit, first enable the bit. Enable the bit by using 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.

For example, suppose your application requires an interrupt whenever any type of error occurs. The error related 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, it 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), an SRQ, service request interrupt, is sent to the external computer.

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 to the Status Byte Register.

User Event Register (UER)

This register hosts the LCL bit (bit 0) from the Local Event 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 (User Event Enable Register) 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 (LER)

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 (OPER)

This register hosts the WAIT TRIG bit (bit 5), the LTEST bit (bit 8), the HIST bit (bit 9), the MASK bit (bit 10), and the PROG bit (bit 14).

The WAIT TRIG bit is set by the Trigger Armed Event Register and indicates that the trigger is armed.

The LTEST bit is set when a limit test fails or is completed and sets the corresponding FAIL or COMP bits in the Limit Test Event Register.

The HIST bit is set when the COMP bit is set in the Histogram Event Register, indicating that the histogram measurement has satisfied the specified completion criteria.

The MASK 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 Event Register.

The PROG bit is reserved for future use.

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.

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.

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.

The Limit Test Event Register is read and cleared with the LTER? query.

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.

Enable	Mask Value
Block COMP and FAIL	0
Enable COMP, block FAIL	1
Enable FAIL, block COMP	2
Enable COMP and FAIL	3

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:RUMode command.

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 polygon defined in the mask.

The Mask Test Event Register is read and cleared with the MTER? query.

When either the COMP or FAIL bits are set, they in turn set the MASK bit (bit 10) of the Operation Status Register. You can mask the COMP and FAIL bits, thus preventing them from setting the MASK bit, by defining a mask using the MTEE command.

Enable	Mask Value
Block COMP and FAIL	0
Enable COMP, block FAIL	1
Enable FAIL, block COMP	2
Enable COMP and FAIL	3

Histogram Event Register (HER)

Bit 0 (COMP) of the Histogram Event Register is set when the Histogram completes. The Histogram completion criteria are set by the HISTogram:RUNTil command. The Histogram Event Register is read and cleared with the HER? query.

When the COMP bit is set, it in turn sets the HIST bit (bit 9) of the Operation Status Register. Results from the Histogram Register can be masked by using the HEEN command to set the Histogram Event Enable Register to the value 0. You enable the COMP bit by setting the mask value to 1.

Arm Event Register (ARM)

This register sets bit 5 (Wait Trig bit) in the Operation Status Register and the OPER bit (bit 7) in the Status Byte Register when the instrument becomes armed.

The ARM Event Register stays set until it is cleared by reading the register with the AER? query or using the *CLS command. If your application needs to detect multiple triggers, the ARM Event Register must be cleared after each one.

If you are using the Service Request to interrupt a program or controller operation when the trigger bit is set, then you must clear the event register after each time it has been set.

Status Reporting Bit Definition

Bit	Description	Definition
PON	Power On	Indicates that a power-off-to-on transition has occured.
URQ	User Request	Indicates that a front-panel key has been pressed.
CME	Command Error	Indicates that the parser detected an error.
EXE	Execution Error	Indicates that a parameter was out of range, or inconsistent with the current settings.
DDE	Device Dependent Error	Indicates that the device was unable to complete an operation for device dependent reasons.
QYE	Query Error	Indicates that the protocol for queries has been violated.
RQL	Request Control	Indicates that the device is requesting control.
OPC	Operation Complete	Indicates that the device has completed all pending operations.
OPER	Operation Status Register	Indicates that any of the enabled conditions in the Operation Status Register have occurred.
RQS	Request Service	Indicates that the device is requesting service.
MSS	Master Summary Status	Indicates that the device has a reason for requesting service.
ESB	Event Status Bit	Indicates that any of the enabled conditions in the Standard Event Status Register have occurred.
MAV	Message Available	Indicates that there is a response in the output queue.
MSG	Message	Indicates that an advisory has been displayed.
USR	User Event Register	Indicates that any of the enabled conditions have occurred in the User Event Register.
TRG	Trigger	Indicates that a trigger has been received.
LCL	Local	Indicates that a remote-to-local transition has occured.
FAIL	Fail	Indicates that the specified test has failed.
COMP	Complete	Indicates that the specified test has completed.
LTEST	Limit Test	Indicates that one of the enabled conditions in the Limit Test Register has occurred.
MTEST	Mask Test	Indicates that one of the enabled conditions in the Mask Test Register has occurred.
HIST	Histogram	Indicates that one of the enabled conditions in the Histogram Register has occurred.
WAIT TRIG	Wait for Trigger	Indicates that the instrument is armed and ready for trigger.

Queues

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 least recent 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).

Error queue

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 items occur:

- the instrument is powered up.
- the instrument receives the *CLS common command.
- the last item is read from the error queue.
- the instrument is switched from talk only to addressed mode on the front panel.

For more information on reading the error queue, refer to the SYSTEM:ERROR? query in the System Commands chapter. For a complete list of error messages, refer to "Error Messages" in this chapter.

Output queue

The output queue stores the instrument-to-controller 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 HP Basic `ENTER` statement.

Message queue

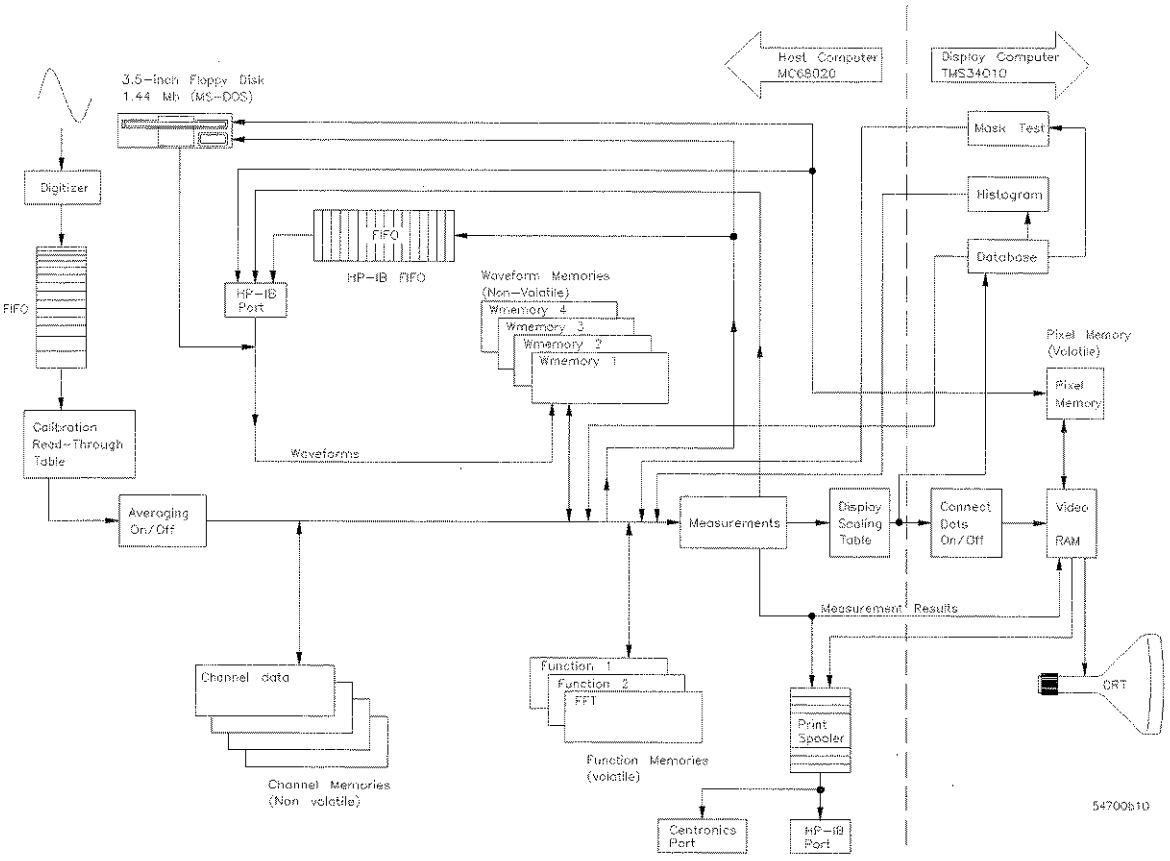
The message queue contains the text of the last message written to the advisory line on the screen of the instrument. The length of the instrument's message queue is 1. The queue is read with the SYSTEM:DSP? query. Note that messages sent with the SYSTEM:DSP command do not set the MSG status bit in the Status Byte Register.

Key queue

The key queue contains the key codes for the last 10 keys pressed on the front panel. This queue is first in, first out. If the key queue overflows, the oldest key codes are discarded as additional keys are pressed. The key queue is read with the SYSTEM:KEY? query.

Instrument Block Diagram

As the following block diagram shows, the digitizer samples the applied signal and converts it to a digital signal. The FISO holds the data until the system bus is ready for the data. The output of the FISO is raw data, and it is used as an address to the calibration read-through table (cal table).



The cal table automatically applies the calibration factors to the raw data, so that the output of the cal table is calibrated data.

Notice that averaging is turned on or off before the data is stored in the channel memories. That means once the data is acquired, if you need to turn averaging on or off before making any measurements, you must reacquire the data.

Also, you may notice that postprocessing the data includes calculating functions, storing data to the waveform memories, transferring data over the HP-IB bus, or transferring data to and from the disk.

After the measurements are performed, the data is sent through the display portion of the instrument. Notice that connected dots is a display feature, and that it has no influence on the measurement results. The pixel memory is also part of the video RAM, which is past the point where the measurements are performed on the data. Therefore, you cannot make measurements on data in the pixel memory. But, you can make measurements on data stored to the waveform memories or the color grade display.

Example Programs

The programs listed in this section are the same as those on the disk provided with this programmer's reference. The disks are provided in both LIF and DOS formats. The disks contain some additional files that are created while running the programs. To preserve the original quality of the example programs disk, make a copy of the originals and use the copy for running the programs.

The following example programs are provided in this section:

- Digitize Example
- Results? Measurement Example
- Learn String Example
- Service Request Example
- Configuration Example
- Limit Test Example
- Automated STM-16 Measurement Example
- Eye Diagram Measurement Example

Digitize Example

```
10    ! RE-SAVE "DIG_83480"
20    !
30    !         Copyright:  (c) 1994, Hewlett-Packard Co.  All rights reserved.
40    !       Contributor:  Colorado Springs Division
50    !           Product:  Example Program
60    !
70    ! $Revision:  3.0
80    ! $Date: 93/06/16 14:31:02 $      6.9.93
90    ! $Author: hmgr $    Ed Mierzejewski
100   !
110   ! Description:  DIG_83480.ibw autoscales to get a waveform on screen and
120   !               digitizes the waveform.  Then the operator can reposition
130   !               before transfering the data to the computer.  Then the
140   !               computer will draw the waveform as repositioned on the
150   !               computer screen.  It also saves the data to a record and
160   !               recalls that data before drawing it.
170   !
180   ! Main Routine:  Begin_main.
190   ! Sub routines:  none.
200   ! Sub programs:  Get_waveform, Graph, Initscope, Readme, Readme2,
210   !                Retrieve_wave, Save_waveform.
220   ! Functions:     none.
230   ! Variable List: Preamble & Waveform, @Path, & @Scope
240   !
250   !      Preamble = Real array for the first 15 parameters of the
260   !                 preamble, they are numerics and the remaining 3
270   !                 parameters are alphas and are not used.
280   !      Waveform = Integer array to store the wavefrom data.
290   !      @Path = the path for saving/recalling data to/from media.
300   !      @Scope = The scope's complete HPIB address.
310   !
320   REAL Preamble(1:15)
330   INTEGER Waveform(1:4096)
340 Begin_main:  !
350   CALL Readme
360   CALL Initscope(@Scope)
370   CALL Get_waveform(@Scope,Waveform(*),Preamble(*))
380   CALL Save_waveform(@Path,Waveform(*),Preamble(*))
390   CALL Readme2
400   CALL Retrieve_wave(@Path,Waveform(*),Preamble(*))
410   CALL Graph(Waveform(*),Preamble(*))
420   PRINT TABXY(15,30);"Program has Ended."
430   LOCAL 707
440 End_main:  !
450   END
460 Begin_subs:!
470   !
480   SUB Readme
490   ! Description:  Readme prints program explaination to the computer
500   !               screen.
510   ! Parameters: none.
520   !
530     CLEAR SCREEN
540     PRINT "DIG_83480.ibw does the following tasks:"
550     PRINT
```

```
560     PRINT "              a.  initialize interface and scope"
570     PRINT "              b.  digitize and acquire data"
580     PRINT "              c.  store data to disk"
590     PRINT "              d.  retrieve data from disk"
600     PRINT "              e.  draw signal on computer"
610     PRINT
620     PRINT "Assumed system configuration is:"
630     PRINT
640     PRINT "    HP-IB address = 7"
650     PRINT "    Scope address = 7"
660     PRINT "    signal attached to channel 1"
670     PRINT
680     PRINT "If the addresses are not correct, change the ASSIGN "
690     PRINT "statements in sub program 'Initscope'."
700     PRINT
710     PRINT "Press Continue when ready to start"
720     PAUSE
730     CLEAR SCREEN
740   SUBEND
750     !
760   SUB Readme2
770     !
780     !  Description:  Readme2 is user information and status.
790     !
800     !  Parameters; none.
810     !
820     CLEAR SCREEN
830     PRINT
840     PRINT "The waveform data and preamble information have now been"
850     PRINT "read from the scope and stored in the computer's disk."
860     PRINT
870     PRINT "When you press continue that information will be retrieved"
880     PRINT "from the disk, and plotted to the computer screen."
890     PRINT
900     PRINT "Press CONTINUE to continue."
910     PAUSE
920     CLEAR SCREEN
930   SUBEND
940     !
950   SUB Initscope(@Scope)
960     !
970     ! Description:  Initscope assigns the path to the scope, initializes
980     !               the scope, autoscales, and sets up the acquisiton
990     !               parameters.
1000    ! Parameters:
1010    !     Passed: @Scope = the HPIB address of the scope.
1020    !   Internal: @Isc = interface select code of the HPIB interface.
1030    ! Modified Variables: @Scope and @Isc
1040    !
1050    CLEAR SCREEN
1060    PRINT "INITIALIZE"
1070 Assign_paths:   !
1080    ASSIGN @Isc TO 7                  ! Interface Select Code = 7
1090    ASSIGN @Scope TO 707              ! scope address
1100 Init_sys:   !_
1110    CLEAR @Isc                        ! clear HP-IB interface
1120    OUTPUT @Scope;"*RST;*CLS"         ! set scope to default config
1130    OUTPUT @Scope;":AUTOscale"
1140    OUTPUT @Scope;":SYStem:HEADer OFF"
1150 Acq_setup:   !
1160    OUTPUT @Scope;":ACQuire;POINts 500"
```

```
1170    OUTPUT @Scope;":WAVeform:FORMat BYTE;SOURce CHANnel1"
1180    !
1190    ! Normally WORD data would be recommended because it allows better
1200    ! use of the full resolution of the scope.
1210    ! Byte data is shown because HPBasic doesn't recognize signed bytes
1220    ! and requires a conversion.  FNBcon will do the conversion.
1230    !
1240    CLEAR SCREEN
1250  SUBEND
1260    !
1270  SUB Get_waveform(@Scope,INTEGER Waveform(*),REAL Preamble(*))
1280    !
1290    ! Description:  Get_waveform digitizes the autoscaled waveform,
1300    !               gets wavefrom data and preamble after the operator
1310    !               adjusts the display to show the data as desired.
1320    !
1330    !               There are 2 forms of digitize:  1.  'with parameters'
1340    !               will digitize the specified channel/function, screen
1350    !               is blanked, then place the data in associated channel/
1360    !               function memory.  2.  'without parameters' digitizes
1370    !               all 'on' channels/functions and places data in the
1380    !               associated channel/function memory, and leaves them on
1390    !               but stopped.
1400    !
1410    !               Both digitizes are here and on adjacent lines.  One of
1420    !               the lines must be commented out or only the last one
1430    !               will be used.
1440    !
1450    ! Parameters:
1460    !   Passed:  @Scope, Waveform, Preamble
1470    !   Internal:  Digits = this is the length of the data header.
1480    !              Length = the number of bytes of data from the scope.
1490    !              End$   = empties output buffer of linefeed.
1500    !              One_char$ = used to find the '#' character.
1510    !
1520    ! Modified Variables:  Waveform, Preamble, Digits, Length, End$, and
1530    !                      One_char$.
1540    !
1550    CLEAR SCREEN
1560    PRINT "Get_waveform"
1570  !  OUTPUT @Scope;":DIGitize CHAN1"
1580    OUTPUT @Scope;":DIGitize"
1590  User_sets_disp:   !
1600    LOCAL 707
1610    PRINT "Adjust Display as you want it.  Press continue when ready."
1620    PAUSE
1630  Read_data:   !
1640    OUTPUT @Scope;":WAVeform:DATA?"
1650    ENTER @Scope USING "#,1A,";One_char$
1660    IF One_char$="#" THEN
1670      ENTER @Scope USING "#,1D";Digits
1680      ENTER @Scope USING "#,"&VAL$(Digits)&"D";Length
1690      CLEAR SCREEN
1700      PRINT
1710      PRINT "Reading ";Length;" bytes from scope"
1720       !
1730       !Redimension the array for the waveform data.  After data is
1740       !read in, one extra byte read to clear the line feed (10)
1750       !attached to the end of the scope's output buffer.
1760       !
1770      REDIM Waveform(1:Length)
```

```
1780      ENTER @Scope USING "#,B";Waveform(*)
1790      ENTER @Scope USING "-K,B";End$
1800      OUTPUT @Scope;":WAVEFORM:PREAMBLE?"
1810      ENTER @Scope;Preamble(*)
1820    ELSE
1830      PRINT "BAD DATA"
1840    END IF
1850  SUBEND
1860  !
1870  SUB Save_waveform(@Path,INTEGER Waveform(*),REAL Preamble(*))
1880  !
1890  ! Description:  Save_waveform sends acquired data and preamble to the
1900  !               computer's disk.  It is stored in 'WAVESAMPLE'.  If
1910  !               'WAVESAMPLE' already exist, it will be purged them a
1920  !               new one created.
1930  !
1940  ! Parameters:
1950  !    Passed:  @Path, Waveform, Preamble
1960  !    Internal:  none
1970  !
1980  ! Modified Variables: none
1990  ! Sub programs: Ertrap
2000  !
2010    ON ERROR CALL Ertrap
2020    CREATE BDAT "WAVESAMPLE",1,4080
2030    ASSIGN @Path TO "WAVESAMPLE"
2040    OUTPUT @Path;Waveform(*),Preamble(*)
2050  SUBEND
2060  !
2070  SUB Retrieve_wave(@Path,INTEGER Waveform(*),REAL Preamble(*))
2080  !
2090  ! Description: Retrieve_wave reads data and preamble stored in
2100  !              'WAVESAMPLE'.
2110  ! Parameters:
2120  !     Passed:  @Path, Waveform, Preamble
2130  !    Internal:  Con = indexing variable
2140  ! Functions:  FNBcon = converts from signed bytes.
2150  !
2160    ASSIGN @Path TO "WAVESAMPLE"
2170    ENTER @Path;Waveform(*),Preamble(*)
2180    FOR Con=1 TO Preamble(3)
2190      Waveform(Con)=FNBcon(Waveform(Con))
2200    NEXT Con
2210  SUBEND
2220  !
2230  SUB Graph(INTEGER Waveform(*),REAL Preamble(*))
2240  !
2250  ! Description:  Graph takes the converted data and plots it on screen.
2260  !               it uses the 'Y Display Range' to show the data as seen
2270  !               on screen vertically, and 'X Display Range' to show
2280  !               as seen horizontally (pre(14 and 12 respectively).
2290  !
2300  ! Parameters: Waveform(*) = array of data values.  Enters as q levels
2310  !                          leaves as voltages.
2320  !            Preamble(*) = the preamble for the data.
2330  !
2340  !  Internal: Vrange = preamble(14), y-axis duration of waveform displayed.
2350  !           Srange = preamble(12), x-axis duration of waveform displayed.
2360  !           Offset = preamble(15), center of screen vertically.
2370  !           vmin   = lower limit vertically.
2380  !           vmax   = upper limit vertically.
```

```
2390 !            Hmin   = lower limit horizontally (preamble(13)).
2400 !            Hmax   = upper limit horizontally.
2410 !            Hdata(*) = Horizontal values in proper units.
2420 !            Vdata(*) = Vertical values in proper units.
2430 !            I = indexing variable.
2440 !
2450 ! Modified variables: Hdata(*), Vdata(*), and I
2460 !
2470 ! Subprogram calls:V_convert and H_convert.
2480 !
2490   ALLOCATE REAL Hdata(1:Preamble(3))
2500   ALLOCATE REAL Vdata(1:Preamble(3))
2510   CALL V_convert(Waveform(*),Preamble(*),Vdata(*))
2520   CALL H_convert(Hdata(*),Preamble(*))
2530   Vrange=Preamble(14)
2540   Srange=Preamble(12)
2550   Offset=Preamble(15)
2560   Vmin=Offset-Vrange/2
2570   Vmax=Vrange/2+Offset
2580   Hmin=Preamble(13)
2590   Hmax=Hmin+Srange
2600   GCLEAR                                  !initialize graphics
2610   CLEAR SCREEN
2620   GINIT
2630   GRAPHICS ON
2640   VIEWPORT 0,130,35,100
2650   WINDOW Hmin,Hmax,Vmin,Vmax
2660   FRAME
2670   PEN 4
2680   MOVE Hdata(1),Vdata(1)
2690   FOR I=1 TO Preamble(3)                  !plot data points
2700     PLOT Hdata(I),Vdata(I),-2
2710     DRAW Hdata(I),Vdata(I)+ABS(Vmax-Vmin)*.002
2720   NEXT I
2730   PAUSE
2740   PRINT TABXY(0,18),"Vertical=";Vrange/8;" V/div";TAB(50),"Offset = ";Offset;"V"
2750   PRINT TABXY(0,19),"Time=";Srange/10;" s/div"
2760   DEALLOCATE Hdata(*)
2770   DEALLOCATE Vdata(*)
2780 SUBEND
2790 !
2800 DEF FNBcon(INTEGER B)
2810 !
2820 ! Description:  FNBcon takes the signed byte value from the scope and
2830 !               converts it to a positive integer of the proper value.
2840 ! Parameters:
2850 !    Passed: B
2860 !  Internal: Orparam = value to OR with the passed value, B, when the
2870 !                      MSB is set.
2880 ! Modified Variables:  B
2890 !
2900   Orparam=-256
2910   IF BIT(B,7) THEN B=BINIOR(Orparam,B)
2920   RETURN B
2930 FNEND
2940 !
2950 SUB Ertrap
2960 !
2970 ! Description:  Ertrap is called by an error interupt.  It checks for
2980 !               error #54 which will occur when there is a duplicate
2990 !               file name.  The existing file will be purged.
```

```
3000  ! Parameters: none
3010  !
3020    IF ERRN=54 THEN PURGE "WAVESAMPLE"
3030    OFF ERROR
3040  SUBEND
3050  !
3060  SUB V_convert(INTEGER Wav(*),REAL Pre(*),Vdata(*))
3070  !
3080  ! Description:  V_convert takes the data from the scope and converts it
3090  !               into voltage values using the equation from the manual.
3100  !
3110  ! Parameters: Wav(*) = array of data values.  Enters as q levels
3120  !                      leaves as voltages.
3130  !             Pre(*)   = the preamble for the data.
3140  !             Vdata (*) = array of vertical values, volts.
3150  !
3160  !  internal: yref = pre(10), level associated with y origin.
3170  !           yinc = pre(8), duration between y-axis levels.
3180  !           yorg = pre(9), y-axis value at level zero.
3190  !              C = indexing variable.
3200  !
3210  ! Modified variables: Vdata(*)
3220  !
3230    Yref=Pre(10)
3240    Yinc=Pre(8)
3250    Yorg=Pre(9)
3260    FOR C=1 TO Pre(3)
3270      Vdata(C)=(Wav(C)-Yref)*Yinc+Yorg
3280    NEXT C
3290  SUBEND
3300  !
3310  SUB H_convert(Hdata(*),Pre(*))
3320  !
3330  ! Description:  H_convert creates horizontal axis values using the
3340  !               equation from the manual.
3350  !
3360  ! Parameters: Hdata(*) = Horizontal values.
3370  !             Pre(*)   = the preamble for the data.
3380  !
3390  !  internal: xref = pre(7), data point associated with the x origin.
3400  !           xinc = pre(5), duration between x-axis data points.
3410  !           xorg = pre(6), x-axis value of first point in record.
3420  !
3430  ! Modified variables: Hdata(*)
3440  !
3450    Xref=Pre(7)
3460    Xinc=Pre(5)
3470    Xorg=Pre(6)
3480    FOR C=1 TO Pre(3)
3490      Hdata(C)=((C-1)-Xref)*Xinc+Xorg
3500    NEXT C
3510  SUBEND
```

Results? Measurement Example

```
10 !  RE-SAVE "RESU_83480"!Operation of SENDValid & STATistics on RESULTS?
20 !
30 !*************************************************************************!
40 !*******************  Main Program, Rev. 1.18  ********************!
50 !*************************************************************************!
60    Readme
70    Initscope(@Scope)
80    True_rep(@Scope)
90    Measure(@Scope)
100   PRINT "End of Program -- Results are on your printer."
110   BEEP 15,2
120         !*************************************************************!
130   END   !*                     End of Main Program                     *!
140         !*************************************************************!
150                                                                       !
160         !*************************************************************!
170         !*                      Begin Sub Programs                     *!
180         !*************************************************************!
190   SUB Readme
200     !*************************************************************!
210     !*This sub program writes user program information to the screen.*!
220     !*************************************************************!
230     CLEAR SCREEN
240     PRINT "This example program will measure the signal."
250     PRINT "from an HP8133A generator (or any similar signal).  "
260     PRINT
270     PRINT "It measures the Positive Pulse Width with Statistics."
280     PRINT "Then uses the *RESULTS?* to report over the HPIB."
290     PRINT
300     PRINT "The report from the RESULTS? varies depending on the status"
310     PRINT "of STATISTICS ON|OFF and SENDValid ON|OFF."
320     PRINT
330     PRINT "This program will print the results for each of the cases."
340     PRINT
350     PRINT "The program assumes that the system is configured such that:"
360     PRINT
370     PRINT "   HP-iB interface is at address 7."
380     PRINT "   Scope is at address 7."
390     PRINT "   An 83483A is installed into slots 1 & 2."
400     PRINT "   Printer at 701"
410     PRINT
420     PRINT "If these addresses are incorrect, break program and set addresses"
430     PRINT "as needed in the ASSIGN statements."
440     PRINT
450     PRINT "PRESS continue to run program"
460     PAUSE
470     CLEAR SCREEN
480   SUBEND
490     !
500   SUB Initscope(@Scope)
510     !*************************************************************!
520     !*  This sub program initializes the I/O and scope.          *!
530     !*************************************************************!
540     ASSIGN @Scope TO 707
550     CLEAR @Scope                      !clear HP-iB interface
```

```
560     OUTPUT @Scope;"*cls"
570     OUTPUT @Scope;"*RST"              !reset scope to default config
580     OUTPUT @Scope;":SYSTEM:HEADER OFF"!turn off header
590     CLEAR SCREEN
600   SUBEND
610   !
620   SUB True_rep(@Scope)
630     !*************************************************************!
640     !*  This sets up the scope to look at the signal of the 8133.   *!
660     !*************************************************************!
670     PRINT "  Connect the HP 8133A CHANNEL 1 Output to the Input  "
680     PRINT "  of the 83483A.                                      "
690     PRINT
700     PRINT "  Press continue when ready to continue.  "
710     PAUSE
720     CLEAR SCREEN
730     OUTPUT @Scope;":channel1:display on"
770     OUTPUT @Scope;":autoscale"
780     OUTPUT @Scope;":display:persistence infinite"
790     WAIT 5
800     PRINT "  The displayed waveform on the 83480A is the true "
810     PRINT "  representation of the 8133A signal."
820     PRINT
840     PRINT "       Sampling rate is 40 kSa/s,"
850     PRINT "       Analog Bandwidth is 20 GHz"
860     PRINT
870   SUBEND
880   !
890   SUB Measure(@Scope)
900     !*************************************************************!
910     !* This sub program will make a +width measurement.            *!
920     !* It will also report the mean and standard deviation.       *!
930     !*************************************************************!
940     CLEAR SCREEN
950     PRINT "  Measuring Waveform and Reporting Results.  "
960     REAL R(1:12)
970     !*************************************************************!
980     !  Normally when making measurements, they should be preceeded  !
990     !  by a DIGITIZE.  But, because I have the scope setup like I   !
1000    !  want it from previous sub programs and I will be using the   !
1010    !  statistics, I will not use the DIGITIZE command.             !
1020    !*************************************************************!
1030    OUTPUT @Scope;":measure:source channel1"
1040    PRINTER IS 701
1050    FOR C=1 TO 4
1060      OUTPUT @Scope;"run"
1070      MAT R= (0)
1080      OUTPUT @Scope;":measure:statistics ";INT(C/3)
1090      OUTPUT @Scope;":measure:sendvalid ";C MOD 2
1100      OUTPUT @Scope;":measure:statistics?"
1110      ENTER @Scope;St$
1120      OUTPUT @Scope;":measure:sendvalid?"
1130      ENTER @Scope;Sv$
1140      OUTPUT @Scope;":measure:pwidth"
1150      OUTPUT CRT;"Measuring for 20 seconds to get good stats."
1160      WAIT 20    !* Give the measurements a chance to build up values *!
1170      OUTPUT @Scope;"stop"!* See that values match ON SCREEN & OVER HPIB *!
1180      OUTPUT @Scope;":measure:results?"
1190      ENTER @Scope USING "%,K";R(*)
1200      OUTPUT CRT;"Printing Results to your printer"
1210      PRINT
```

```
1220      PRINT "Statistics is ";St$,"SendValid is ";Sv$
1230      PRINT
1240      PRINT "First value is ";R(1)
1250      PRINT "Second value is ";R(2)
1260      PRINT "Third value is ";R(3)
1270      PRINT "Fourth value is ";R(4)
1280      PRINT "Fifth value is ";R(5)
1290      PRINT "Sixth value is ";R(6)
1300      PRINT "Seventh value is ";R(7)
1310      WAIT 5
1320      CLEAR SCREEN
1330    NEXT C
1340    PRINTER IS CRT
1350  SUBEND
```

Learn String Example

```
10   !RE-SAVE "LSTG_83480"  !HP Basic for HP-IB interface, rev 2.0
20   !*********************************************************************
30   !*  This program reads and returns the learn string from and to a  *
40   !*  83480 Oscilloscope.                                            *
50   !*                      ********************                       *
60   !*                      ********************                       *
70   !*                       Begin MAIN PROGRAM                        *
80   !*********************************************************************
90   COM /Io/ @Scope,Hpib
100  Readme                        !Description of the program
110  Initscope                     !initialize interface and scope
120  Length=FNStsize               !find setup string size.
130  Get_learnstr(Length)          !save 3 configurations on disk
140  Recall_learnstr(Length)       !select & recall 1 of 3 setups
150  PRINT
160  BEEP 15,1
170  PRINT "program done"
180    !*************************************************************
190    !*                    End of Main Programs                   *
200    !*************************************************************
210  END
220    !*************************************************************
230    !*                    Begin Sub Programs                     *
240    !*************************************************************
250  SUB Initscope
260    !*********************************************************
270    !*  This sub program initializes the INTERFACE AND SCOPE  *
280    !*********************************************************
290    COM /Io/ @Scope,Hpib
300    Hpib=7
310    Scope=7
320    ASSIGN @Scope TO Hpib*100+Scope  !scope address
330    CLEAR Hpib                       !clear HPIB interface
340    OUTPUT @Scope;"*RST"             !set scope to default config
350    OUTPUT @Scope;":AUTOSCALE"       !AUTOSCALE
360    OUTPUT @Scope;":SYST:HEAD OFF"
370    OUTPUT @Scope;"*OPC?"            !wait for scope to finish auto
380    ENTER @Scope;Opc
390  SUBEND
400    !
410  SUB Readme
420    !*********************************************************************
430    !*  This sub program displays a message about the program for the    *
440    !*  user.                                                            *
450    !*********************************************************************
460    CLEAR SCREEN
470    PRINT "This sample program will prompt the user to set up the"
480    PRINT "scope in three different configurations and will store"
490    PRINT "them to the computer disk.  Any of the three configurations"
500    PRINT "may then be recalled from the disk and sent to the scope"
510    PRINT
520    PRINT "The program assumes that the system is configured such that:"
530    PRINT "      HP-IB interface is at address 7"
540    PRINT "      scope is at address 7"
550    PRINT "      a signal is attached to channel 1"
```

```
560     PRINT
570     PRINT "If these addresses are incorrect, break program and set addresses"
580     PRINT "as needed in the Initialize in the ASSIGN statements."
590     PRINT
600     PRINT "Press CONTINUE when ready to start.  Scope will first autoscale"
610     PRINT "on signal on channel 1 and will then prompt for user to setup"
620     PRINT "scope as desired before saving configurations in computer."
630     PRINT
640     PAUSE
650     CLEAR SCREEN
660   SUBEND
670   !
680   SUB Get_learnstr(Length)
690     !*************************************************************
700     !*  This sub program will get the learn string from the 83480 *
710     !*  and place it in SET$.  Then it will create a BDAT file    *
720     !*  called "JSETUPS" which holds 3 records.  If this file is  *
730     !*  already created it will be PURGED!                        *
740     !*************************************************************
750     COM /Io/ @Scope,Hpib
760     ON ERROR CALL Ertrap
770     CREATE BDAT "JSETUPS",3,Length    !create 3 files for 3 different
780                                       !setups.
790     ALLOCATE Set$[Length]             !temp variable to hold string.
800     ASSIGN @Path TO "JSETUPS"         !open file
810     FOR I=1 TO 3
820       CLEAR SCREEN
830       LOCAL @Scope
840       PRINT "PLEASE HAVE SETUP #";I;" READY AND PRESS RETURN"
850       INPUT A$
860       OUTPUT @Scope;":SYSTEM:SETUP?"   !query learnstring from scope
870       ENTER @Scope USING "-K";Set$     !read learn string from scope
880       IF Set$[1;1]="#" THEN
890         OUTPUT @Path,I;Set$            !store setup string to disk
900       ELSE
910         CLEAR SCREEN
920         PRINT "Received bad data.  No setup saved."
930       END IF
940     NEXT I
950     ASSIGN @Path TO *                                    !close file
960     DEALLOCATE Set$
970   SUBEND
980   !
990   SUB Ertrap
1000    !*********************************************************
1010    !*  The program will branch to this Error Trap if the      *
1020    !*  ON ERROR is ON and an error occurs.  It reset the      *
1030    !*  ON ERROR to OFF the return to where it was called.     *
1040    !*********************************************************
1050    IF ERRN=54 THEN !  Error 54 is Duplicate File Name
1060      PURGE "JSETUPS"
1070      OFF ERROR
1080    ELSE
1090      CLEAR SCREEN
1100      PRINT ERRM$
1110      BEEP
1120      PAUSE
1130    END IF
1140  SUBEND
1150  !
1160  SUB Recall_learnstr(Length)
```

```
1170    !
1180    !  This sub program lets the user select which of the 3 setups
1190    !  that have been stored on the disk in "JSETUPS1, 2, or 3 to
1200    !  use to setup the scope.  It will loop until the user selects
1210    !  (E) to exit.
1220    !
1230    COM /io/ @Scope,Hpib
1240    ASSIGN @Path TO "JSETUPS"               !open file
1250    ALLOCATE Set$[Length]                   !create temp variable.
1260    Done=0
1270    REPEAT
1280      CLEAR SCREEN
1290      PRINT "Please enter (1) to recall setup 1"
1300      PRINT "             (2) to recall setup 2"
1310      PRINT "             (3) to recall setup 3"
1320      PRINT "             (E) to exit"
1330      INPUT A$
1340      SELECT UPC$(A$)
1350      CASE "1","2","3"
1360        ENTER @Path,VAL(A$);Set$                !read data from disk.
1370        IF Set$[1;1]="#" THEN                   !Have good data.
1380    !
1390    ! Add command header to setup string and send entire string to scope.
1400    !
1410          OUTPUT @Scope USING "#,K";":SYSTEM:SETUP ";Set$
1420        ELSE
1430          CLEAR SCREEN
1440          PRINT "Received bad data, no string entered."
1450        END IF
1460      CASE "E"
1470        Done=1
1480      END SELECT
1490    UNTIL Done
1500    DEALLOCATE Set$
1510    ASSIGN @Path TO *
1520  SUBEND
1530  !
1540  DEF FNStsize
1550      !
1560      !The setup string size can vary, depending on operating system
1570      !revision.  Must read the header to determine the proper lengths.
1580      !The format of the data is #NX...X<setup data string>.  Then I
1590      !add 5 for the bdat file management headers.
1600      !
1610    COM /io/ @Scope,Hpib
1620    DIM Psign$[1]
1630    INTEGER Length,Cnt,L
1640    ON TIMEOUT Hpib,3 CALL Tout
1650            !Set the bus timeout so if there is no/bad data, can't find
1660            !the # sign, we will stop and let the operator know.
1670    OUTPUT @Scope;":SYSTEM:SETUP?"
1680            !Query scope for the setup string.
1690    Cnt=0
1700    REPEAT
1710      ENTER @Scope USING "#,A";Psign$
1720            !Enter a character at a time until find the # sign.  It
1730            !indicates the beginning of the block header.
1740      Cnt=Cnt+1
1750            !FN must keep track of the number of characters before the #
1760            !sign for cases where the system headers are ON.
1770    UNTIL Psign$="#"
```

```
1780    ENTER @Scope USING "#,A";Psign$
1790            !Next character tells the number of digits in the header.
1800    ENTER @Scope USING "#,"&Psign$&"D";Length
1810            !Length is the number of data values to follow before the NL.
1820    ALLOCATE Temp$[Length+1]
1830    ENTER @Scope USING "#,-K";Temp$
1840    L=7+Cnt+VAL(Psign$)+Length
1850    DEALLOCATE Temp$
1860    RETURN L
1870  FNEND
1880  !
1890  SUB Tout
1900      !Branching here says that the HPIB bus was idle for 3 seconds.
1910      !This would be caused by reaching the end of the setup data without
1920      !finding a # sign.
1930      CLEAR SCREEN
1940      PRINT "Bad Data, query aborted."
1950      BEEP
1960      PAUSE
1970  SUBEND
```

Service Request Example

```
10    !  RE-SAVE "SRQ_83480"
20    !  This program sets the Event Status Enable Register and the
30    !  Service Request Enable Register so that an error will cause
40    !  a service request.  It also saves a setup to a setup memory
50    !  and recalls that setup.
60    DIM Query$[15],Command$[15],Q$[9000]
70    COM /Err/ Hpib,Scope
80    COM /S/ @S
90    Hpib=7
100   CLEAR Hpib
110   Scope=7
120   Saddr=Hpib*100+Scope       !Sets the I/O address.
130   Readme
140   ASSIGN @S TO Saddr
150   ON INTR Hpib,15 CALL Ermsg  !Tells computer where to go on an error
160   CLEAR Saddr
170   OUTPUT @S;"*ESE 60;*SRE 32;*CLS"
180   !   *ESE XX sets the Event Status Enable Regiser.
190   !        32 => CME or Command Error
200   !        16 => EXE or Execution Error
210   !         8 => DDE or Device Dependent Error
220   !         4 => QYE or Query Error
230   !        ----------------------------------
240   !        60
250   !   *SRE XX sets the Service Request Enable Register.
260   !        32 => ESB or Event Status Bit
270   ENABLE INTR Hpib;2       !Allows the HPIB to interrupt the computer.
280   Saveset$="*SAV "         !This command is missing the parameter, 1.
290   Recallset$="*RCL 1"      !This recalls setup #1.
300   !
310   OUTPUT @S;Saveset$    !Send the command, causes CME until the 1 is added.
320   WAIT 1
330   LOCAL Saddr
340   Readme1
350   OUTPUT @S;Recallset$     !Recalls setup #1.
360   LOCAL Saddr
370   Readme2
380   END
390   !
400   SUB Ermsg               !Error Trap
410     COM /S/ @S
420     COM /Err/ Hpib,Scope
430     DIM E$[50]
440     PRINT "An error occured.",
450     Sp=SPOLL(707)
460     IF BIT(Sp,6) THEN
470     ! Testing bit 6 will tell us if the scope is the source of the interrupt.
480       OUTPUT @S;"*ESR?"!Reads then clears the Standard Event Status Register
490       ENTER @S;J
500       Srq_type(J)         !Call to the SRQ interpreter subprogram.
510       Done=0
520       REPEAT
530     !  Read the Error Queue to determine the specific error.  Repeat reading
540     !  until the Queue is empty.  Each time the queue is read the error will
550     !  be reported or the message 'THERE ARE NO MORE ERRORS' will be returned.
```

```
560         OUTPUT @S;":SYSTEM:ERROR? STRING"
570         ENTER @S;E$
580         IF E$[1;1]="0" THEN
590           PRINT "THERE ARE NO MORE ERRORS."
600           OUTPUT @S;"*CLS"   !Clears all event registers and queues except
610                              !the output queue.
620           ENABLE INTR Hpib;2
630           Done=1
640           Readme3
650         ELSE
660           PRINT E$
670         END IF
680       UNTIL Done
690     ELSE
700       CLEAR SCREEN
710       PRINT "An interrupt on the HPIB has occured but it is not the "
720       PRINT "scope.  Please clear the other source of interrupt before"
730       PRINT "restarting this program."
740       PAUSE
750     END IF
760   SUBEND
770   !
780   SUB Readme
790     PRINT "  This program sets the Event Status Enable Register and the"
800     PRINT "  Service Request Enable Register so that an error will cause"
810     PRINT "  a service request. "
820     PRINT
830     PRINT "  The second function of this program is to save and then recall"
840     PRINT "  the current scope setup to and from setup memory 1.  However,"
850     PRINT "  there is a bug in this program.  The save command is missing a"
860     PRINT "  parameter.  It needs a 1 after the space or '*SAV 1'."
870     PRINT
880     PRINT "  After you have seen how the SRQ's work you may edit line 280"
890     PRINT "  to save the setup."
900     PRINT
910     PRINT "  Once you have fixed the bug the program will run and save the"
920     PRINT "  current setup to setup memory 1 then pause and allow you to "
930     PRINT "  change the setup.  When you continue, the original setup will"
940     PRINT "  be restored."
950     PRINT
960     PRINT "  The expected configuration is;"
970     PRINT "       The scope is at address 7"
980     PRINT "       The HPIB is at address 7"
990     PRINT
1000    PRINT "  If the configuration is different, break program and set "
1010    PRINT "  the addresses as required using the variables Hpib and "
1020    PRINT "  Scope.  Then run again."
1030    PRINT
1040    PRINT "  Press Continue when ready to resume operation."
1050    PRINT
1060    PAUSE
1070    CLEAR SCREEN
1080  SUBEND
1090  !
1100  SUB Readme1
1110    PRINT
1120    PRINT "   The current setup has been saved in setup memory #1."
1130    PRINT
1140    PRINT "   Change the scope's setup from the front panel.  When you"
1150    PRINT "   press continue, the original setup will be restored."
1160    PRINT
```

Example Programs

```
1170    BEEP
1180    PAUSE
1190    CLEAR SCREEN
1200  SUBEND
1210  !
1220  SUB Readme2
1230    PRINT "   The program has ended.  Thanks for trying our Save "
1240    PRINT "   and Recall setup memories."
1250    PRINT
1260  !  PRINT "   Now you have the opportunity to edit the program"
1270  !  PRINT "   to generate an error and see how the interrupt masking"
1280  !  PRINT "   works."
1290    PRINT
1300    PRINT "          GOODBYE.     "
1310  SUBEND
1320  !
1330  SUB Readme3
1340    PRINT
1350    PRINT "  Break the program at this time and determine the cause of "
1360    PRINT "  the error."
1370    PRINT
1380    PRINT "  Once you have fixed the cause of the error described by"
1390    PRINT "  the error code and message, rerun the program by pressing"
1400    PRINT "  RUN."
1410    STOP
1420    CLEAR SCREEN
1430  SUBEND
1440  !
1450  SUB Srq_type(J)
1460  !  The scope has interrupted the computer and we have read the
1470  !  Standard Event Status Register.  Now the value, J, that was
1480  !  read by the *ESR? will be evaluated to determine why the
1490  !  SRQ was generated.
1500    PRINT "ESR value is ";J
1510    SELECT J
1520    CASE 32
1530      PRINT "32 => CME or Command Error."
1540    CASE 16
1550      PRINT "16 => EXE or Execution Error."
1560    CASE 8
1570      PRINT "8 => DDE or Device Dependent Error."
1580    CASE 4
1590      PRINT "4 => QYE or Query Error."
1600    END SELECT
1610  SUBEND
```

Configuration Example

```
10   !RE-SAVE "CONFIG"   !This is an RMB and IBasic Program.
20   !
30   !It queries the scope to determine the configuration and then
40   !prints it to the crt.  It assumes that the scope is at HPIB 7
50   !
60   DIM Mframe$[13],Slot$(1:4)[13]
70   OUTPUT 707;":SYSTEM:HEADER ON"
80   OUTPUT 707;":SYSTEM:LONGFORM ON"
90   !
100  !***** DETERMINE THE FRAME MODEL NUMBER ******!
110  !
120  OUTPUT 707;":MODEL? FRAME"
130  ENTER 707;Mframe$
140  !
150  !*****  DETERMINE THE PLUG-INS AND THEIR LOCATIONS ********!
160  !
170  FOR I=1 TO 4
180    OUTPUT 707 USING "K";":MODEL? PLUGIN";I
190    ENTER 707;Slot$(I)
200  NEXT I
210  !
220  !*****  REPORT THE MAINFRAME MODEL # AND PLUG-INS *******!
230  !
240  CLEAR SCREEN
250  PRINT "The Main frame is ";Mframe$
260  PRINT
270  PRINT "The plug-in in slot 1 is ";Slot$(1)
280  PRINT "The plug-in in slot 2 is ";Slot$(2)
290  PRINT "The plug-in in slot 3 is ";Slot$(3)
300  PRINT "The plug-in in slot 4 is ";Slot$(4)
310  PRINT
320  PRINT "End of Program"
330  END
```

Limit Test Example

```
10    ! RE-SAVE "NLIM"
20    !
30    !       Copyright: (c) 1994, Hewlett-Packard Co.  All rights reserved.
40    !     Contributor: Colorado Springs Division
50    !         Product: Throughput Application
60    !
70    ! $Revision: 1.1 $       3.0
80    ! $Date: 93/06/16 14:32:52 $          6-14-93
90    ! $Author: hmgr $         Ed Mierzejewski
100   !
110   ! Structure Chart: None
120   !     Description: This Uses Measure Limit Testing to make 3 measurements
130   !                  on 10 successive pulses at a 10 Hz rate.
140   !
150   !  Considerations: None
160   !    Main routine: Begin_main
170   !    Sub-routines: None
180   !       Functions: None
190   !    Sub-programs: Readme, Set_paths, Set_scope, Meas, Tcount
200   !
210 Variable_list:!
220 Begin_main:   !
230   CALL Readme
240   CALL Set_paths(@Scope,Isc)
250   CALL Set_scope(@Scope)
260   CALL Meas(@Scope,Isc)
270 End_of_main:  !
280   END
290   !
300 Begin_subs:  !
310   !
320   SUB Readme
330   !
340   ! Description: Readme writes instructions and information at the
350   !              beginning of the program for the user to ensure
360   !              proper setup prior to continuing the program.
370   !
380   !  Parameters: None
390   !
400     CLEAR SCREEN
410     PRINT TABXY(5,5)
420     PRINT "NLIN.ibw uses a HP8133 to generate "
430     PRINT " pulses with a 30.303 ns period, 0.1 V amplitude"
440     PRINT "(these settings can be selected by recalling the"
450     PRINT "standard settings.)"
460     PRINT
470     PRINT "    It will work with any similar signal."
480     PRINT
490     PRINT "It makes a Vpp, Risetime, and Positive pulse width measurement"
500     PRINT "on each and reports the mean after all 10 measurement sets are"
510     PRINT "complete."
520     PRINT
530     PRINT "There are NO specific plug-ins required, except a suitable one"
540     PRINT "must be in channel 1.  (Program was developed using a 83483A"
550     PRINT "installed in slot 1 of a 83480A."
```

```
560     PRINT
570     PRINT "The HPIB card is assumed to be at interface select code 7 and"
580     PRINT "the scope is at address 7."
590     PRINT
600     PRINT "If using the 8133A, "
610     PRINT "connect CHANNEL1 OUTPUT to Channel 1 Input, and"
620     PRINT "connect TRIGGER CHANNEL OUTPUT to Trigger 2 Input, then"
630     PRINT "enable CHANNEL1 OUTPUT and TRIGGER CHANNEL OUTPUT."
640     PRINT
650     PRINT "Ensure all of this is correct before continuing."
660     PRINT
670     PRINT "press continue when through reading this."
680     PAUSE
690     CLEAR SCREEN
700   SUBEND
710   !
720   SUB Set_scope(@S)
730   !
740   ! Description: Set_scope has 2 parts:
750   !              1 -- initialize the scope and i/o.
760   !              2 -- set for measurement of the pulses.
770   !
780   !  Parameters:
790   !       Passed: (@S) @Scope = specific scope's address,
800   !     Internal:
810   !
820   ! Modified Variables: None
830   !
840   ! Omit Part_1 if you already have a good instrument state.
850 Part_1:                                ! Initialize  to measure pulses.
860     OUTPUT @S;"*rst;*cls"
870     OUTPUT @S;":opee 256"                 ! Unmasks the Lim.Tst. Comp. bit.
880     OUTPUT @S;"*sre 128"                  ! Unmasks the oper bit, see Ltest.
890     OUTPUT @S;":disp:grat fram"
900     OUTPUT @S;";:blan chan2;view chan1"
910     OUTPUT @S;";:blan chan3;blan chan4"
920     OUTPUT @S;";:acq:poin 512"
930     OUTPUT @S;";:chan1:band high;disp on;offs 0;prob 1,rat;scal .1"
940     OUTPUT @S;";:autoscale" !  Get a suitable scale for measurement.
950 Part_2:                                                  ! Set Measurements
960     OUTPUT @S;";:meas:send off;stat on;sour chan1"
970     !
980     ! Turn off sendvalid, on statistics, and sets source to channel 1
990     !
1000    OUTPUT @S;";:meas:vpp;ris;pwid"
1010  SUBEND
1020  !
1030  SUB Set_paths(@Scope,Isc)
1040  !
1050  ! Description: Set_paths simply assigns HPIB select code to be 7 and
1060  !              the scope address to be 7.
1070  !
1080  !  Parameters:
1090  !    Passed:   @Scope = I/O path 707
1100  !              scope = the HPIB address that the scope is selected to.
1110  !  Internal:   Isc   = the interface select code for the HPIB card.
1120  ! Modified Variables: @Scope
1130  !
1140    CLEAR SCREEN
1150    Isc=7
1160    Scope=7
```

```
1170    ASSIGN @Scope TO Isc*100+Scope
1180  SUBEND
1190  !
1200  SUB Meas(@S,Isc)
1210  !
1220  ! Description: The scope is setup and waiting to make continuous meas's.
1230  !              1 -- Setup On interupt so Lim. Tst. Comp. gives SRQ.
1240  !              2 -- Setup Limit test.
1250  !              3 -- Set RUN Limit Tests.
1260  !              4 -- Report results.
1270  !
1280  !  Parameters:
1290  !       Passed: (@S) @Scope = specific scope's address,
1300  !               COM /For_cnt/ INTEGER num_acq, N
1310  !
1320  !     Internal: Results(*) = array of values returned from a RESULTS?
1330  !                            Value 4 of each set is the mean.  Therefore,
1340  !                            Results(4), (13), and (22) are the ones of
1350  !                            interest.
1360  !               N = measurement sets requested.
1370  !               Num_acq = the termination variable.
1380  !
1390  ! Modified Variables: Num_acq
1400  !
1410  ! Calls sub programs: Tcount
1420  !
1430    COM /For_cnt/ INTEGER Num_acq,N,S  ! need to pass num_acq on intr.
1440    REAL Results(1:27)                 ! 9 parameters per measurement.
1450    N=10
1460    Num_acq=0
1470    CLEAR SCREEN
1480  Part1:                               ! Setup interupt
1490    ON INTR Isc,9 CALL Tcount
1500  Part2: !
1510    OUTPUT @S;":ites:sour 1;fail nev;mnf pass;run wav,";N
1520    OUTPUT @S;":ites:sour 2;fail nev;mnf pass;run wav,";N
1530    OUTPUT @S;":ites:sour 3;fail nev;mnf pass;run wav,";N
1540    OUTPUT @S;":stop;cdis"
1550    OUTPUT @S;":ites:test on"
1560  Part3: !
1570    ENABLE INTR Isc;2
1580    OUTPUT @S;":run"                   ! scope will wait for triggers.
1590    REPEAT                             ! wait for limit test complet.
1600    UNTIL Num_acq=N
1610  Part4: !
1620    OUTPUT @S;":meas:res?"             ! read summary of measurements.
1630    ENTER @S;Results(*)
1640    CLEAR SCREEN
1650    PRINT "        The results are;"
1660    PRINT "The vpp mean is ";Results(4);","
1670    PRINT "the rise time mean is ";Results(13);", and"
1680    PRINT "the +width mean is ";Results(22);"."
1690    PRINT
1700    PRINT "The results report is:";Results(*);"."
1710  SUBEND
1720  !
1730  SUB Tcount
1740  !
1750  ! Description: Tcount will set Num_acq to the stop value when the Lim.
1760  !              Test Complete interupt occurs.
1770  !  Parameters:
```

```
1780  !      Passed:
1790  !              COM /For_cnt/ INTEGER Num_acq,M
1800  !              Num_acq = the variable used to terminate at proper number.
1810  !              M = the number of acquisitions wanted, termination value.
1820  !    Internal: None
1830  !
1840  ! Modified Variables: num_acq
1850  !
1860  ! Calls sub programs: None
1870  !
1880    COM /For_cnt/ INTEGER Num_acq,M,S
1890    PRINT "hello"
1900    Num_acq=M
1910  SUBEND
```

Automated STM-16 Measurement Example

```
10    !  RE-SAVE "TEST_83480"
20    !  This program prompts the user to set up a source and then
30    !  test for compliance with STM-16, with 15% margin.
40    !
50    !  It assumes an optical module is in Channel 1; otherwise,
60    !  the average power measurement will be invalid.
70    !
80    DIM Query$[15],Command$[15],Q$[9000]
90    COM /Err/ Hpib,Scope
100   COM /S/ @S
110   Hpib=7
120   CLEAR Hpib
130   Scope=7
140   Saddr=Hpib*100+Scope       !Sets the I/O address.
150   Readme_init
160   ASSIGN @S TO Saddr
170   CLEAR Saddr
180   !
190   OUTPUT @S;"*RST" ! Set the scope to a known state.
200   OUTPUT @S;":CHAN1:FILTer ON" ! Turn on built-in filter.
210   OUTPUT @S;":TIMebase:UNITs BITS" ! Select timebase units to be bit period.
220   OUTPUT @S;":TIMebase:BRATe 2.48832E+09" ! Set bit rate (in Hz) for STM-16.
230   OUTPUT @S;":AUToscale" ! Get a good trigger level and vertical scale.
240   OUTPUT @S;":TIMebase:SCALe 2" ! Show 2 bit periods on screen.
250   OUTPUT @S;":MTESt:STANdard STM16" ! Select a standard mask.
260   OUTPUT @S;":MTESt:MMARgin:STATe ON" ! Turn on mask margin.
270   OUTPUT @S;":MTESt:MMARgin:PERCent 15" ! Set amount of mask margin.
280   OUTPUT @S;":MTESt:TEST ON" ! Enable mask testing.
290   Waitfor(@S) ! Wait for scope to finish the operation.
300   WAIT 1 ! Wait a bit more, since scope needs extra time.
310   OUTPUT @S;":MTESt:ALIGn" ! Scale mask to trace.
320   Waitfor(@S) ! Wait for scope to finish the operation.
330   OUTPUT @S;":TIMebase:UNITs TIME" ! Select timebase units to be bit period.
340   Begin$=FNQuery$(@S,":MTESt:SCALe:X1?") ! Set the position to the beginning of the mask.
350   OUTPUT @S;":TIMEBASE:POSITION "&Begin$
360   End$=FNQuery$(@S,":MTESt:SCALe:XDELta?") ! and the scale to the width of the mask.
370   OUTPUT @S;":TIMEBASE:SCALe "&VAL$(VAL(End$)/10)
380   OUTPUT @S;":TIMebase:UNITs BITS" ! Select timebase units to be bit period.
390   OUTPUT @S;":MTESt:RUNode SAMPles,300000" ! Test for 300k samples.
400   OUTPUT @S;":RUN" !
410   Reg=VAL(FNQuery$(@S,":MTER?")) ! Clear the Mask Test Event Register.
420   REPEAT ! Wait until the Mask RUNTIL is achieved.
430     Reg=VAL(FNQuery$(@S,":MTER?"))
440   UNTIL BIT(Reg,0)
450   Fwavs=VAL(FNQuery$(@S,":MTESt:COUNt:FWAVeforms?"))
460   IF Fwavs<2 THEN
470     PRINT "MASK TEST PASSED"
480     Fail=0
490   ELSE
500     PRINT "MASK TEST FAILED"
510     Fail=1
520     PRINT FNQuery$(@S,":MTESt:COUNt:FSAMples?");"samples failed."
530   END IF
540   Readme_pwr
```

```
550   OUTPUT @S;":MTESt:TEST OFF" ! Disable mask testing.
560   OUTPUT @S;":RUN" ! Re-enable data acquisition.
570   Avgpower=VAL(FNQuery$(@S,":MEASure:APOWer? DECibel, CHANnel1")) ! Measure channel 1
580                ! average power in dBm
590   SELECT Avgpower
600   CASE <-2
610     PRINT "AVERAGE POWER TEST FAILED"
620     Fail=1
630     PRINT "Average power is ";Avgpower;"dBm."
640   CASE >9.99E+37
650     PRINT "AVERAGE POWER TEST DATA INVALID"
660   CASE ELSE
670     PRINT "AVERAGE POWER TEST PASSED"
680   END SELECT
690   Readme_cal
700   OUTPUT @S;":DISPlay:CGRade ON" ! Turn on color grade database.
710   !                                              assure meaningful results.
720   OUTPUT @S;":MEASure:CGRade:ERCalibrate" ! Calibrate input offset.
730   Waitfor(@S)
740   Readme_extinct
750   OUTPUT @S;":CDISplay" ! Clear calibration data from database.
760   REPEAT
770     WAIT .1 ! Allow some time for the instrument to acquire some data.
780     OUTPUT @S;":MEASure:CGRade:PEAK?" ! Find out the number of hits in the densest pixel.
790     ENTER @S;Peakvalue
800   UNTIL Peakvalue>50 AND Peakvalue<9.999E+37 ! Loop until the amount returned is valid and large enough to
810   Eratio=VAL(FNQuery$(@S,":MEASure:CGRade:ERATio? DECibel")) ! Measure database
820                ! extinction ratio in dB.
830   SELECT Eratio
840   CASE <5
850     PRINT "EXTINCTION RATIO TEST FAILED"
860     Fail=1
870     PRINT "Extinction ratio is ";Eratio;"dB."
880   CASE >9.99E+37
890     PRINT "EXTINCTION RATIO TEST DATA INVALID"
900   CASE ELSE
910     PRINT "EXTINCTION RATIO TEST PASSED"
920   END SELECT
930   IF Fail THEN
940     PRINT "TEST FAILED"
950   ELSE
960     PRINT "TEST PASSED"
970   END IF
980   LOCAL Saddr ! Return DCA to user's control.
990   Readme_end
1000  END
1010  !
1020  SUB Readme_init
1030    CLEAR SCREEN
1040    PRINT "  This program tests for compliance with STM-16, with 15%"
1050    PRINT "  margin.  You can easily edit it to customize it for your"
1060    PRINT "  needs. "
1070    PRINT
1080    PRINT "  The expected configuration is;"
1090    PRINT "       The scope is at address 7"
1100    PRINT "       The HPIB is at address 7"
1110    PRINT
1120    PRINT "  If the configuration is different, break program and set "
1130    PRINT "  the addresses as required, using the variables Hpib and "
1140    PRINT "  Scope.  Then run again."
1150    PRINT
```

```
1160    PRINT "  Connect your source and set it for STM-16 (2.48832 Gbit/s)."
1170    PRINT
1180    PRINT "  The recommended source is an HP 70841B Pattern Generator."
1190    PRINT
1200    PRINT "  Connect Clock Out to Trigger 2."
1210    PRINT "  Connect the signal to be tested to Input 1."
1220    PRINT
1230    PRINT "  The recommended pattern is 2^31-1."
1240    PRINT
1250    PRINT "  Press Continue when ready to resume operation."
1260    PRINT
1270    PAUSE
1280    CLEAR SCREEN
1290  SUBEND
1300  !
1310  SUB Readme_cal
1320    PRINT
1330    PRINT "   We will now measure extinction ratio.  For best accuracy,"
1340    PRINT "   it is recommended that you allow the instrument to warm up"
1350    PRINT "   for at least 10 minutes."
1360    PRINT "   After the warm-up is complete, turn off the laser to allow "
1370    PRINT "   calibration of the input offset."
1380    PRINT
1390    PRINT "  Press Continue to resume operation when the warmup is complete"
1400    PRINT "  and the laser is turned off."
1410    PRINT
1420    PAUSE
1430    CLEAR SCREEN
1440  SUBEND
1450  !
1460  SUB Readme_end
1470    PRINT
1480    PRINT "   The program has ended.  Thanks for trying our compliance "
1490    PRINT "   testing program."
1500    PRINT
1510    PRINT "   How you can edit the program to customize it for your needs."
1520    PRINT
1530    PRINT "           GOODBYE.     "
1540  SUBEND
1550  !
1560  SUB Readme_extinct
1570    PRINT
1580    PRINT "   The input offset is now calibrated; turn the laser back"
1590    PRINT "   on."
1600    PRINT
1610    PRINT "  Press Continue to resume operation when the laser is back"
1620    PRINT "  on."
1630    PRINT
1640    PAUSE
1650    CLEAR SCREEN
1660  SUBEND
1670  !
1680  SUB Waitfor(@Path)
1690    OUTPUT @Path;"*OPC?"
1700    ENTER @Path;Completion$
1710  SUBEND
1720  !
1730  SUB Readme_pwr
1740    PRINT
1750    PRINT "   We will now measure Average Power."
1760    PRINT
```

```
1770    PRINT "  Press Continue to resume operation."
1780    PRINT
1790    PAUSE
1800    CLEAR SCREEN
1810  SUBEND
1820  !
1830  DEF FNQuery$(@Path,Cmd$)
1840    OUTPUT @Path;Cmd$
1850    ENTER @Path;Result$
1860    RETURN Result$
1870  FNEND
1880  !
```

Eye Diagram Measurement Example

```
10   !  RE-SAVE "EYE_83480"
20   !  This program prompts the user to set up a source and then
30   !  makes all the common eye diagram measurements.
40   DIM Query$[15],Command$[15],Q$[9000]
50   COM /Err/ Hpib,Scope
60   COM /S/ @S
70   Hpib=7
80   CLEAR Hpib
90   Scope=7
100  Saddr=Hpib*100+Scope       !Sets the I/O address.
110  Readme_init
120  ASSIGN @S TO Saddr
130  CLEAR Saddr
140  !
150  OUTPUT @S;"*RST" ! Reset the instrument to a standard state.
160  OUTPUT @S;":TIMebase:UNITs BITS" ! Use bit periods for the horizontal scale.
170  OUTPUT @S;":AUTOscale" ! Get a sensible trigger level and vertical scale.
180  OUTPUT @S;":TIMebase:SCALe 2" ! Measurements require 2 crossings.
190  OUTPUT @S;":DISPlay:CGRade ON" ! Turn on color grade display for eye measurements.
200  OUTPUT @S;"*OPC?" ! Wait for scope to finish before going on.
210  ENTER @S;Done
220  LOCAL Saddr
230  Readme_scope
240  !
250  REPEAT
260    WAIT .2 ! Allow some time for the instrument to acquire some data.
270    OUTPUT @S;":MEASure:CGRade:PEAK?" ! Find out the number of hits in the densest pixel.
280    ENTER @S;Peakvalue
290    PRINT "Waiting until peak color grade hits reaches 50; currently";Peakvalue;"."
300  UNTIL Peakvalue>=50 AND Peakvalue<9.999E+37 ! Loop until the amount returned is valid and large enough to
310  !                                             assure meaningful results.
320  PRINT
330  OUTPUT @S;":MEASure:RISetime? CGRade" ! Measure rise time.
340  ENTER @S;Risetime
350  PRINT "Rise Time= ";FNEng_unit$(Risetime);"seconds."
360  !
370  OUTPUT @S;":MEASure:FALLtime? CGRade" ! Measure fall time.
380  ENTER @S;Falltime
390  PRINT "Fall Time= ";FNEng_unit$(Falltime);"seconds."
400  !
410  OUTPUT @S;":MEASure:OVERshoot? CGRade" ! Measure overshoot.
420  ENTER @S;Overshoot
430  PRINT "Overshoot=";Overshoot;"%."
440  !
450  OUTPUT @S;":MEASure:CGRade:CROSsing?" ! Measure crossing level percentage.
460  ENTER @S;Crossing
470  PRINT "Crossing Level=";Crossing;"%."
480  !
490  OUTPUT @S;":MEASure:CGRade:DCDistortion? PERCent" ! Measure duty cycle distortion in percent.
500  ENTER @S;Dcdistortion
510  PRINT "Duty Cycle Distortion=";Dcdistortion;"%."
520  !
530  OUTPUT @S;":MEASure:CGRade:EHEight?" ! Measure eye height.
540  ENTER @S;Eye_height
```

```
550   OUTPUT @S;":CHANnel1:UNITs?" ! Determine eye height units.
560   ENTER @S;Units$
570   PRINT "Eye Height= ";FNEng_unit$(Eye_height);Units$;"."
580   !
590   OUTPUT @S;":MEASure:CGRade:EWIDth?" ! Measure eye width.
600   ENTER @S;Eye_width
610   PRINT "Eye Width= ";FNEng_unit$(Eye_width);"seconds."
620   !
630   OUTPUT @S;":MEASure:CGRade:JITTer? RMS" ! Measure RMS jitter.
640   ENTER @S;Jitter
650   PRINT "RMS Jitter= ";FNEng_unit$(Jitter);"seconds."
660   !
670   LOCAL Saddr
680   Readme_end
690   END
700   !
710   SUB Readme_init
720     PRINT "  This program performs automated tests on eye diagrams."
730     PRINT "  You can easily edit it to customize it for your needs."
740     PRINT
750     PRINT "  The expected configuration is;"
760     PRINT "       The scope is at address 7"
770     PRINT "       The HPIB is at address 7"
780     PRINT
790     PRINT "  If the configuration is different, break program and set "
800     PRINT "  the addresses as required, using the variables Hpib and "
810     PRINT "  Scope.  Then run again."
820     PRINT
830     PRINT "   Connect your source and set it for your signal requirements."
840     PRINT
850     PRINT "   The recommended source is an HP 70841B Pattern Generator."
860     PRINT
870     PRINT "   Connect Clock Out to Trigger 2."
880     PRINT "   Connect the signal to be tested to input 1."
890     PRINT
900     PRINT "   The recommended pattern is 2^31-1."
910     PRINT
920     PRINT "  Press Continue when ready to resume operation."
930     PRINT
940     PAUSE
950     CLEAR SCREEN
960   SUBEND
970   !
980   SUB Readme_end
990     PRINT
1000    PRINT "   The program has ended.  Thanks for trying our eye diagram "
1010    PRINT "   analysis program."
1020    PRINT
1030    PRINT "   Now you can edit the program to customize it for your needs."
1040    PRINT
1050    PRINT "           GOODBYE.     "
1060  SUBEND
1070  !
1080  SUB Readme_scope
1090    PRINT
1100    PRINT "   Adjust signal, using the controls of the 83480, to show "
1110    PRINT "   one ""Eye"" centered in the display and parts of two"
1120    PRINT "   more ""Eyes"" on the edges of the display."
1130    PRINT
1140    PRINT "  Press Continue when ready to resume operation."
1150    PRINT
```

```
1160    PAUSE
1170    CLEAR SCREEN
1180  SUBEND
1190  !
1200  DEF FNEng_unit$(Number)
1210    SELECT Number/1000
1220    CASE <1.E-18
1230      RETURN VAL$(Number/1.E-18)&" a"
1240    CASE <1.E-15
1250      RETURN VAL$(Number/1.E-15)&" f"
1260    CASE <1.E-12
1270      RETURN VAL$(Number/1.E-12)&" p"
1280    CASE <1.E-9
1290      RETURN VAL$(Number/1.E-9)&" n"
1300    CASE <1.E-6
1310      RETURN VAL$(Number/1.E-6)&" u"
1320    CASE <1.E-3
1330      RETURN VAL$(Number/1.E-3)&" m"
1340    CASE <1
1350      RETURN VAL$(Number)
1360    CASE <1.E+3
1370      RETURN VAL$(Number/1.E+3)&" k"
1380    CASE <1.E+6
1390      RETURN VAL$(Number/1.E+6)&" M"
1400    CASE <1.E+9
1410      RETURN VAL$(Number/1.E+9)&" G"
1420    CASE <1.E+12
1430      RETURN VAL$(Number/1.E+12)&" T"
1440    CASE <1.E+15
1450      RETURN VAL$(Number/1.E+9)&" P"
1460    CASE ELSE
1470      RETURN VAL$(Number/1.E+12)&" E"
1480    END SELECT
1490  FNEND
1500  !
```

Error Messages

This section describes the error messages and how they are generated. The possible causes for the generation of the error messages are also listed in the table at the end of this section.

Error queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. You can read an error from the error queue using the :SYSTEM:ERROR? query.

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.
- the instrument is switched from talk only to addressed mode on the front panel.

Errors −100 to −199

These errors indicate that a 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.

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

Errors –200 to –299

These errors indicate 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.

Errors –300 to –399 or +1 to +32767

These errors indicate 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.

Errors –400 to –499

These errors indicate 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 should cause 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.

Error Messages, Positive Values

Error Number	Error Message	Cause
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.
+1 to +32767		Indicates an instrument specific error has been detected.
2	The % option is not installed	
3	No applications are currently installed	
6	Incompatible signal	
7	Mask test mask align failed	
8	System clock failure occurred: Service is required	
9	A memory error was detected: Service is required	
11	System date and time are incorrect	
12	Low battery voltage detected: Service is required	
13	System firmware error occurred	
14	Fatal system firmware error occurred: Please cycle power	
15	Execution is not possible	
35	No signal was found: Instrument setup is unchanged	
37	System load failed: System firmware is unchanged	
38	Measurement cannot be performed on the selected waveform	
39	Function cannot be performed on the selected waveform	
40	Command execution not possible on the selected waveform	

Error Messages, Positive Values (continued)

Error Number	Error Message	Cause
41	Waveform data is not valid	
48	Print cancelled: Printer is not responding	
50	Forms cannot be printed to disk	
59	Overload condition on %	
61	Memory error occurred in plug-in %: Try reinstalling plug-in	
62	Busy timeout occurred with plug-in %: Try reinstalling plug-in	
63	Communication failure exists at slot %: Service is required	
64	Id error occurred in plug-in %: Service is required	
65	Plug-in % is not supported: System firmware upgrade is needed	
66	Mainframe cal factor memory error: Please perform calibration	
72	Could not store calibration factors: Service is required	
73	Execution not possible: Calibration is required	
74	Execution not possible: Mainframe calibration is required	
79	Probe attenuation (or gain) exceeds calibration limits	
85	Incompatible setup	
88	Testing failed: Please calibrate mainframe and retry self-test	
96	A system firmware file was not found on the disk	

Error Messages, Positive Values (continued)

Error Number	Error Message	Cause
112	Unknown file type	
113	LIF disk format not supported	
114	The disk directory is full	
116	Too many points sent	
120	Execution not possible: Calibration does not match mainframe	
121	Execution not possible: Calibration does not match plug-in	
122	GSP not responding: %	
125	Header information is not valid	
126	Signal is clipped: Instrument setup is unchanged	
127	All labels are in use: To add a new label, delete an old one	
128	Selected polygon is not valid	
129	The selected point caused an intersection and is not valid	
130	Too many vertices	
134	Insufficient points to compute FFT	
136	Can not store ASCII format for records greater than 128k points	
139	Execution not possible: Waveform memory destination is illegal	
140	Exceeded maximum ASCII list length	
141	Measurement cannot be performed: There are no valid sources	
143	Calibration is not possible for plug-in %	
145	Display one channel when making color grade measurements	

Error Messages, Positive Values (continued)

Error Number	Error Message	Cause
146	Unable to measure dark level: Check signal level	
147	Plug-in configuration not allowed	
148	Plug-in initialization will destroy factory calibration factors	
150	Operation permitted for standard masks only	
151	Turn on color grade to measure eye parameters	
154	Control cannot be changed: Timebase view is set to window	
158	Comments not supported for this type of plug-in	
160	Execution not possible: Reference plane calibration is required	
161	Calibration data is not valid	
162	Execution not possible: Select % destination	
163	Execution not possible: No valid % destination available	
164	Mask test mask align failed: Display one channel to align	

Error Messages, Negative Values

Error Number	Error Message	Cause
−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.
−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	Numeric overflow	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.

Error Messages, Negative Values (continued)

Error Number	Error Message	Cause
—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.

Error Messages, Negative Values (continued)

Error Number	Error Message	Cause
–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.
–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.
–241	Hardware missing	
–256	File name not found	
–310	System error	Indicates that a system error occurred.
–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 used if the instrument can not detect more specific errors.
–410	Query INTERRUPTED	
–420	Query UNTERMINATED	
–430	Query DEADLOCKED	
–440	Query UNTERMINATED after indefinite response	

2

Common Commands

Common Commands

Common commands are defined by the IEEE 488.2 standard. They control generic device functions which could be common among many different types of instruments. Common commands can be received and processed by the instrument whether they are sent over the HP-IB as separate program messages or within other program messages.

The following common commands and queries are implemented in this instrument:

- *CLS (Clear Status)
- *ESE (Event Status Enable)
- *ESR? (Event Status Register)
- *IDN? (Identification Number)
- *LRN (Learn)
- *OPC (Operation Complete)
- *OPT? (Option)
- *RCL (Recall)
- *RST (Reset)
- *SAV (Save)
- *SRE (Service Request Enable)
- *STB? (Status Byte)
- *TRG (Trigger)
- *TST? (Test)
- *WAI (Wait-to-Continue)

Receiving Common Commands

Common commands can be received and processed by the instrument whether they are sent over the HP-IB as separate program messages or within other program messages. If a subsystem is currently selected and a common command is received by the instrument, the instrument remains in the selected subsystem. For example, if the program message:

```
"ACQUIRE:TYPE AVERAGE;*CLS;COUNT 1024"
```

is received by the instrument, the instrument sets the acquire type, clears the status information, and then sets the acquire count 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 Registers" in Chapter 1, for a complete discussion of status.

Status Registers

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

Command Descriptions

The following pages describe the Common Commands listed at the beginning of this chapter.

These descriptions include:

- the literal command
- an example of how the command is used within a program
- the query command
- common variables
- where to look for additional information

*CLS (Clear Status)

Command

```
*CLS
```

Clears the status data structures, including the device-defined error queue. It also clears the Request-for-OPC flag.

Example

The following example clears the status data structures of the instrument.

```
10  OUTPUT 707;"*CLS"
20  END
```

See Also

"Monitoring the Instrument " in Chapter 1.

*ESE (Event Status Enable)

Command

```
*ESE <mask>
```

Sets the Standard Event Status Enable Register bits.

<mask>

An integer, 0 to 255, representing a mask value for the bits to be enabled in the Standard Event Status Register as explained in "Monitoring the Instrument " in Chapter 1.

Example

The following 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"
20  END
```

Query

```
*ESE?
```

The query returns the current contents of the Standard Event Status Enable Register.

Returned Format

```
<mask><NL>
```

<mask>

An integer, +0 to +255 (plus sign (+) also returned), representing a mask value for the bits enabled in the Standard Event Status Register as explained in "Monitoring the Instrument " in Chapter 1.

Example

The following example places the current contents of the Standard Event Status Enable Register in the numeric variable, Event. The value of the variable is printed on the controller's screen.

```
10  OUTPUT 707;"*ESE?"
20  ENTER 707;Event
30  PRINT Event
40  END
```

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.

Standard Event Status Enable Register Bits

Bit	Weight	Enables
7	128	PON - Power On
6	64	URQ - User Request
5	32	CME - Command Error
4	16	EXE - Execution Error
3	8	DDE - Device Dependent Error
2	4	QYE - Query Error
1	2	RQC - Request Control
0	1	OPC - Operation Complete

See Also

"Monitoring the Instrument " in Chapter 1.

*ESR? (Event Status Register)

Query

```
*ESR?
```

The query returns the contents of the Standard Event Status Register. Reading this register clears the Standard Event Status Register.

Returned Format

```
<status><NL>
```

<status>

An integer, +0 to +255 (plus sign (+) also returned), representing the total bit weights of all bits that are high at the time you read the register.

Example

The following example places the current contents of the Standard Event Status Register in the numeric variable, Event, then prints the value of the variable to the controller's screen.

```
10  OUTPUT 707;"*ESR?"
20  ENTER 707;Event
30  PRINT Event
40  END
```

See Also

"Monitoring the Instrument " in Chapter 1.

Standard Event Status Register Bits

Bit	Weight	Bit Name	Condition
7	128	PON	1=OFF to ON transition has occurred.
6	64	URQ	0=no front-panel key has been pressed. 1=a front-panel key has been pressed.
5	32	CME	0=no command errors. 1=a command error has been detected.
4	16	EXE	0=no execution errors. 1=an execution error has been detected.
3	8	DDE	0=no device-dependent errors. 1=a device-dependant 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.
0	1	OPC	0=operation is not complete. 1=operation is complete.
0=False=Low 1=True=High			

*IDN? (Identification Number)

Query

```
*IDN?
```

The query returns the instrument model number, serial number, and software version.

Returned Format

```
HEWLETT-PACKARD,54750A,,<XXXXAYYYYY>,<X.XX><NL>
```

<XXXXAYYYYY> This specifies the serial number of the instrument. The first four digits and letter are the serial prefix, which is the same for all identical instruments. The last five digits are the serial suffix, which is assigned sequentially, and is different for each instrument.

<X.XX> This specifies the software version of the instrument and is the revision number.

Example The following example places the instrument's identification number in the string variable, Identify$, then prints the identification number to the controller's screen.

```
10  DIM Identify$[50]       !dimension variable
20  OUTPUT 707;"*IDN?"
30  ENTER 707;Identify$
40  PRINT Identify$
50  END
```

*LRN (Learn)

Command

```
*LRN <setup>
```

<setup>

Sends a previously stored setup to the instrument. This is a definite-length arbitrary block response specifying the current instrument setup. The block size is subject to change with different firmware revisions.

Query

```
*LRN?
```

The query returns a response message (learn string) that contains the instrument's current setup. The instrument's setup can be stored and sent back to the instrument at a later time.

Returned Format

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

Example

The following example places the instrument's current setup in the string variable, Current$.

```
10  DIM Current$[10000]          !Dimension variable
20  OUTPUT 707;"*LRN?"
30  ENTER 707 USING "-K";Current$
40  OUTPUT 707; Current$          !Sends entire string including
50                                !command header
60  END
```

NOTE

The *LRN query always returns :SYSTEM:SETUP as a prefix to the setup block. The :SYSTEM:HEADER command has no effect on this response.

When HEADers and LONGform are ON, the SYSTEM:SETUP? query performs the same function as the *LRN query. Otherwise, *LRN and SETup are not interchangeable.

*OPC (Operation Complete)

Command

```
*OPC
```

Sets the operation complete bit in the Standard Event Status Register when all pending device operations have finished.

Example

The following example sets the operation complete bit in the Standard Event Status Register when the DIGITIZE operation is complete.

```
10  OUTPUT 707;":DIGITIZE:CHANNEL1;*OPC"
20  END
```

Query

```
*OPC?
```

The query places an ASCII character "1" in the instrument's output queue when all pending selected device operations have finished.

Returned Format

```
1<NL>
```

Example

The following example places an ASCII character "1" in the instrument's output queue when the AUTOSCALE operation is complete.

Then the value in the output queue is placed in the numeric variable, Complete.

```
10  OUTPUT 707;":AUTOSCALE;*OPC?"
20  ENTER 707;Complete
30  END
```

The *OPC query allows synchronization between the controller 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.

*OPT? (Option)

Query

```
*OPT?
```

The query returns the model number and serial number of the mainframe and all installed plug-in options.

Plug-in information is returned for each plug-in option installed in the mainframe. If plug-in options are not installed, a zero is returned.

Note that the length of the returned option string may increase in future releases as more options become available. Once implemented, the option name (delimited by a comma) will be appended to the end of the returned string.

Returned Format

```
<MF_MOD_NO>
<XXXXAYYYYY>,<PI_MOD_NO>
<XXXXAYYYYY>,...
```

<MF_MOD_NO> This is the HP model number of the instrument mainframe.

<PI_MOD_NO> This is the HP model number of the plug-in.

<XXXXAYYYYY> This is the serial number of the mainframe or plug-in, respectively. The first four digits and letter are the serial prefix, which is the same for all identical units. The last five digits are the serial suffix which is assigned sequentially and is different for each unit.

Example

This example places all options into the string variable, Options$, then prints the option model and serial numbers to the controller's screen.

```
10  DIM Options$[100]
20  OUTPUT 707;"*OPT?"
30  ENTER 707;Options$
40  PRINT Options$
50  END
```

*RCL (Recall)

Command

```
*RCL <register>
```

Restores the state of the instrument to the setup previously stored in the specified save/recall register. An instrument 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>

An integer, 0 through 9, specifying the save/recall register that contains the instrument setup you want to recall.

Example

The following example restores the instrument to the instrument setup stored in register 3.

```
10  OUTPUT 707;"*RCL 3"
20  END
```

An error message appears on screen if nothing has been previously saved in the specified register.

*RST (Reset)

Command

```
*RST
```

Places the instrument in a known state. The following table lists the reset conditions as they relate to the instrument commands.

Example

The following example resets the instrument to a known state.

```
10  OUTPUT 707;"*RST"
20  END
```

***RST (Reset) Conditions**

Item	Setting
Global	
Run/Stop	Run
Menu	Unchanged
Headers	Off
Longform	Off
Time Base	
Scale	1 ns/div
Position	22 ns
Reference	Left
Windowing	Disabled
Window scale	500 ps/div
Window position	22 ns
Trigger	
Level	0 V
Sweep	Triggered
Hysteresis	Normal
Edge source	First trigger source
Slope	Positive
Bandwidth limit	Off
Acquisition	
Record length	Automatic
Averaging	Off
Number of averages	16
Display	
Persistence	Variable
Persistence time	Minimum
Draw waveforms	Fast
Graticule	Grid
Intensity	20
Graphs	1
Channel position	Channel 1 upper Channel 2 lower Channel 3 upper Channel 4 lower
Colors	Default
Color grade	Off
Labels	Off

***RST (Reset) Conditions (continued)**

Item	Setting
Marker	
Mode	Measurement
Readout	Off
X1, Y1 source	First available channel
X1 position	22 ns
Y1 position	0 V
X2, Y2 source	First available channel
X2 position	22 ns
Y2 position	0 V
Define measure	
Thresholds-percent	10%, 50%, 90%
Top-base	Calculated
Statistics	Off
Delta time	
Start edge	Rising
Start edge number	1
Start edge threshold	Middle
Stop edge	Falling
Stop edge number	1
Stop edge threshold	Middle
Waveform	
Pixel memory state	Off
Waveform save format	text
Byte order	MSB first
Waveform source	Memory 1
Memory type	Waveform
Memory Display	Off
Y Scale	Unchanged
Y Offset	Unchanged
X Scale	Unchanged
X Position	Unchanged

***RST (Reset) Conditions (continued)**

Item	Setting
Math	
Function	f1
Function state	Off
Operator	Magnify
Operand 1	First available channel or memory 1
Operand 2	First available channel or memory 1
Vertical scaling	Track Source
Horizontal Scaling	Track Source
Channel (electrical channels)	
Display	On
Scale	10 mV/div or maximum
Offset	0 V
Probe atten units	Ratio
Probe attenuation	Unchanged
Units	Volts
External offset	0.0
External gain	1.0
Utility	
CRT pattern	Off
Light output	Off
Color purity	Off
Color ID	Current color ID (0)
Disk	
operation	Unchanged

*SAV (Save)

Command

```
*SAV <register>
```

Stores the current state of the instrument in a save register.

<register>

An integer, 0 through 9, specifying the save/recall register in which to save the current instrument setup.

Example

The following example stores the current instrument setup to register 3.

```
10  OUTPUT 707;"*SAV 3"<R>
20  END
```

*SRE (Service Request Enable)

Command

```
*SRE <mask>
```

Sets the Service Request Enable Register bits.

<mask>

An integer, 0 to 255, representing a mask value for the bits to be enabled in the Service Request Enable Register as explained in "Monitoring the Instrument " in Chapter 1.

Example

The following 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"
20  END
```

Query

```
*SRE?
```

The query returns the current contents of the Service Request Enable Register.

Returned Format

```
<mask><NL>
```

<mask>

An integer, +0 to +255 (plus sign (+) also returned), representing a mask value for the bits enabled in the Service Request Enable Register.

Example

The following example places the current contents of the Service Request Enable Register in the numeric variable, Value, then prints the value of the variable to the controller's screen.

```
10  OUTPUT 707;"*SRE?"
20  ENTER 707;Value
30  PRINT Value
40  END
```

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.

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?
```

The query returns the current contents of the Status Byte including the Master Summary Status (MSS) bit.

Returned Format

```
<value><NL>
```

<value>

An integer, +0 to +255 (plus sign (+) also returned), representing a mask value for the bits enabled in the Status Byte.

Example

The following example reads the contents of the Status Byte into the numeric variable, Value, then prints the value of the variable to the controller's screen.

```
10  OUTPUT 707;"*STB?"
20  ENTER 707;Value
30  PRINT Value
40  END
```

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 instrument has at least one reason for requesting service.

Status Byte Register Bits

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 = instrument has no reason for service. 1 = instrument 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
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
```

Identical to the Group Execute Trigger message (GET) or RUN command. It acquires data for the active waveform display according to the current settings.

Example

The following example starts the data acquisition for the active waveform display according to the current settings.

```
10  OUTPUT 707;"*TRG"
20  END
```

*TST? (Test)

Query

```
*TST?
```

The query causes the instrument 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. A zero indicates that the test passed and a non-zero indicates the test failed.

NOTE

All front-panel inputs must be disconnected before sending this command.

Returned Format

```
<result><NL>
```

<result>

0 for pass; non-zero for fail.

Example

The following example performs a self-test on the instrument and places the results in the numeric variable, Results, then prints the results to the controller's screen.

```
10  OUTPUT 707;"*TST?"
20  ENTER 707;Results
30  PRINT Results
40  END
```

If a test fails, refer to the troubleshooting section of the service guide.

*WAI (Wait-to-Continue)

Command

```
*WAI
```

Prevents the instrument from executing any further commands or queries until all currently executing commands are completed.

NOTE

This command stops the parsing of commands, *not* the entry of bus data into an input buffer. To stop the execution of a program at a specific point, use the *OPC command/query.

The *WAI command does not set the operation complete bit in the Standard Event Status register.

3

Root Level Commands

Root Level Commands

Root level commands control many of the basic operations of the instrument 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.

The following root level commands and queries are implemented in this instrument:

- AER? (Arm Event Register)
- AUToscale
- BLANk
- CDISplay
- DIGitize
- ERASe
- HEEN (Histogram Event Enable)
- HER? (Histogram Event Register)
- LER? (Local Event Register)
- LTEE (Limit Test Event Enable)
- LTER? (Limit Test Event Register)
- MENU
- MERGe
- MODel?
- MTEE (Mask Test Event Enable),
- MTER? (Mask Test Event Register)
- OPEE (Operation Status Enable)
- OPER? (Operation Status Register)
- PRINt
- RECall:SETup
- RUN
- SERial (Serial Number)
- SINGle
- STOP
- STORe
  - PMEM1
  - SETup
  - WAVeform

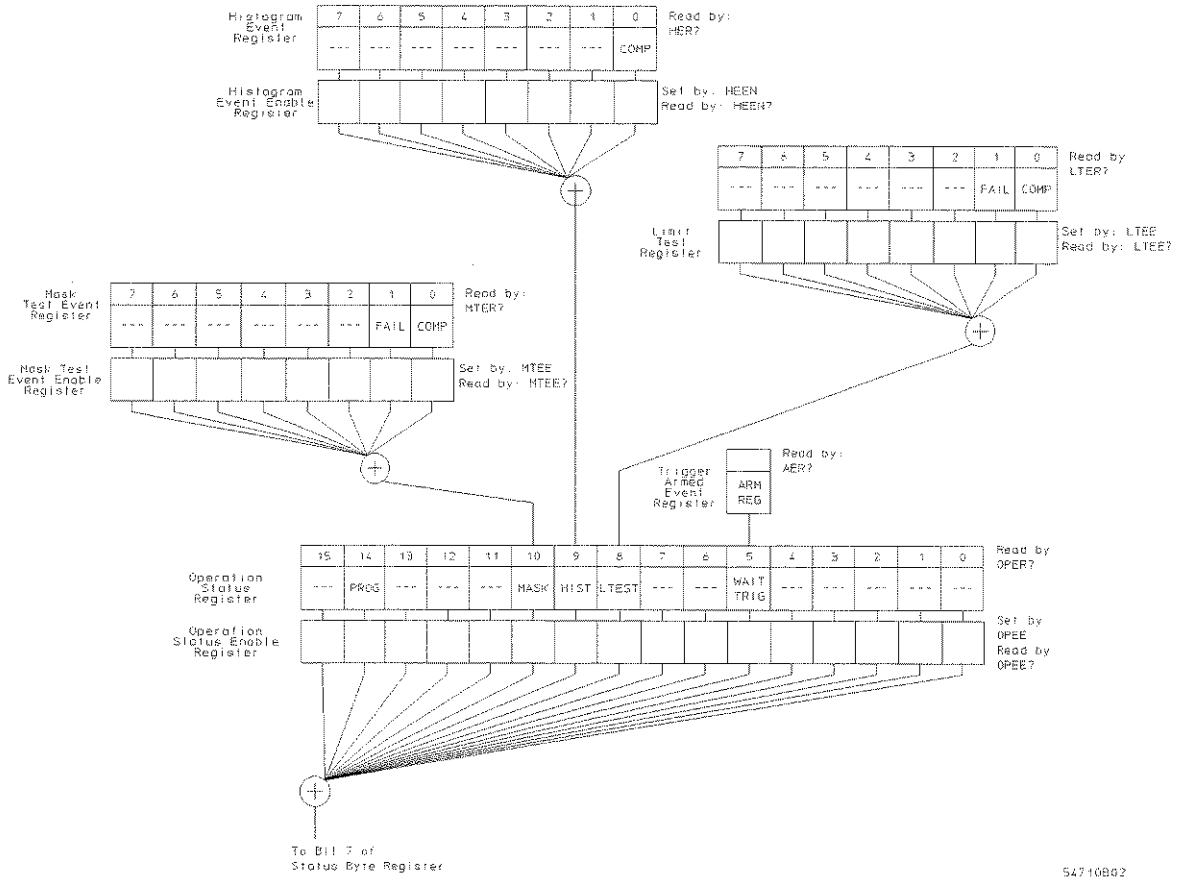
- TEER (Trigger Event Enable Register)
- TER? (Trigger Event Register)
- UEE (User Event Enable)
- UER? (User Event Register)
- VIEW

Status Reporting Data Structures

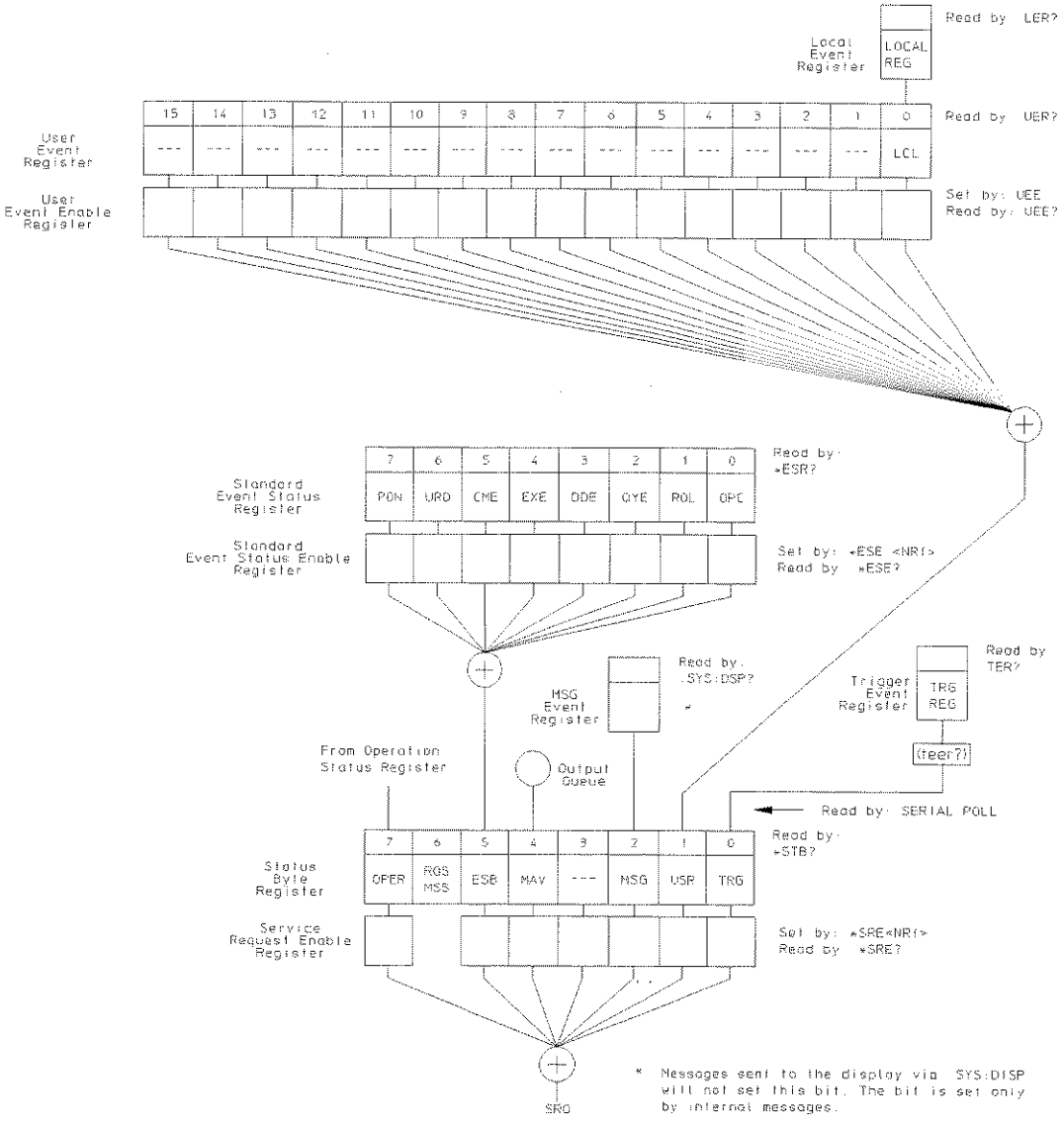
The following figure shows the different status reporting data structures and how they work together. To make it possible for any of the Standard Event Status Register bits to generate a summary bit, the bits must be enabled. These bits are enabled by using the *ESE common command to set the corresponding bit in the Standard Event Status Enable Register.

To generate a service request (SRQ) interrupt to an external controller, 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.

Various root level commands, documented in this chapter, are used to query and set various registers within the register set.



Status Reporting Data Structure



Status Reporting Data Structure (continued)

AER? (Arm Event Register)

Query

```
:AER?
```

Reads the Arm Event Register and returns "1" or "0". After the Arm Event Register is read, the register is cleared. The returned value "1" indicates a trigger has occurred and "0" indicates a trigger has not occurred.

NOTE

The bit will only be set on the first trigger armed after the *CLS command is sent to the instrument. Once this bit is set, it is cleared only by reading the Status Byte or by sending a *CLS command.

Returned Format

```
[:AER] {1 | 0}
```

AUToscale

Command

```
:AUToscale
```

Causes the instrument to evaluate all input signals and to find the optimum conditions for displaying the signal. It searches each of the installed channels for input signals and shuts off channels on which no signal is found. It adjusts the vertical gain and offset for each channel that has a signal, and sets the time base on the lowest numbered input channel with a signal. The trigger is found by searching each module from, left-to-right, until a trigger signal is detected. If signals cannot be found on any vertical input, the instrument is returned to its former state.

Autoscale sets the following conditions:

- Channel Display, Scale, and Offset
- Trigger Sweep, Source, Level, Slope, and Hysteresis
- Acquisition Record length
- Time base Scale and Position
- Marker mode set to measurement

Autoscale turns off the following items:

- Measurements on sources that are turned off
- Functions
- Windows
- Memories
- Measurement limit test
- FFT display
- Mask Test
- Histograms

No other controls are affected by Autoscale.

Example

```
10  OUTPUT 707;":AUTOSCALE"
20  END
```

BLANk

Command

```
:BLANK {CHANnel<number> | FUNCtion<number> |
 WMEMory<number> | PMEMory<number> | FFT |HISTogram}
```

Turns off an active channel, function, waveform memory, pixel memory, FFT, or histogram. The VIEW command turns them on.

<number>

For channels: 1, 2, 3, or 4.

For functions: 1 or 2.

For waveform memories (WMEMory): 1, 2, 3, or 4.

For pixel memory (PMEMory): 1.

Example

```
10  OUTPUT 707;":BLANK CHANNEL1"
20  END
```

CDISplay

Command

```
:CDISplay
```

Clears the display and resets all associated measurements. If the instrument is in the stopped mode, all data that are currently displayed are erased. If the instrument is running, all data in active channels and functions are erased; however, new data are displayed on the next acquisition. Waveform and pixel memories are not erased. Sending this command is the same as pressing the front-panel Clear Display key, or sending :ERASE PMEMORY0.

Example

```
10  OUTPUT 707;":CDISPLAY"
20  END
```

DIGitize

Command

```
:DIGitize [<waveform_name>][,<waveform_name>]
```

Invokes a special mode of data acquisition that is more efficient than using the RUN command. This command initializes the selected channels, functions, or FFT to "unacquired," and then acquires them according to the current instrument settings. When all signals are completely acquired, the instrument is placed in the stopped state.

If channel, function, or FFT parameters are specified, then these are the only waveforms acquired. To speed acquisition, these 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 are invalidated.

NOTE

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

If the DIGitize command is issued without parameters, the digitize operation is performed on the channels, functions, and FFT 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.

Also, this mode can be used if it is desirable to view the digitize results because the display state of the digitized waveforms is not affected.

The data acquired with the DIGitize command are placed in the normal channel, function, or FFT memories, just as when acquired via the RUN command.

<waveform_name> CHANnel<number>, FUNCtion<number>, or FFT

<number> For channels: 1, 2, 3, or 4.

For functions: 1 or 2.

Example The following example uses the Digitize command to acquire data on channel 1 and function 2.

```
10  OUTPUT 707;":DIGITIZE CHANNEL1,FUNCTION2"
20  END
```

The Acquire subsystem commands are used to set up conditions, such as Type and Count for the next Digitize command. The Waveform subsystem commands are used to determine how the data is transferred out of the instrument and how to interpret the data.

ERASe

Command

```
:ERASe [PMEMory<number>]
```

Erases the specified pixel memory.

<number>

0 or 1.

Example

The following example clears the specified pixel memory and anything on the display from that pixel memory.

```
10  OUTPUT 707;":ERASE PMEMORY1"
20  END
```

Erasing pixel memory 0 is a special case, depending on whether the instrument is in the stopped mode or is running. If the instrument is in the stopped mode, all data that are currently displayed are erased. If the instrument is running, all data in active channels and functions are erased; however, new data are displayed on the next acquisition. Waveform and pixel memories are not erased. In addition, erasing pixel memory 0 is the same as pressing the Clear Display front-panel key. It clears the display and resets all associated measurements.

Erasing Pixel Memory 1 clears the static pixel memory and anything on the display from that pixel memory. Pixel Memory 1 is the pixel memory accessible from the front-panel Waveform menu.

HEEN (Histogram Event Enable register)

Command

```
:HEEN <mask>
```

Sets a mask into the Histogram Event Enable register. A "1" in a bit position enables the corresponding bit in the Histogram Event register to set bit 9 in the Operation Status Register.

<mask>

The decimal weight of the enabled bits. Only bit 0 is used at this time; so to enable the COMP bit, set bit 0 to "1" with the HEEN command. Otherwise, set bit 0 to "0".

Query

```
:HEEN?
```

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

Returned Format

```
[HEEN?] <mask><NL>
```

HER? (Histogram Event Register)

Query

```
:HER?
```

The query returns the current value of the Histogram Event Register as a decimal number.

Bit 0 (COMP) of the Histogram Event Register is set when the histogram completes. The histogram completion criteria are set by the HISTogram:RUNTil command.

Returned Format

```
[HER?] <value><NL>
```

LER? (Local Event Register)

Query

```
:LER?
```

The query 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.

Returned Format

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

Example

The following example checks to see if a remote-to-local transition has taken place and places the result in the string variable, Answer$, and then it prints the result to the controller's screen.

```
10  DIM Answer$[50]      !Dimension variable
20  OUTPUT 707;":LER?"
30  ENTER 707;Answer$
40  PRINT Answer$
50  END
```

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.

LTEE (Limit Test Event Enable register)

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>

The decimal weight of the enabled bits. Only bits 0 and 1, of the Limit Test Event Register, are used at this time. The useful mask values are shown in the following table:

Enable	Mask Value
Block COMP and FAIL	0
Enable COMP, block FAIL	1
Enable FAIL, block COMP	2
Enable COMP and FAIL	3

Query

```
:LTEE?
```

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

Returned Format

```
[LTEE] <mask><NL>
```

LTER? (Limit Test Event Register

Query

```
:LTER?
```

Returns the current value of the Limit Test Event Register as a decimal number.

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>
```

MENU

Command

```
:MENU {APPLication | CHANnel<number> | ACQuire
| TIMebase | TRIGger | DISK | DISPlay | MARKer | MEASure | MATH
| WAVeform | SETup | PRINt | HELP | UTILity | FFT | LTESt
| HISTogram | MTESt | MEYE}
```

Selects one of the menus on the front panel.

Example

The following example displays the Timebase menu on the instrument's screen.

```
10  OUTPUT 707;":MENU TIMEBASE"
20  END
```

Query

```
:MENU?
```

The query returns the name of the current menu into a string.

Returned Format

```
[:MENU] {APPLication | CHANnel<number> | ACQuire
| TIMebase | TRIGger | DISK | DISPlay | MARKer | MEASure | MATH
| WAVeform | SETup | PRINt | HELP | UTILity | FFT | LTESt
| HISTogram | MTESt | MEYE}
```

Example

The following example places the name of the currently displayed menu in the string variable, Answer$, and then prints the contents of the variable on the controller's screen.

```
10  DIM Answer$[50]          !Dimension variable
20  OUTPUT 707;":MENU?"
30  ENTER 707;Answer$
40  PRINT ANSWER$
50  END
```

MERGe

Command

```
:MERGe [PMEMory<number>]
```

Stores the contents of the active display into the pixel memory. The stored data are essentially a copy of the waveform data that are presently displayed on screen. Display labels are also copied to pixel memory.

Since there is only one pixel memory in the instrument, the PMEMory<number> parameter is optional and PMEMory1 is the only choice if a parameter is sent.

Example

The following example stores the contents of the active display into Pixel Memory 1.

```
10  OUTPUT 707;":MERGE PMEMORY1"
20  END
```

MODel?

Query

```
:MODel? {FRAMe | PLUGin<number>}
```

The query returns the HP model number for the frame or plug-in.

<number> The residing slot number of the plug-in.

Returned Format A six-character alphanumeric model number within quotes. Output is determined by header and longform status as in the following table. A response of KNOWN indicates that the identifier is corrupt.

Model? Returned Format

Header		Longform		Response
ON	OFF	ON	OFF	
	•		•	83480A *or* 54750A
	•	•		83480A *or* 54750A
•			•	:MOD 83480A *or* :MOD 54750A
•		•		:MODEL 83480A *or* :MODEL 54750A

Example The following example places the model number of the frame in a string variable, Model$, then prints the contents of the variable on the controller's screen.

```
10 Dim Model$[13] !Dimension variable
20 OUTPUT 707;":Model? FRAME"
30 ENTER 707; Model$
40 PRINT MODEL$
50 END
```

MTEE (Mask Event Enable register)

Command

```
:MTEE <mask>
```

Sets a mask into the Mask Event Enable register. A "1" in a bit position enables the corresponding bit in the Limit Event Register to set bit 10 in the Operation Status Register.

<mask>

The decimal weight of the enabled bits. Only bits 0 and 1 of the Mask Test Event Register are used at this time, so the useful mask values are defined by the following table:

Enable	Mask Value
Block COMP and FAIL	0
Enable COMP, block FAIL	1
Enable FAIL, block COMP	2
Enable COMP and FAIL	3

Query

```
:MTEE?
```

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

Returned Format

```
[MTEE] <mask><NL>
```

MTER? (Mask Test Event Register)

Query

```
:MTER?
```

Returns the current value of the Mask Event Register as a decimal number.

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:RUMode command.

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 polygon defined in the mask.

Returned Format

```
[MTER?] <value><NL>
```

OPEE (Operation Status Enable register)

Command

```
:OPEE <mask>
```

Sets a mask in the Operation Status Enable Register. Each bit set to a "1" enables that bit to set bit 7 in the Status Byte Register and potentially cause an SRQ to be generated. Bit 5, Wait for Trig (Trigger Armed), bit 8 Ltest (Limit Test), bit 9 Hist (Histogram), and bit 10 Mask (Mask Test) bits are 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? (Operation Status Register)

Query

```
:OPER?
```

The query returns the value contained in the Operation Status Register as a decimal number. This register hosts the WAIT TRIG bit (bit 5), the LTEST bit (bit 8), the HIST bit (bit 9), the MASK bit (bit 10), and the PROG bit (bit 14).

The WAIT TRIG bit is set by the Trigger Armed Event Register and indicates that the trigger is armed.

The LTEST bit is set when a limit test fails or is completed and sets the corresponding FAIL or COMP bits in the Limit Test Event Register.

The HIST bit is set when the COMP bit is set in the Histogram Event Register, indicating that the histogram measurement has satisfied the specified completion criteria.

The MASK 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 Event Register.

The PROG bit is reserved for future use.

Returned Format

```
[OPER?] <value><NL>
```

PRINt

Command

```
:PRINt
```

Outputs a copy of the screen to a printer or other device destination specified in the Hardcopy subsystem. The selection of the output and the printer can be specified using the Hardcopy subsystem commands.

Example

The following example outputs a copy of the screen to a printer, on the Centronics port, or to a disk file.

```
10  OUTPUT 707;":PRINT"
20  END
```

Example

If the destination is HP-IB, use the following commands after the PRINT command. This example assumes the printer HP-IB address is 1.

```
10 OUTPUT 707;":PRINT"
20 SEND 7;UNT UNL
30 SEND 7; LISTEN 1
40 SEND 7; TALK 7
50 SEND 7; DATA
60 END
```

NOTE

After the print has completed, a "LOCAL 707" must be sent to release Local Lockout when using the example program.

RECall:SETup

Command

```
:RECall:SETup <setup_num>
```

Recalls a setup. The SETup command, sent with the RECall command, recalls a setup from the internal memory that has been previously saved using the STORe:SETup command or saved from the front panel.

<setup_num>

Setup number 0 through 9.

Examples

The following command recalls a setup from a setup memory that has been previously saved using the STORe:SETup command or saved from the front panel.

```
10 OUTPUT 707;":RECall:SETup 2"
20 END
```

RUN

Command

```
:RUN
```

Places the instrument in the running state, in which waveforms are acquired according to the current settings. It enables acquisition cycles to run repetitively until the STOP command is received.

Example

The following example acquires data for the instrument in a repetitive manner.

```
10  OUTPUT 707;":RUN"
20  END
```

SERial (Serial Number)

Command

```
:SERial {FRAMe | PLUGin<number>},<string>
```

Sets the serial number for the instrument frame or plug-in. The instrument's serial number is entered at HP; therefore, setting the serial number is not normally required unless the instrument is serialized for a different application. The mainframe serial number is protected by the "Frame cal protected" switch on the rear panel. The plug-in serial numbers are protected by calibration protect switches inside the plug-in. The switches need to be in the unprotected position in order to program the serial numbers. The switch position is displayed in the Utility/System Config front-panel menu.

This serial number is part of the string returned for the *IDN? query in the common commands.

<string>

A ten-character alphanumeric serial number within quotes.

Example

The following example sets the serial number for the instrument's frame to "1234A56789."

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

Query

```
:SERial? {FRAMe | PLUGin<number>}
```

The query returns the current serial number string for the specified frame or plug-in.

Returned Format

```
[:SERial {FRAMe | PLUGin<number>}]
```

Example

```
10 Dim Serial$[50]       !Dimension variable
20 OUTPUT 707;":SERIAL? FRAME"
30 ENTER 707; Serial$
40 PRINT SERIAL$
50 END
```

SINGle

Command

```
:SINGle
```

Causes the instrument to make a single acquisition when the next trigger event occurs.

The :STOP command should be sent before this command to ensure the instrument is not running. This will ensure that only a single trigger is accepted and one acquisition is acquired.

Example

The following example sets up the instrument to make a single acquisition when the next trigger event occurs.

```
10  OUTPUT 707;":SINGLE"
20  END
```

STOP

Command

```
:STOP
```

Causes the instrument to stop acquiring data for the active display. The RUN or SINGle command must be used to restart the acquisition.

Example

The following example stops the current data acquisition.

```
10  OUTPUT 707;":STOP"
20  END
```

STORe:PMEMory1

Command

```
:STORe:PMEMory1 PMEMory1
```

Stores the current display in pixel memory 1.

STORe:SETup

Command

```
:STORe:SETup <setup_num>
```

Saves the current instrument setup in setup memories 0 through 9.

<setup_num> For setup memories 0 through 9.

STORe:WAVeform

Command

```
:STORe:WAVeform {CHANnel<number> | FUNCtion<number> |
 WMEMory<number> | HISTogram | FFT},{WMEMory<number>}
```

Copies a channel, function, stored waveform, histogram, or FFT to a waveform memory. The first parameter specifies the source of the copy and can be any channel, function, waveform memory, histogram, or the FFT. The second parameter is the destination of the copy, and can be any waveform memory.

<number> For channels: 1, 2, 3, or 4.

For functions: 1 or 2.

For waveform memories (WMEMory): 1, 2, 3, or 4.

Example The following example copies channel 1 to waveform memory 3.

```
10  OUTPUT 707;":STORE:WAVEFORM CHANNEL1,WMEMORY3"
20  END
```

TEER (Trigger Event Enable Register)

Command

```
:TEER <mask>
```

Sets a mask into the Trigger Event Enable Register. A "1" in a bit position enables the corresponding bit in the Trigger Event Register to set bit 0 in the Operation Status Register.

<mask> The decimal weight of the enabled bits. Only bit 0 is used at this time; so to enable the COMP bit, set bit 0 to 1 with the HEEN command; otherwise, set bit 0 to 0.

Query

```
:TEER?
```

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

Returned Format

```
:[TEER] <mask><NL>
```

TER? (Trigger Event Register)

Query

```
:TER?
```

The query reads the Trigger Event Register. A "1" is returned if a trigger has occurred. A "0" is returned if a trigger has not occurred.

Returned Format

```
{1 | 0}<NL>
```

Example

The following example checks the current status of the Trigger Event Register and places that status in the string variable, Current$. Then it prints the contents of the variable to the controller's screen.

```
10  DIM Current$[50]          !Dimension variable
20  OUTPUT 707;":TER?"
30  ENTER 707;Current$
40  PRINT Current$
50  END
```

Once this bit is set, it can only be cleared by reading the Status Byte, reading the register with the :TER? query, or by sending a *CLS common command.

After the Trigger Event Register is read, it is cleared.

UEE (User Event Enable register)

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>

The decimal weight of the enabled bits.

Query

```
:UEE?
```

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

Returned Format

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

UER? (User Event Register)

Query

```
:UER?
```

The query returns the current value of the User Event Register as a decimal number. Bit 0 (LCL - Remote/Local change) is used. All other bits are reserved.

Returned Format

```
[:UER?] <value><NL>
```

VIEW

Command

```
:VIEW {CHANnel<number> | FUNCtion<number> | WMEMory<number> |
PMEMory<number> | HISTogram | FFT}
```

Turns on a channel, function, pixel memory, waveform memory, histogram, or FFT.

<number>

For channels: 1, 2, 3, or 4.

For functions: 1 or 2.

For waveform memories (WMEMory): 1, 2, 3, or 4.

For pixel memories (PMEMory): 1.

Example

The following example turns on channel 1.

```
10  OUTPUT 707;":VIEW CHANNEL1"
20  END
```

See Also

The BLANk command turns off a channel, function, pixel memory, waveform memory, or FFT.

4

System Commands

System Commands

System subsystem commands control the way in which query responses are formatted, simulate front-panel key presses, send and receive setup strings, and enable reading and writing to the advisory line of the instrument. The date and time in the instrument can also be set and read using the System subsystem commands.

The System subsystem contains the following commands and queries:

- DATE
- DSP
- ERRor?
- HEADer
- KEY
- LONGform
- SETup
- TIME

DATE

Command

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

Sets the date in the instrument and is not affected by the *RST common command.

<year>

```
<yyyy> | <yy>
```

<month>

```
<1, 2,  . . . 12> | <JAN, FEB, MAR . . .>
```

<day>

```
<1 . . . 31>
```

Example

The following example sets the date to February 1, 1993.

```
10  OUTPUT 707;":SYSTEM:DATE 1,2,93"
20  END
```

Query

```
:SYSTem:DATE?
```

The query returns the current date in the instrument.

Returned Format

```
[:SYSTem:DATE] <day> <month> <year>
```

Example

The following example queries the date.

```
10 DIM DATE$ [50]
20 OUTPUT 707;":SYSTEM:DATE?"
30 ENTER 707; DATE$
40 PRINT DATE$
50 END
```

DSP

Command

```
:SYSTem:DSP <string>
```

Writes a quoted string, excluding quotes, to the advisory line of the instrument display. If you want to clear a message on the advisory line, send a null (empty) string.

<string>

An alphanumeric character array up to 64 bytes long.

Example

The following example writes the message, "Test 1," to the advisory line of the instrument.

```
10  OUTPUT 707;":SYSTEM:DSP ""Test 1"""
20  END
```

Query

```
:SYSTem:DSP?
```

The query 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 instrument in the string variable, Advisory$. Then, it prints the contents of the variable to the controller's screen.

```
10  DIM Advisory$[64]          !Dimension variable
20  OUTPUT 707;":SYSTEM:DSP?"
30  ENTER 707;Advisory$
40  PRINT Advisory$
50  END
```

ERRor?

Query

```
:SYSTem:ERRor? [{NUMBer | STRing}]
```

The query outputs the next error number in the error queue over the HP-IB. 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 quoted string describing the error. The following table lists the error numbers and their corresponding error messages. The error messages are also listed in "Error Messages" in Chapter 1, where possible causes are given for each message.

Returned Format

```
[:SYSTem:ERRor]<error_number>[,<quoted_string>]<NL>
```

<error_number> A numeric error code.

<quoted_string> A quoted string describing the error.

Example The following example reads the oldest error number and message in the error queue into the string variable, Condition$, then prints the contents of the variable to the controller's screen.

```
10  DIM Condition$[64]          !Dimension variable
20  OUTPUT 707;":SYSTEM:ERROR? STRING"
30  ENTER 707;Condition$
40  PRINT Condition$
50  END
```

This instrument has an error queue that 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.

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

See Also

Refer to "Error Messages" in Chapter 1 for more information on error messages and their possible causes.

Error Messages

Error Number	Description	Error Number	Description
0	No error	−158	String data not allowed
−100	Command error	−160	Block data error
−101	Invalid character	−161	Invalid block data
−102	Syntax error	−168	Block data not allowed
−103	Invalid separator	−170	Expression error
−104	Data type error	−171	Invalid expression
−105	GET not allowed	−178	Expression data not allowed
−108	Parameter not allowed	−200	Execution error
−109	Missing parameter	−222	Data out of range
−112	Program mnemonic too long	−223	Too much data
−113	Undefined header	−310	System error
−121	Invalid character in number	−350	Too many errors
−123	Numeric overflow	−400	Query error
−124	Too many digits	−410	Query INTERRUPTED
−128	Numeric data not allowed	−420	Query UNTERMINATED
−131	Invalid suffix	−430	Query DEADLOCKED
−138	Suffix not allowed	−440	Query UNTERMINATED after indefinite response
−141	Invalid character data		
−144	Character data too long		

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.

Example

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

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

Query

```
:SYSTem:HEADer?
```

The query returns the state of the SYSTem:HEADer command.

Returned Format

```
[:SYSTem:HEADer] {1 | 0}<NL>
```

Example

The following example checks the current status of headers and places the result in the string variable, Result$, then it prints the contents of the variable to the controller's screen.

```
10  DIM Result$[50]      !Dimension variable
20  OUTPUT 707;":SYSTEM:HEADER?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

NOTE

Turn headers off when returning values to numeric variables. Headers are always off on all COMMON command queries because headers are not defined in the IEEE 488.2 standard.

KEY

Command

```
:SYSTem:KEY <key_code>
```

Simulates the pressing of a specified front-panel key. SYSTem:KEY commands may be sent over HP-IB in any order that are legal key presses from the front panel. Make sure the instrument is in the desired state before executing the SYSTem:KEY command.

<key_code> An integer, as described in the following table.

NOTE

The positions of softkeys will change in future firmware releases. Programs that depend on the order of softkey presses may not be compatible with future firmware releases.

CAUTION Use of the :SYSTem:KEY commands can place the instrument into undefined states. We recommend using the specific command for the action desired. The definition of keys varies with each menu and could change with a new firmware revision. Programs using these commands might not be transportable.

Example

The following example performs an AUTOSCALE operation by simulating the front-panel AUTOSCALE key.

```
10  OUTPUT 707;":SYSTEM:KEY 2"
20  END
```

Query

```
:SYSTem:KEY?
```

The query returns the key code for the last key pressed on the front panel.

Returned Format

```
[:SYSTem:KEY] <key_code>
```

NOTE

The Shift key is not accessible over the HP-IB. Shifted keys have their own codes as listed in the following tables.

Key Codes

Key	Key Code	Key	Key Code	Key	Key Code
0	43	5	3	NANO	45
1	35	6	12	PICO	53
1A	129	7	19	POINT	52
1B	133	8	11	PRINT	10
1C	137	9	4	RIGHT ARROW	5
1D	141	ACQUISITION	40	RUN	50
2	60	APPLICATION	0	SETUP	18
2A	130	AUTOSCALE	2	SIGN	28
2B	134	CLEAR DISPLAY	58	Softkey 0 (top key)	59
2C	138	CLEAR	44	Softkey 1	1
2D	142	DEFINE MEASURE	16	Softkey 2	9
3	20	DISK	34	Softkey 3	17
3A	131	DISPLAY	32	Softkey 4	25
3B	135	EEX	36	Softkey 5	33
3C	139	ENTER	21	Softkey 6 (bottom key)	41
3D	143	HELP	57	STOP SINGLE	42
4	27	LEFT ARROW	13	TIMEBASE	56
4A	132	MARKER	24	TRIGGER	48
4B	136	MATH	8	UTILITY	49
4C	140	MICRO	37	WAVEFORM	26
4D	144	MILLI	29		

Key Codes for Shifted Keys (64 + unshifted key value)

Key	Key Code	Key	Key Code	Key	Key Code
–WIDTH	67	HISTOGRAM	96	SETUP_PRINT	74
+WIDTH	91	LIMIT TEST	80	UNDO_AUTOSCALE	66
CANCEL PRINT	82	LOCAL	106	V AMPTD	107
CLEAR MEASURE	108	MASK TEST	88	V BASE	116
DELTA TIME	93	MEASURE EYE	104	V MAX	84
DUTY CYCLE	76	MORE MEAS	109	V MIN	124
FALLTIME	75	OVERSHOOT	117	V P-P	99
FFT	72	PERIOD	85	V RMS	101
FINE	77	PRESHOOT	100	V TOP	92
FREQUENCY	68	RISETIME	83		

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 responses from the instrument to the short form (abbreviated spelling).

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

Query

```
:SYSTem:LONGform?
```

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

Returned Format

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

Example

The following example checks the current format for query responses from the oscillscope and places the result in the string variable, Result$, then it prints the contents of the variable to the controller's screen.

```
10  DIM Result$[50]           !Dimension variable
20  OUTPUT 707;":SYSTEM:LONGFORM?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

SETup

Command

```
:SYSTem:SETup <binary block data>
```

Sets up the instrument as defined by the data in the setup string from the controller.

Example

The following example sets up the instrument as defined by the setup string stored in the variable, Set$.

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

NOTE

is an HP BASIC image specifier that suppresses the automatic output of the EOI sequence following the last output item.

K is an HP BASIC image specifier that outputs a number or string in standard form with no leading or trailing blanks.

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.

Returned Format

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

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

Example

The following example stores the current instrument setup in the string variable, Set$.

```
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
```

NOTE

–K is an HP 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.

See Also

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.

TIME

Command

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

Sets the time in the instrument and is not affected by the *RST common command.

Example

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

Query

```
:SYSTem:TIME?
```

The query returns the current time in the instrument.

Returned Format

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

5

Acquire Commands

Acquire Commands

The Acquire subsystem commands set up conditions for executing a DIGITIZE root level command to acquire waveform data. The commands in this subsystem select the type of data, the number of averages, and the number of data points.

The Acquire subsystem contains the following commands and queries:

- AVERage
- BEST (*HP 54750A only*)
- COUNt
- POINts

AVERage

Command

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

Controls averaging.

Example

The following example turns averaging on.

```
10  OUTPUT 707;":ACQUIRE:AVERAGE ON"
20  END
```

Query

```
:ACQuire:AVERage?
```

The query returns the averaging state.

Returned Format

```
[:ACQuire:AVERage] {ON | OFF}<NL>
```

Example

The following example places the current setting of the bandwidth limit filter in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":ACQUIRE:AVERAGE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

BEST (HP 54750A Only)

Command

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

When averaging is turned ON, the flatness option invokes an algorithm within the instrument to improve the step flatness. This mode should be used when performing TDR measurements or when step flatness is important. Thruput mode is faster and can be used when best flatness is not required.

COUNt

Command

```
:ACQuire:COUNt <value>
```

Sets the ensemble count for ensemble-type 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> An integer, 1 to 4096, specifying the number of data values to be averaged.

Example The following example specifies that 16 data values must be averaged for each time bucket to be considered complete.

```
10  OUTPUT 707;":ACQUIRE:COUNT 16"
20  END
```

Query

```
:ACQuire:COUNt?
```

The query returns the currently selected count value.

Returned Format

```
[:ACQuire:COUNt] <value><NL>
```

<value> An integer, 1 to 4096, specifying the number of data values to be averaged.

Example The following example checks the currently selected count value and places that value in the string variable, Result$. Then the program prints the contents of the variable to the controller's screen.

```
10  DIM Result$[50]          !Dimension variable
20  OUTPUT 707;":ACQUIRE:COUNT?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

POINts

Command

```
:ACQuire:POINts {AUTO | <points_value>}
```

Specifies the requested record length for an acquisition. Use the WAVeform:POINts query or WAVeform:PREamble query to determine the actual number of acquired points. The points value can be set to AUTO, which allows the instrument to select the number of points. If the color graded display mode or histograms are used, the record length should be an integer multiple of 450 points.

<points_value>

An integer representing the record length. The points value range is 16 to 4096 points.

Example

The following example sets the record length to 500 points.

```
10  OUTPUT 707;":ACQUIRE:POINTS 500"
20  END
```

Query

```
:ACQuire:POINts?
```

The query returns the requested record length.

Returned Format

```
[:ACQuire:POINts] <points_value><NL>
```

Example

The following example checks the current setting for record length and places the result in the string variable, Length$. Then the program prints the contents of the variable to the controller's screen.

```
10  DIM Length$[50]           !Dimension variable
20  OUTPUT 707;":ACQUIRE:POINTS?"
30  ENTER 707;Length$
40  PRINT Length$
50  END
```

See Also

:WAVeform:DATA command.

6

Calibration Commands

Calibration Commands

This section briefly explains the calibration of the HP 54750A digitizing oscilloscope and the HP 83480A digital communications analyzer. It is intended to give you or the calibration lab personnel an understanding of the various calibration levels available, and how they were intended to be used. Also, this section acquaints you with the terms used in this manual, the Help screens, and the datasheets.

The Calibration subsystem commands initiate the instrument calibration over the HP-IB. The Calibration subsystem consists of the following commands and queries:

- FRAMe
  - CANCel
  - CONTinue
  - DATA
  - DONE?
  - LABel
  - MEMory?
  - STARt
  - TIME?
- OUTPut
- PLUGin
  - ACCuracy
  - CANCel
  - CONTinue
  - DONE?
  - MEMory?
  - OFFSet
  - OPOWer
  - OPTical
  - OWAVelength
  - TIME?
  - VERTicle
- PROBe
- SAMPlers
- SKEW
- STATus

Mainframe Calibration

Mainframe calibration allows the instrument to establish the horizontal calibration factors independent of the plug-ins. These factors are stored in the instrument's nonvolatile RAM. Mainframe calibration is initiated from the "Utility/Calibrate/Calibrate Frame".

Mainframe calibration should be done periodically (at least annually), or if the ambient temperature since the last mainframe 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 "Utility/Calibrate/Calibrate Status On" menu. It is the line labeled "Current Frame Δ Temp: _°C."

The equipment needed to perform mainframe calibration is:

- A plug-in with an electrical channel and a trigger input
- A BNC cable
- An HP 8664A, 8665A, or 8665B signal generator
- An SMA cable

When the calibration is initiated, instructions appear on the screen on how to connect the various pieces of equipment to perform the calibration.

There is a switch on the back panel of the instrument that allows the mainframe calibration to be enabled or disabled. The rear panel has a drawing that shows the function of each switch and which is the protected position. To prevent access to the mainframe calibration switch, place a sticker over the access hole to this switch.

See Also

The Service Guide has more details about the mainframe calibration and the position of the rear-panel memory protect switches.

Plug-in Calibration

Plug-in calibration allows the instrument to establish the calibration factors for a particular plug-in, independent of the mainframe in which it is calibrated. These calibration factors are stored in the plug-in's EEPROM, so the factors stay with the plug-in, not with the mainframe in which the plug-in was calibrated. Plug-in calibration is initiated from the "Utility/Calibrate/Calibrate Plug-in" menu.

Plug-in calibrations should be done annually or if the plug-in has been repaired or reprogrammed.

For maximum measurement accuracy, the calibration procedure should be repeated whenever a plug-in is removed, or whenever another plug-in is removed or installed.

The equipment needed to perform a plug-in calibration is a BNC cable (1 to 2 ft) and a female-BNC-to-male-SMA adapter.

Probe Calibration

Probe calibration allows the instrument to establish the gain and offset of a probe that is connected to a channel of the instrument, and apply those factors to the calibration of that channel. Probe calibration is initiated from the "Channel/Calibrate/Calibrate Probe" menu. Probe calibration commands are part of the Channel subsystem.

For active probes that the instrument can identify through the probe power connector, like the HP 54701A, the instrument automatically adjusts the vertical scale factors for that channel even if a probe calibration is not performed. For passive probes or non-identified probes, the instrument adjusts the vertical scale factors only if a probe calibration is performed. If you do not perform a probe calibration, but want to use a passive probe, the attenuation factor can be entered in the "Channel/Calibrate/Probe atten" menu.

If the probe being calibrated has an attenuation factor that allows the instrument to adjust the gain (in hardware) to produce even steps in the vertical scale factors, the instrument will do so. If the probe being calibrated has an unusual attenuation, like 3.75, the instrument may have to adjust the vertical scale factors to an unusual number, like 3.75 V/div. Typically, probes have standard attenuation factors such as divide by 10, divide by 20, or divide by 100.

FRAMe:CANCel

Command

```
:CALibrate:FRAMe:CANCel
```

Equivalent to pressing the Cancel softkey when in the front-panel Calibrate Frame menu. This command cancels the calibration on the instrument mainframe.

FRAMe:CONTinue

Command

```
:CALibrate:FRAMe:CONTinue
```

Equivalent to pressing the front-panel Continue softkey when in the Calibration Frame menu. This command continues the calibration on the instrument mainframe.

FRAMe:DATA

Command

```
:CALibrate:FRAMe:DATA <cal factors>
```

Sends the FRAMe data to the instrument.

<cal factors> Is a <long_block> containing a copy of the Frame calibration structure.

Query

```
:CALibrate:FRAMe:DATA?
```

The query returns the calibration factors for the instrument mainframe.

Returned Format

```
[CALibrate:FRAMe:DATA] <cal factors>
```

<cal factors> Is a <long_block> containing a copy of the Frame calibration structure.

FRAMe:DONE?

Query

```
:CALibrate:FRAMe:DONE?
```

The query returns the pass/fail status of the last frame calibration.

Returned Format

```
[:CALibrate:FRAMe:DONE] {1 | 0}
```

FRAMe:LABel

Command

```
:CALibrate:FRAMe:LABel <label>
```

Accepts a string of up to 80 characters. It is displayed as part of the frame calibration status screen and is optional. It is intended for user notes, such as name/initials of the calibrator or special notes about the calibration.

<label> Is a <quoted string>.

Query

```
:CALibrate:FRAMe:LABel?
```

The query returns the currently defined label for the frame.

Returned Format

```
[CALibrate:FRAMe:LABel]<quoted string><NL>
```

FRAMe:MEMory?

Query

```
:CALibrate:FRAMe:MEMory?
```

The query returns the state of the frame calibration write-protect switch.

Returned Format

```
[:CALibrate:FRAMe:MEMory]{PROTected|UNPRotected}<NL>
```

FRAMe:STARt

Command

```
:CALibrate:FRAMe:STARt
```

Starts the annual calibration on the instrument mainframe.

FRAMe:TIME?

Query

```
:CALibrate:FRAMe:TIME?
```

The query returns the date, time and temperature at which the last full frame calibration process was completed.

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
-12C
```

OUTPut

Command

```
:CALibrate:OUTPut <value>
```

Sets the dc level of the calibrator signal output through the front-panel CAL connector.

<value> Is a dc level value in volts.

Query

```
:CALibrate:OUTPut?
```

The query returns the current dc level of the calibrator output.

Returned Format

```
[:CALibrate:OUTPut] <value>
```

PLUGin:ACCuracy

Command

```
:CALibrate:PLUGin:ACCuracy{SLOT1 | SLOT2 | SLOT3 | SLOT4}
```

Returns the plug-in calibration status. It will return UNKNOWN if the plug-in is not calibrated or unknown.

Returned Format

```
[:CALibrate:PLUGin:ACCuracy}{SLOT1 | SLOT2 | SLOT3 | SLOT4}
```

PLUGin:CANCel

Command

```
:CALibrate:PLUGin:CANCel
```

Equivalent to pressing the front-panel Cancel softkey when in the Calibrate Plugin menu. This command cancels the calibration on a selected slot (plug-in).

PLUGin:CONTinue

Command

```
:CALibrate:PLUGin:CONTinue
```

Equivalent to pressing the front-panel Continue softkey when in the Calibrate Plug-in menu. This command continues the calibration on a selected slot (plug-in).

PLUGin:DONE?

Query

```
:CALibrate:PLUGin:DONE?
```

The query returns the pass/fail status of the last plug-in calibration.

Returned Format

```
[:CALibrate:PLUGin:DONE] {1 | 0}
```

PLUGin:MEMory?

Query

```
:CALibrate:PLUGin:MEMory? {SLOTN}
```

The query returns the state of the plug-in memory write-protect switch for the selected slot. The switch must be in the UNPRotected state in order to calibrate the plug-in, or to program the module serial number. If no plug-in is present in the selected slot, the message `Empty Slot` is returned.

Returned Format

```
[:CALibrate:PLUGin:MEMory] {PROTected|UNPRotected}<NL>
```

PLUGin:OFFSet

Command

```
:CALibrate:PLUGin:OFFSet {CHAN1 | CHAN2 | CHAN3 | CHAN4}
```

Initiates an offset calibration on the selected channel. The selected channel must be an optical channel.

Example

HP-IB sequence for offset cal:

```
10 OUTPUT 707;":CAL:PLUG:OFFS CHAN1"
20 OUTPUT 707;":CAL:PLUG:CONT"
30 END
```

PLUGin:OPOWer

Command

```
:CALibrate:PLUGin:OPOWer
```

Sets the optical power level for an optical channel plug-in calibration. This command should only be used for plug-ins with an optical channel.

Example

```
10 OUTPUT 707;":CAL:PLUG:OPOW 500E-6"
20 END
```

PLUGin:OPTical

Command

```
:CALibrate:PLUGin:OPTical {CHAN1 | CHAN2 | CHAN3 | CHAN4},
[W1310 | W1550 | udef]
```

Initiates an O/E calibration on the selected channel. The selected channel must be an optical channel. The wavelength reflection after the channel selection is optional and should generally not be specified. The only time the wavelength should be given is to overwrite the factory O/E calibration.

Example

The following sequence of commands perform an optical channel calibration on optical channel 1.

```
10 OUTPUT 707;":CAL:PLUG:OPT CHAN1"
20 OUTPUT 707;":CAL:PLUG:OWAV 1340E-9"
30 OUTPUT 707;":CAL:PLUG:CONT"
40 OUTPUT 707;":CAL:PLUG:OPOW 500E-6"
50 OUTPUT 707;":CAL:PLUG:CONT"
<disconnect all inputs>
60 OUTPUT 707;":CAL:PLUG:CONT"
<connect laser to channel 1>
70 OUTPUT 707;":CAL:PLUG:CONT"
80 END
```

PLUGin:OWAVelength

Command

```
:CALibrate:PLUGin:OWAVelength <wavelength>
```

Sets the optical wavelength for an optical channel measurement. This command should only be used for plug-ins with an optical channel.

Example

```
10 OUTPUT 707;":CAL:PLUG:OWAV 1340E-9"
20 END
```

PLUGin:TIME?

Query

```
:CALibrate:PLUGin:TIME? {SLOTN}
```

The query returns the date, time and temperature at which the plug-in, in slot N, was last calibrated. If no plug-in is present in the selected slot, the message, `Empty Slot`, is returned.

Returned Format

```
[:CALibrate:PLUGin:TIME] <value>
```

<value> 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
-12C
```

PLUGin:VERTical

Command

```
:CALibrate:PLUGin:VERTical {SLOT1 | SLOT2 | SLOT3 | SLOT4}
```

Initiates a vertical calibration on a selected slot. The specified slot should be the first slot of a double-wide module.

Example HP-IB sequence for vertical calibration:

```
10 OUTPUT 707;":CAL:PLUG:VERT SLOT1"
<disconnect all inputs>
20 OUTPUT 707;":CAL:PLUG:CONT"
30 END
```

PROBe

Command

```
:CALibrate:PROBe {CHAN 1 | CHAN 2 | CHAN 3 | CHAN 4}
```

Starts the probe's calibration for the selected channel.

Example

The following example starts the calibration for channel 1.

```
10 OUTPUT 707;":CALIBRATE:PROBE CHAN1"
20 END
```

SAMPlers

Command

```
:CALibrate:SAMPlers {DISable | ENABle}
```

Enables or disables samplers for service. (See Service Guide.)

SKEW

Command

```
:CALibrate:SKEW {CHANnelN},<skew>
```

Sets the channel-to-channel skew factor for channel N. The numerical argument is a real number in seconds which is to be added to the current timebase position to effectively shift the position of CHANnel N's data in time. It can be used to compensate for differences in the electrical lengths of input paths due to cabling and probes.

<skew>

Is a real number.

Query

```
:CALibrate:SKEW? {CHANnelN}
```

The query returns the current skew factor.

Returned Format

```
[:CALibrate:SKEW] <skew_factor><NL>
```

STATus?

Query

```
:CALibrate:STATus?
```

The query returns the calibration status of the instrument. These are nine, comma-separated, 32-bit integers. The data are available from the front panel via the Display Frame Calibration screen accessed from the Utility calibration menu.

Returned Format

```
[:CALibrate:STATUs] <status>
```

<status>

```
<Frame Status>, <Slot1 Vertical>, <Slot1 Trigger>, <Slot2 Vertical>,
<Slot2 Trigger>, <Slot3 Vertical>, <Slot3 Trigger>, <Slot4 Vertical>,
<Slot4 Trigger>
```

7

Channel Commands

Channel Commands

The Channel subsystem commands control all vertical (Y axis) functions of the instrument. The options for the channel subsystem commands vary, depending on which plug-in you are using.

The channel displays may be toggled on and off with the root level commands VIEW and BLANK.

The Channel subsystem contains the following commands and queries:

- AUTOscale
- BANDwidth
- DISPlay
- FDEScription
- FILTer
- FSELect
- OFFSet
- PROBe
  - CALibrate
- RANGe
- SCALe
- SKEW
- UNITs
  - ATTenuation
  - OFFSet
- WAVelength

AUToscale

Command

```
:CHANnel<number>:AUToscale
```

Evaluates the input signal and determines the optimum vertical scale for displaying the signal.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

The following example evaluates the input signal on channel 1.

```
10 OUTPUT 707;":CHANNEL1:AUToscale"
20 END
```

BANDwidth

Command

```
:CHANnel<number>:BANDwidth {HIGH | 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. See the plug-in manual for cutoff frequency specifications.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

The following example sets the channel 1 bandwidth to "HIGH".

```
10 OUTPUT 707;":CHANNEL1:BANDwidth HIGH"
20 END
```

Query

```
:CHANnel<number>:BANDwidth?
```

The query returns the state of the bandwidth for the specified channel.

Returned Format

```
[:CHANnel<number>:BANDwidth] {HIGH | LOW}
```

Example

The following example places the current setting of the channel bandwidth in the string variable, Band$, and then prints the contents of the variable to the controller's screen.

```
10  DIM Limit$[50]            !Dimension variable
20  OUTPUT 707;":CHANNEL1:BANDwidth?"
30  ENTER 707;Band$
40  PRINT Band$
50  END
```

DISPlay

Command

```
:CHANnel<number>:DISPlay {ON | 1 | OFF | 0}
```

Sets the specified channel to ON or OFF.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

The following example sets channel 1 display to ON.

```
10 OUTPUT 707;"CHANNEL1:DISPLAY ON"
20 END
```

Query

```
:CHANnel<number>:DISPlay?
```

The query returns the current display condition for the specified channel.

Returned Format

```
[:CHANnel<number>:DISPlay] {1 | 0}
```

Example

The following example places the current setting of the channel 1 display in the string variable, Display$, and then prints the contents of the variable to the controller's screen.

```
10  DIM Display$[50]                !Dimension variable
20  OUTPUT 707;":CHANNEL1:DISPLAY?"
30  ENTER 707;Display$
40  PRINT Display$
50  END
```

FDEScription

Requires firmware revision A.04.00 and above.

Query

```
:CHANnel<number>:FDEScription?
```

The query returns the number of filters and a brief description of each filter for channels with more than one internal low-pass filter.

The filter description is the same as the softkey label for the control used to select the active filter.

Returned Format

```
[:CHANnel<n>] <,filter1_description> <,filter2_description>
```

<n>

```
number of filters
```

<filter_description>

```
XXX b/s:N
XXX is bit rate of filter
N is filter order
```

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

FILTer

Command

```
:CHANnel<number>:FILTer {ON | 1 |  OFF | 0}
```

Controls an internal low-pass filter, if one is present, in the channel hardware.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

```
10 OUTPUT 707;"CHANNEL1:FILTER ON"
20 END
```

Query

```
:CHANnel<number>:FILTer?
```

The query returns the filter setting for the specified channel.

Returned Format

```
[:CHANnel<number>:FILTer] {1 | 0}
```

Example

The following example places the current setting of the filter in the string variable, Filter$, and then prints the contents of the variable to the controller's screen.

```
10  DIM Filter$[50]                !Dimension variable
20  OUTPUT 707;":CHANNEL1:FILTER?"
30  ENTER 707;Filter$
40  PRINT Filter$
50  END
```

FSELect

Requires firmware revision A.04.00 and above.

Command

```
:CHANnel<number>:FSELect FILTer <filter_number>
```

Selects which filter is controlled by on/off for channels with more than one internal low-pass filter.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

<filter_number>

Integer 1 or 2

Example

```
10  OUTPUT 707;"CHANNEL1:FSELECT FILTER1"
20  END
```

Query

```
:CHANnel<number>:FSELect?
```

The query returns the current filter number for the specified channel.

Returned Format

```
[CHANnel<number>:FSELect] {FILT <filter_number>}
```

Example

The following example places the current setting of the filter in the string variable, Filter$, and then prints the contents of the variable to the controller's screen.

```
10 DIM Filter$[50]
20 OUTPUT 707;"CHANNEL1:FSELECT?"
30 ENTER 707;Filter$
40 PRINT Filter$
50 END
```

OFFSet

Command

```
:CHANnel<number>:OFFSet <offset value>
```

Sets the voltage that is represented at center screen (2 division from bottom for optical channels) for the selected channel. Offset parameters are plug-in, probe, and vertical scale dependent.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

<offset value>

Offset value at center screen. 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.

Example

The following example sets the offset for a plug-in to 125 m in the current measurement units:

```
10  OUTPUT 707;":CHANNEL1:OFFSET 125E-03"
20  END
```

Query

```
:CHANnel<number>:OFFSet?
```

The query returns the current offset value for the specified channel.

Returned Format

```
[CHANnel<number>:OFFSet] <offset value><NL>
```

Example

The following example places the offset value of the specified channel in the string variable, Offset$, and then prints the contents of the variable to the controller's screen.

```
10 DIM Offset$[50]
20 OUTPUT 707;"CHANNEL1:OFFSET?"
30 ENTER 707;Offset$
40 PRINT Offset$
50 END
```

NOTE

When you are receiving numeric data into numeric variables, the headers should be turned off. Otherwise, the headers may cause misinterpretation of returned data.

PROBe

Command

```
:CHANnel<number>:PROBe <attenuation factor>[,{RATio | DECibel}]
```

Sets the probe attenuation factor and, optionally, the units for the probe attenuation factor.

The range of the probe attenuation factor is from 0.0001 to 1000000 and −80 dB to 120 dB. The reference constants that are used for scaling the display factors are changed with this command, including automatic measurements and trigger levels.

<number> The channel number represents an integer, 1 to 4 followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

<attenuation factor> A real number from 0.0001 to 1000000, or −80 dB to 120 dB representing the probe attenuation factor and depends on the units.

Example The following example sets the probe attenuation factor of channel 1 to 10 and the units to decibel.

```
10  OUTPUT 707;":CHANNEL1:PROBE 10, DEC"
20  END
```

See Also

For information on skew, see CALibrate commands.

Query

```
:CHANnel<number>:PROBe?
```

The query returns the current probe attenuation setting for the selected channel, and the units.

Returned Format

```
[:CHANnel<number>:PROBe]<attenuation>,{RATio | DECibel}<NL>
```

Example

The following example places the current attenuation setting for channel 1 in the string variable, Atten$, and then the program prints the contents.

```
10  DIM Atten$[50]       !Dimension variable
20  OUTPUT 707;":SYSTEM:HEADER OFF"
30  OUTPUT 707;":CHANNEL1:PROBE?"
40  ENTER 707;Atten$
50  PRINT Atten$
60  END
```

If you use a string variable, the query returns the attenuation value and the factor (decibel or ratio). If you use an integer variable, the query returns the attenuation value. You must then read the attenuation units into a string variable.

PROBe:CALibrate

Command

```
:CHANnel<number>:PROBe:CALibrate
```

Starts the probe's calibration for the selected channel.

Example

The following example starts calibration for Channel 1.

```
10  OUTPUT 707;":CHANNEL1:PROBE:CALIBRATE"
20  END
```

RANGe

Command

```
:CHANnel<number>: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.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

<range value>

Full-scale voltage of the specified channel number. Available ranges are dependent on the plug-in being used and attenuation factor.

Example

The following example sets the full-scale range for channel 1 to 500 mV.

```
10  OUTPUT 707;":CHANNEL1:RANGE 500E-3"
20  END
```

Query

```
:CHANnel<number>:RANGe?
```

The query returns the current full scale vertical axis setting for the selected channel.

Returned Format

```
[:CHANnel<number>:RANGe]<range value><NL>
```

Example

The following example places the current range value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:RANGE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

SCALe

Command

```
:CHANnel<number>:SCALe <scale value>
```

Defines the vertical scale of the channel in units per division. This command is the same as the front-panel channel scale.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

<scale value>

Vertical scale of the channel in units per division.

Example

The following example sets the scale value for channel 1 to 500 mV.

```
10  OUTPUT 707;":CHANNEL1:SCALE 500E-3"
20  END
```

Query

```
:CHANnel<number>:SCALe?
```

The query returns the current scale setting for the specified channel.

Returned Format

```
[:CHANnel<number>:SCALe] <scale value><NL>
```

Example

The following example places the current scale value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:SCALE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

SKEW

Command

```
:CHANnel<number>:SKEW
```

Adjusts a channel in time relative to another channel. This is useful for aligning channels relative to another and to account for different delays through cables relative to the trigger.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Query

```
:CHANnel<number>:SKEW?
```

Returns the current skew setting for the channel.

Returned Format

```
[:CHANnel<number>:SKEW?] <skew_value><NL>
```

Example

The following example sets the channel 1 skew, relative to the trigger, to 300 ps:

```
10 OUTPUT 707;":CHANNEL1:SKEW 300E-12"
20 END
```

UNITs

Command

```
:CHANnel<number>:UNITs {VOLT | AMPere | WATT | UNKNown}
```

Allows you to work in vertical units other than volts. The units command changes the name of the Y-axis units from VOLT to AMP, WATT, or UNKNown. The units are implied for other pertinent Channel commands (such as RANGe and OFFSet).

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

The following example sets the units for channel 1 to amperes.

```
10  OUTPUT 707;":CHANNEL1:UNITS AMPERE"
20  END
```

Query

```
:CHANnel<number>:UNITs?
```

The query returns the current units setting for the specified channel.

Returned Format

```
[:CHANnel<number>:UNITs] {VOLT | AMPere | WATT | UNKNown}<NL>
```

Example

The following example places the vertical units for the specified channel in the string variable, Units$, and then prints the contents of the variable to the controller's screen.

```
10 DIM Units$[50]
20 OUTPUT 707;"CHANNEL1:UNITS?"
30 ENTER 707;Units$
40 PRINT Units$
50 END
```

UNITs:ATTenuation

Command

```
:CHANnel<number>:UNITs:ATTenuation <attenuation>
```

Specifies a ratio that indicates how the unit specified in the :CHANnel<number>:UNITs command relates to one volt.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

<attenuation>

The number of volts per unit, such as volts/watt.

Example

The following example specifies channel 1 units as watts and that the device under test supplies the channel input with 30 volts for every watt.

```
10 OUTPUT 707;"CHANNEL1:UNITS WATT"
20 OUTPUT 707;"CHANNEL1:UNITS:ATTENUATION 30"
30 END
```

Query

```
:CHANnel<number>:UNITs:ATTenuation?
```

The query returns the number of volts per user specified units.

Returned Format

```
[CHANnel<number>:UNITs:ATTenuation]<attenuation>
```

Example

The following example places the current unit attenuation value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:UNITS:ATTENUATION?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

UNITs:OFFSet

Command

```
:CHANnel<number>:UNITs:OFFSet <offset>
```

Specifies a constant value in volts that will be added to the signal to compensate for any offset. For example, if the units are watts and the measured signal has 0.5 volts offset at 0 watts, then the argument for this command should be −0.5 to ensure that 0 watts will appear at center screen when CHANnel:OFFSet is set to 0.

<number>

The channel number represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

<offset>

The constant value in volts to be added to the measured signal.

Example

The following example sets the offset compensation value for the assigned units of channel 1 to 0.1 volts.

```
10  OUTPUT 707;":CHANNEL1:UNITS:OFFSET 0.1"
20  END
```

Query

```
:CHANnel<number>:UNITs:OFFSet?
```

This query returns the current units offset value for the specified channel.

Returned Format

```
[:CHANnel<number>:UNITs:OFFSet] <offset>
```

Example

The following example places the current unit offset value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:UNITS:OFFSET?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

WAVelength

Command

```
:CHANnelN:WAVelength {W1310 | W1550 | USER}
```

Sets the wavelength selection for optical channels. Two factory calibrations have been performed for 1310 nm and 1550 nm. Invoke these calibrations using W1310 or W1550, respectively. 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.

Query

```
:CHANnelN:WAVelength?
```

The query returns the currently selected wavelength for the channel.

Returned Format

```
[:CHANnelN:WAVelength] {W1310 | W1550 | USER}
```

Example

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANnel1:WAVELENGTH?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

8

Disk Commands

Disk Commands

The Disk subsystem commands perform the disk operations as defined under the Disk menu. This allows storage and retrieval of waveforms, setups and pixel memory, as well as, formatting the disk.

The filenames used for files are compatible with DOS. They consist of up to 8 characters for the name with a 0 to 3 character extension separated by a "." (dot). File names are all uppercase but can be entered in either upper or lowercase and will be forced to uppercase internally. Valid characters are: A-Z, 0-9, _ (the underscore character).

NOTE

The filename must be enclosed in quotes.

The Disk subsystem contains the following commands and queries:

- DELete
- DIRectory?
- FORMat
- LOAD
- STORe

DELete

Command

```
:DISK:DELete "<filename>"
```

Deletes a file from the disk. An error is displayed on the screen if the requested file does not exist.

<filename> A DOS compatible filename. Up to 8 characters with a 3 character extension.

Example

```
10 OUTPUT 707;":DISK:DELETE ""FILE1.SET"""
20 END
```

DIRectory?

Query

```
:DISK:DIRectory?
```

The query returns the directory listing of the currently installed disk. Each entry is 63 bytes long, including a carriage return and line feed.

Returned Format

```
[:DISK:DIRectory] <directory>
```

<directory> List containing <file1.ext> <type> <size> <date> <time> for each entry.

<type>

```
{SETUP | WAVEFORM | WF_TEXT | PIXEL | TIFF | GIF | SUMMARY
| PCX | PCL | EPSON | MASK | DATABASE | DOS | ASCII | IBPRG}
```

<size> Is an integer.

<date> In the format DD MMM YY

<time> In the format HH:MM

FORMat

Command

```
:DISK:FORMat
```

Formats a disk in the drive. It is assumed that the disk that is to be formatted is in the drive when the command is issued.

Example

```
10 OUTPUT 707;":DISK:FORMAT"
20 END
```

LOAD

Command

```
:DISK:LOAD "<filename>"[,<destination>]
```

Restores a setup, a waveform, a mask, a database, or a pixel memory from the disk. The type of file is determined by the filename suffix, if one is present, or by the destination field if one is not present. The database may only be loaded in internal format.

<filename>

DOS-compatible filename. Up to 8 characters with 0 to 3 character extension. Either .wav, .msk, .wdb, .txt, .pix, or .set may be used as a suffix after the filename. For a database, the file suffix must be .wdb. If no file suffix is specified, the default is .wav.

<destination>

```
{WMEMoryN  |  SETup  |  PMEMory1 | DATAbase | MASK}
```

The maximum size of .txt waveforms is 128K points. Up to two waveform memories may be used when loading in a .txt file for records exceeding 64K points. Up to four waveform memories can be used if the file is in internal format.

Example

```
10 OUTPUT 707;":DISK:LOAD ""FILE1.WAV"",WMEM1"
20 END
```

STORe

Command

```
:DISK:STORe <source>, "<filename>"[,<format>]
```

Stores a setup, a waveform, a database, a mask, or pixel memory to the disk. The filename does not include a suffix. The suffix is supplied by the instrument depending on the source and file format specified. The database may only be saved in internal format.

<source>

```
{CHANnel<number> | FUNCtionN<number> | WMEMory<number> | SETup | PMEMory1
 | HISTogram | DATabase | FFT | MASK}
```

<number>

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For functions: 1 or 2.

For waveforms: 1, 2, 3, or 4.

<filename>

Is an MS-DOS®-compatible name of the file up to 8 characters long.

<format>

```
{TEXT [,<YVALues> | <VERBose>] | INTernal}
```

NOTE

The format field is for waveforms, 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 is stored. See the Waveform commands chapter for information on converting data to values.

Example

```
10 OUTPUT 707;":DISK:STORE SET,""FILE1.SET"""
20 END
```

9

Display Commands

Display Commands

The Display subsystem commands control the display of data, markers, text, graticules, and the use of color.

The display mode is selected by the ACQUIRE:TYPE command and the number of averages is selected by the ACQUIRE:COUNT command.

The Display subsystem contains the following commands:

- ASSign
- CGRade
  - LEVels?
- COLumn
- DATA
- DCOLor (Default COLor)
- DWAVeform (Draw WAVeform)
- FORMat
- GRATicule
- INVerse
- LINE
- MASK
- PERSistence
- ROW
- SCOLor (Set COLor)
- SOURce
- STRing
- TEXT

ASSign

Command

```
:DISPlay:ASSign {CHANnel<number> | FUNCtion<number> |
WMEMory<number> | FFT},{UPPer | LOWer}
```

Assigns the specified waveform, function, or FFT to the specified portion of the waveform area on the screen. This command has no effect when the graticule format is single.

<number>

For channels: 1 through 4, optionally followed by an A or B.

For functions: 1 or 2.

For waveform memories: 1 to 4.

Example

The following example assigns channel 1 to the upper portion of the instrument screen.

```
10  OUTPUT 707;":DISPLAY:ASSIGN CHANNEL1,UPPER"
20  END
```

Query

```
:DISPlay:ASSign? {CHANnel<number>|FUNCtion<number>|
 WMEMory<number> | FFT}
```

The query returns the portion of the waveform area where the specified waveform is displayed or was last displayed if it is not currently on screen.

Returned Format

```
[:DISPlay:ASSign? {CHANnel<number>|FUNCtion<number>|
 WMEMory<number> | FFT} {UPPer |LOWer}<NL>
```

Example

The following example returns the portion of the waveform area where the channel 1 waveform resides to the string variable, Setting$, then it prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]              !Dimension variable
20  OUTPUT 707;":DISPLAY:ASSIGN? CHANNEL1"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

CGRade

Command

```
:DISPlay:CGRade {ON | 1 | OFF | 0}
```

Sets the color-graded display on or off.

When in the color-graded display mode, all signals are mapped into a database and shown with different colors representing varying number of hits in a pixel. "Connected dots" and "high-resolution" display modes are disabled when the color-graded display is on.

The instrument has three features that use a specific database. This database uses a different memory area than the waveform record for each channel. The three features that use the database are histograms, mask testing, and color-graded display. When any one of these three features is turned on, the instrument starts building the database. The database is the size of the graticule area, which is 256 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 DISPlay:CGRade:LEVels? query. The color-graded display uses colors to represent the number of hits on various areas of the display.

The default color-grade state is off.

Example

The following example sets the color-graded display on.

```
10  OUTPUT 707;":DISPLAY:CGRADE ON"
20  END
```

Query

```
:DISPlay:CGRade?
```

The query returns the current color-grade state.

Returned Format

```
[:DISPlay:CGRade] {ON | OFF} <NL>
```

Example

The following example returns the current color-grade state.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:CGRADE?"
30  ENTER 707;Cgrade$
40  PRINT Cgrade$
50  END
```

CGRade:LEVels?

Query

```
:DISPlay:CGRade:LEVels?
```

The query returns the range of hits represented by each color. 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>
```

<color format>

<intensity color min / max> is an integer value from 0 to 65535.

Example

The following example gets the range of hits represented by each color and prints it on the controller screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:CGRADE:LEVELS?"
30  ENTER 707;Cgrade$
40  PRINT Cgrade$
50  END
```

COLumn

Command

```
:DISPlay:COLumn <value>
```

Specifies the starting column for subsequent DISPlay:STRing and DISPlay:LINE commands.

<value>

An integer, 0 to 81, representing the starting column for subsequent STRING and LINE commands. The entire viewing area of the screen is divided into 28 rows by 82 columns, so your text can be written anywhere on the screen.

Example

The following example sets the starting column for subsequent DISPlay:STRing and DISPlay:LINE commands to column 10.

```
10  OUTPUT 707;":DISPLAY:COLUMN 10"
20  END
```

Query

```
:DISPlay:COLumn?
```

The query returns the column where the next DISPlay:LINE or DISPlay:STRing starts.

Returned Format

```
[:DISPlay:COLumn] <value><NL>
```

Example

The following example returns the current column setting to the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:COLUMN?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

DATA

Command

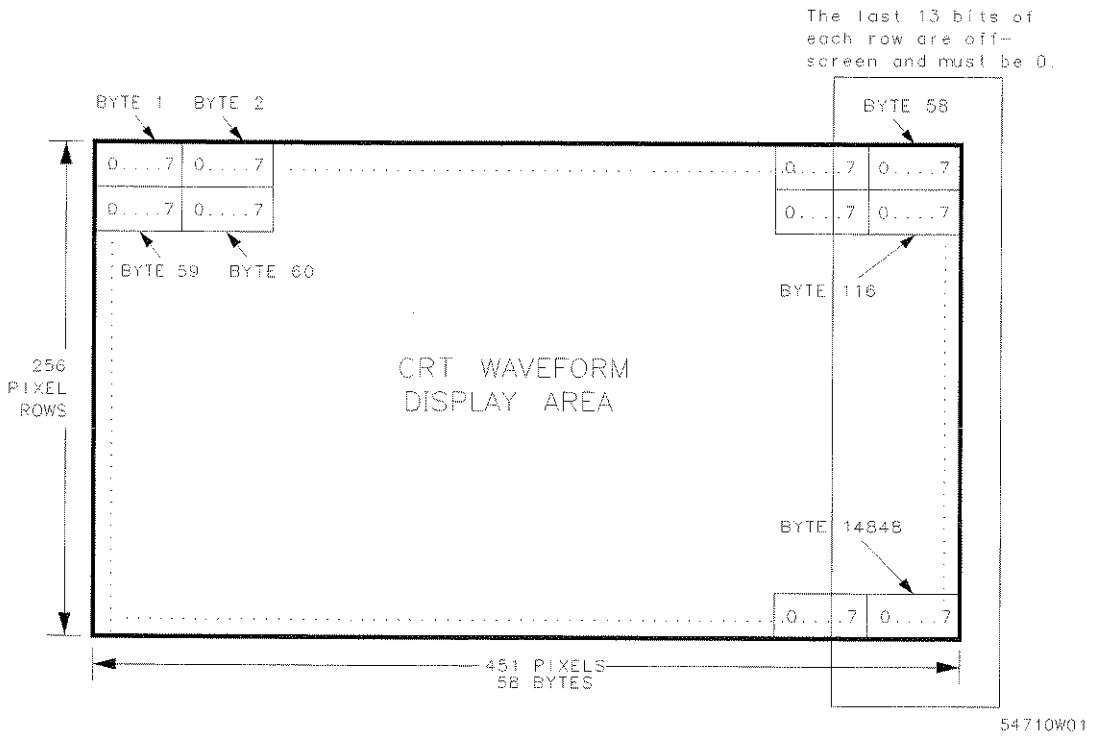
```
:DISPlay:DATA <binary_block_data>
```

Writes waveform data to the pixel memory of the instrument (PMEMory1). The DATA command is followed by a block of binary data that is transferred from the controller to the pixel memory in the instrument.

<Binary_block_data> Data in the IEEE 488.2 definite block format.

Pixel Format

The data that is output by this command is the displayed data from the waveform display area (inside the graticule) of the instrument CRT. When using the DISPLAY:DATA command, the information is output as bytes of pixel data. The waveform display area is 451 pixels horizontally by 256 pixels vertically. Each pixel row is divided into 58 bytes, of eight bits each, as shown in the following figure.



CRT Pixel Data

Data Output Format

The pixel plane is 14,848 bytes representing the waveform display portion of the display. The waveform display portion is from and including the top graticule line, to and including the bottom graticule line, as well as all information inside the graticule. The first 58 bytes of data that are output correspond to the top row of the waveform display. The next 58 bytes are the second row of the waveform display, the third 58 bytes correspond to the third row, and so on. The data is output in rows until the 256th pixel row of data is sent. The 256th row corresponds to the bottom graticule line of the waveform display area.

Each of the pixel rows contain 451 pixels that are sent in 8-bit bytes, and 256 rows of pixel data are sent. Each row is sent as 58 bytes. There are 13 unused bits in each row, because 58 bytes times 8 bits equals 464 bits per row. 256 rows of data are sent, so 58 bytes times 256 rows equals 14,848 bytes sent for each screen of data. The 13 unused bits in each row are off-screen, and must be set to zero when sending data to display.

Query

```
:DISPlay:DATA?
```

The query returns waveform data from the pixel memory of the instrument. The query causes the instrument to output pixel data from the pixel memory. The query also follows the IEEE 488.2 definite block format.

This instrument has one pixel memory.

Returned Format

```
[:DISPlay:DATA] <binary_block_data><NL>
```

DCOLor (Default COLor)

Command

```
:DISPlay:DCOLor
```

Returns the screen colors to the predefined colormap.

Example

This example sends the DCOLor command.

```
10  OUTPUT 707;":DISPLAY:DCOLOR"
20  END
```

DWAVeform (Draw WAVeform)

Command

```
:DISPlay:DWAVeform {FAST | CDOTs | HRESolution}
```

Sets the waveform draw mode to FASt, Connected DOTs, or High RESolution. This command has no effect when the color-graded display is on.

Example

This example sets the waveform draw mode to high resolution.

```
10  OUTPUT 707;":DISPLAY:DWAVEFORM HRESOLUTION"
20  END
```

Query

```
:DISPlay:DWAVeform?
```

The query returns the current waveform draw mode.

Example

The following example places the current setting for the waveform draw mode in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:DWAVEFORM?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

FORMat

Command

```
:DISPlay:FORMat {SINGle | 1 | DUAL | 2}
```

Sets the number of graphs. Sending "1" or SINGle sets the number of graphs to 1. Sending "2" or DUAL sets the number of graphs to 2. Each graph defines a separate graticule area within the waveform display area.

Eight divisions are used for the full scale range in both format modes. Waveforms can be assigned to each area of the screen with the DISPLAY:ASSIGN command.

Example

The following example sets the number of graphs to 2.

```
10  OUTPUT 707;":DISPLAY:FORMAT DUAL"
20  END
```

Query

```
:DISPlay:FORMat?
```

The query returns the current number of display areas on the screen.

Returned Format

```
[:DISPlay:FORMat] {1 | 2}<NL>
```

Example

The following example places the current setting for the display format in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:FORMAT?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

GRATicule

Command

```
:DISPlay:GRATicule {GRID | FRAMe}
```

Selects the type of graticule that is displayed.

Example

The following example sets up the background of the instrument's display with a frame separated into major and minor divisions.

```
10  OUTPUT 707;":DISPLAY:GRATICULE FRAME"
20  END
```

Query

```
:DISPlay:GRATicule?
```

The query returns the type of graticule currently displayed.

Returned Format

```
[:DISPlay:GRATicule] {GRID | FRAMe}<NL>
```

Example

The following example places the current setting of the display graticule in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:GRATICULE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

INVerse

Command

```
:DISPlay:INVerse {ON | 1 | OFF | 0}
```

Determines whether or not text sent with the DISPlay:LINE or DISPlay:STRing command will be written with the INVERSE attribute. If the inverse attribute is on, the text is written in inverse video.

Example

The following example turns the inverse attribute on for future DISPlay:LINE and DISPlay:STRing commands.

```
10  OUTPUT 707;":DISPLAY:INVERSE ON"
20  END
```

Query

```
:DISPlay:INVerse?
```

The query returns the current state of the DISPlay:INVerse command.

Returned Format

```
[:DISPlay:INVerse] {1 | 0}<NL>
```

Example

The following example places the current setting of the inverse attribute in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:INVERSE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

LINE

Command

```
:DISPlay:LINE "<string>"
```

Writes a quoted string parameter to the screen, starting at the location specified by the DISPlay:ROW and DISPlay:COLumn commands.

<string> Any series of ASCII characters enclosed in quotes.

Example The following example writes the message `Test 1` to the screen, starting at the current row and column location.

```
10  OUTPUT 707;":DISPLAY:LINE ""TEST 1"""
20  END
```

Text may be written up to column 81. If the characters in the string parameter do not fill the line, the rest of the line is blanked. If the string is longer than the space available on the current line, the excess characters are discarded. In any case, the ROW is incremented and the COLumn remains the same. The next DISPlay:LINE command will write on the next line of the display. After writing line 27, the last line in the display area, the ROW is reset to 0.

MASK

Command

```
:DISPlay:MASK <mask_value>
```

Inhibits the instrument from writing to selected areas of the screen. The mask parameter is an 8-bit integer in which each bit controls writing to an area of the screen. A "0" inhibits writing to the area represented by the bit, a "1" enables writing to that area.

<mask_value> An integer, 0 to 255, representing the areas of the screen that will be masked off to inhibit the instrument from writing to them, as defined in the table below.

Example The following example inhibits the menu area (bit 6) from being written to by the instrument.

```
10  OUTPUT 707;":DISPLAY:MASK 191"
20  END
```

Query

```
:DISPlay:MASK?
```

The query returns the current value of the mask.

Returned Format

```
[:DISPlay:MASK] <mask_value>
```

<mask_value> An integer, 0 to 255, representing the areas of the screen that are masked off to inhibit the instrument from writing to them. Refer to the following table.

Display Byte Mask

Bit	Weight	Screen Area Affected
7	128	Unused
6	64	Menu Area (right side)
5	32	Time Base Scale Area (graticule frame)
4	16	Measurement Area (bottom five lines)
3	8	Graticule Area
2	4	Unused
1	2	Status Lines (second row)
0	1	Advisory Area (message location, top row)

Example The following example returns the current value of the mask to the string variable, Value$, then prints the contents of the variable to the controller's screen.

```
10  DIM Value$[50]            !Dimension variable
20  OUTPUT 707;":DISPLAY:MASK?"
30  ENTER 707;Value$
```

```
40  PRINT Value$
50  END
```

Text sent with the DISPlay:LINE and DISPlay:STRing commands, or with the SYSTem:DSP command, is not affected by this command.

The purpose of the command is to allow HP-IB text to be written anywhere on screen and to prevent the instrument from overwriting the text through its normal operation.

When an area that has been masked off is unmasked, the instrument immediately redraws that area with the information that would normally appear in that area. If any area of the screen is masked off, then full screen text windows, such as Help, System Configuration, and Calibration status will not be displayed when they are selected (such as with the MENU:HELP command).

The DISPlay:MASK parameters are not reset with a *RST common command. The mask value is reset to 255 whenever a remote-to-local transition occurs.

PERSistence

Command

```
:DISPlay:PERSistence {MINimum | INFinite | <value>}
```

Sets the display persistence. The parameter for this command can be either MINIMUM (zero persistence), INFINITE, or a real number from 0 to 40.0, representing the persistence in seconds.

<value>

A real number, 0 to 40.0, representing the persistence in seconds.

Example

The following example sets the persistence to infinite.

```
10  OUTPUT 707;":DISPLAY:PERSISTENCE INFINITE"
20  END
```

Query

```
:DISPlay:PERSistence?
```

The query returns the current persistence value.

Returned Format

```
[:DISPlay:PERSistence] {MINimum | INFinite |<value>}<NL>
```

Example

The following example places the current persistence setting in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:PERSISTENCE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

ROW

Command

```
:DISPlay:ROW <row_number>
```

Specifies the starting row on the screen for subsequent DISPlay:STRing and DISPlay:LINE commands. The row number remains constant until another DISPlay:ROW command is received, or the row is incremented by the DISPlay:LINE command.

<row_number>

An integer, 0 to 27, representing the starting row for subsequent DISPlay:STRing and DISPlay:LINE commands. The entire screen viewing area is divided into 28 rows by 82 columns, so your text can be written anywhere on the screen.

Example

The following example sets the starting row for subsequent DISPlay:STRing and DISPlay:LINE commands to 10.

```
10  OUTPUT 707;":DISPLAY:ROW 10"
20  END
```

Query

```
:DISPlay:ROW?
```

The query returns the current value of the row.

Returned Format

```
[:DISPlay:ROW] <row_number><NL>
```

Example

The following example places the current value for row in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:ROW?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

SCOLor

Command

```
DISPlay:SCOLor <color_name>,<hue>,<saturation>,<luminosity>
```

<color_name>

```
{PLUGin1 | PLUGin2 | SWForms | TOVerlap | WMARker | MARKer
 | GRATicule | TBKGd | ERRor | ADVisory | STATus | KCONtrol
 | MFRGd | MBKGd | MOUTline | MHILight | MSHadow | MBBKgd
 | MBFRgd | DLABel | DB1..DB7}
```

Changes the colors used on the display of the instrument. The :DISPlay:DCOLor command restores the colors to their factory default settings.

The database colors, DB1..DB7, can be changed only when the color-graded display is on. See the DISPlay:CGRade command.

The color names are defined as follows:

Color Names

Color Name	Definition
PLUGin1	Used for waveforms acquired from slot 1 and for the function 1 waveform.
PLUGin2	Used for waveforms acquired from slot 2 and for the function 2 waveform.
SWForms	Saved waveforms. Used for waveform and pixel memories.
TOVerlap	The trace overlap color. Shown where waveforms of different colors overlap.
WMARker	Timebase window marker. Also used for waveform masks and waveform mask coordinate system markers.
MARKer	General purpose markers (waveform markers, trigger level, histograms, and so on).
GRATicule	Waveform area graticule.
TBKGd	Trace background color. The background color in the graticule area.

Color Names (continued)

Color Name	Definition
ERRor	Error messages. Typically used for messages displayed above the graticule area.
ADVisory	Advisory messages. Used for text displayed above the graticule area.
STATus	Status messages displayed above the graticule.
KCONtrol	Knob control indicator color. Used to indicate control to which knob is slaved.
MFRGd	Menu foreground.
MBKGd	Menu background and graticule border.
MOUTline	Menu outline and screen background outside the graticule area.
MHILight	Menu highlight color and edges of graticule border.
MSHadow	Menu shadow color. Used to give menu graphics a 3-dimensional look.
MBBKgd	Memory bar background color.
MBFRgd	Memory bar foreground color. The color of the memory bar above the graticule.
DLABel	User-defined display labels.
DB1..DB7	The seven ranges of pixel counts for the color-graded display.

<hue> An integer, 0 to 100, that determines the graduation of color. As the hue number increases, the selected color cycles through the color spectrum. There is no difference between hue 0 and hue 100.

<saturation> An integer, 0 to 100, that selects the percentage of the pure color that gets mixed with white. Zero is white and 100 is the maximum saturation of the selected color.

<luminosity> An integer, 0 to 100, that determines the brightness of the selected hue. Zero is black and 100 is the maximum brightness.

Example The following example sets the hue to 50, the saturation to 70, and the luminosity to 90 for the markers..

```
10  OUTPUT 707;":DISPLAY:SCOLOR MARKER,50,70,90"
20  END
```

Query

```
:DISPlay:SCOLor? <color_name>
```

The query returns the hue, saturation, and luminosity for the specified color.

Returned Format

```
[:DISPlay:SCOLor] <color_name>,<hue>,<saturation>,<luminosity><NL>
```

Example

The following example places the current settings for the graticule color in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":DISPLAY:SCOLOR? GRATICULE"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

SOURce

Command

```
:DISPlay:SOURce PMEMory1
```

Specifies the destination or source for the DISPlay:DATA command and query. Pixel memory 1 (PMEMORY1) is the only source available for this command.

Example

The following example selects Pixel memory 1 as the display source.

```
10  OUTPUT 707;":DISPLAY:SOURCE PMEMORY1"
20  END
```

Query

```
:DISPlay:SOURce?
```

The query always returns PMEMory1.

Returned Format

```
[:DISPlay:SOURce] PMEMory1<NL>
```

Example

The following example places the current selection for the display source in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":DISPLAY:SOURCE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

STRing

Command

```
:DISPlay:STRing "<string>"
```

Writes text to the screen of the instrument. The text is written starting at the current row and column settings. If the column limit is reached (81), the excess text is discarded. The DISPlay:STRing command does not increment the row value; however, the DISPlay:LINE command does.

<string>

Any series of ASCII characters enclosed in quotes.

Example

The following example writes the message `Example 1` to the instrument's display starting at the current row and column settings.

```
10  OUTPUT 707;":DISPLAY:STRING ""Example 1"""
20  END
```

TEXT

Command

```
:DISPlay:TEXT BLANk
```

Blanks the user text area of the screen. The user text area includes rows 0 through 27, columns 0 through 81.

Example

The following example blanks the user text area of the instrument's screen.

```
10  OUTPUT 707;":DISPLAY:TEXT BLANK"
20  END
```

10

FFT Commands

FFT Commands

The FFT subsystem commands turn on, scale, and set the FFT (fast Fourier transform) to particular windows. Other FFT commands are located in the Function and Measure subsystems.

- DISPlay
- FREQuency
- MAGNify
- MSPan
- OFFSet
- RANGe
- SOURce
- WINDow

DISPlay

Command

```
:FFT:DISPlay {ON | 1 | OFF | 0}
```

Turns the FFT function on and off.

Example

The following example turns on the FFT display:

```
10  OUTPUT 707;":FFT:DISPLAY ON"
20  END
```

Query

```
:FFT:DISPlay?
```

The query returns the current setting of the FFT display.

Returned Format

```
[:FFT:DISPlay] {1 | 0}<NL>
```

FREQuency

Command

```
:FFT:FREQuency <frequency_value>
```

Sets the center frequency value of the FFT display when MAGNify is on. Magnify is turned on and off with the FFT:MAGNify command.

<frequency_value> A frequency from 0 Hz to 1.0 times the unmagnified span.

Example The following example sets the FFT center frequency value to 25 MHz.

```
10 OUTPUT 707;":FFT:FREQUENCY 25E6"
20 END
```

Query

```
:FFT:FREQuency?
```

The query returns the current center frequency value of the FFT set with the command.

Returned Format

```
[:FFT:FREQuency] <frequency_value>><NL>
```

Example The following example places the current setting of the center frequency value of the FFT in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:FREQUENCY?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

MAGNify

Command

```
:FFT:MAGNify {ON | 1  | OFF | 0}
```

Turns the magnification mode of the FFT on and off. When magnify is on, changing MSPAN and FREQuency results in software magnification of the display. The hardware setup is not changed.

Example

```
10 OUTPUT 707;":FFT:MAGNIFY ON"
20 END
```

Query

```
:FFT:MAGNify?
```

The query returns the current setting of the magnification mode.

Returned Format

```
[:FFT:MAGNify] { 1 | 0}<NL>
```

Example

The following example places the current setting of the magnify mode in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:MAGNIFY?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

MSPan

Command

```
:FFT:MSPan <magnify_span_value>
```

Sets the span when magnify is on. Magnify is controlled with the FFT:MAGNify command. MSPAN causes software magnification of the FFT display. The hardware setup is not affected.

<magnify_span_value> Span frequency in Hertz. Allowable values range from 1 to 1/200 Hz plus the value of (unmagnified) SPAN.

Example The following example sets the FFT magnify span value to 25 MHz.

```
10 OUTPUT 707;":FFT:MSPAN 25E6"
20 END
```

Query

```
:FFT:MSPan?
```

The query returns the current setting of the magnify span value.

Returned Format

```
[:FFT:MSPan] <magnify_span_value> <NL>
```

Example The following example places the current setting of the magnify span value in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:MSPAN?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

OFFSet

Command

```
:FFT:OFFSet <offset_value>
```

Sets the dBm offset value represented at the center of the screen for the FFT display. This command controls software magnification of the FFT display. It does not affect the hardware settings.

<offset_value> Offset value in dBm.

Query

```
:FFT:OFFSet?
```

The query returns the offset value.

Returned Format

```
[:FFT:OFFSet] <offset_value><NL>
```

RANGe

Command

```
:FFT:RANGe <range_value>
```

Sets the vertical range value for the FFT. This command controls software scaling of the FFT display. The hardware settings are not affected.

<range_value> Full-scale vertical range in dB.

Query

```
:FFT:RANGe?
```

The query returns the current vertical range setting for the FFT.

Returned Format

```
[:FFT:RANGe] <range_value><NL>
```

Example The following example places the current setting of the range value in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:RANGE?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

SOURce

Command

```
:FFT:SOURce {CHANnel<number> | FUNCtion<number> |
 WMEMory<number> | <float_value>}
```

Selects the source for the FFT.

<number>

For channels: 1, 2, 3, or 4.

For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Example

The following program sets the FFT source to channel 1.

```
10  OUTPUT 707;":FFT:SOURCE CHANNEL1"
20  END
```

Query

```
:FFT:SOURce?
```

The query returns the current source setting for the FFT.

Returned Format

```
[:FFT:SOURce] {CHANnel<number> | FUNCtion<number> |
WMEMory<number> | <float_value>}<NL>
```

WINDow

Command

```
:FFT:WINDow {RECTangular | HANNing | FLATtop}
```

Sets the window type for the FFT. The FFT function assumes that the time record repeats. Unless there is an integral number of cycles in the sampled waveform in the record, a discontinuity is created at the end of the record.

This introduces additional frequency components about the actual peaks. This is referred to as spectral leakage. In order to minimize spectral leakage, windows that approach zero smoothly at the beginning and end of the record are employed as filters to the FFTs.

Each window is useful for certain classes of input signals.

The RECTangular window is essentially no window, all points are multiplied by 1. The RECTangular window is useful for transient signals and signals where there are an integral number of cycles in the time record. The HANNing window is useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together or for making frequency measurements. The FLATtop window is best for making accurate amplitude measurements of frequency peaks.

Query

```
:FFT:WINDow?
```

The query returns the current window for the FFT.

Returned Format

```
[:FFT:WINDow] {RECTangular | HANNing | FLATtop}
```

Example

The following example places the current setting of the window in the string variable, Wind$, then prints the contents of the variable to the controller's screen.

```
10 DIM Wind$[50]
20 OUTPUT 707;":FFT:WINDOW?"
30 ENTER 707;Wind$
40 PRINT Wind$
50 END
```

11

Function Commands

Function Commands

The Function subsystem defines two functions, function 1 and function 2. The operands of these two functions can be any installed channels in the instrument, waveform memories 1 through 4, function, or a constant.

The Function subsystem contains the following commands and queries:

- ADD
- BWLimit
- DIFFerentiate
- DISPlay
- DIVide
- FFT
  - FREQuency
  - MAGNify
  - MSPan
  - WINDow
- FFTMagnitude
- HORizontal
  - POSition
  - RANGe
- INTegrate
- INVert
- MAGNify
- MAXimum
- MINimum
- MULTiply
- OFFSet
- ONLY
- RANGe
- SUBTract
- VERSus
- VERTical
  - OFFSet
  - RANGe

The vertical scaling and offset functions can be controlled remotely using the RANGE and OFFSET commands in this subsystem. The horizontal scaling and delay values of the functions can be obtained using the XRANGE and XPOSITION queries in this subsystem.

If a channel is not on and is specified as an operand, then that channel is acquired. If the operand waveforms have different record lengths, then the function takes the shorter of the two.

If the two operands have the same time scales, then the resulting function has the same time scale. If the operands have different time scales, then the resulting function has no valid time scale, since operations are done based on waveform record position and the time relationship of the data records is not considered. When the time scale is not valid, delta time pulse parameter measurements have no meaning and the unknown result indicator is displayed on screen.

Constant operands take on the same time scale as the associated waveform operand.

ADD

Command

```
:FUNCtion<number>:ADD <operand>,<operand>
```

Defines a function that takes the algebraic sum of two defined operands.

<number> An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

NOTE

Function 2 may use Function 1 as an operand, but Function 1 may *not* use Function 2 as an operand.

Example The following example sets up function 1 to add channel 1 to channel 2.

```
10  OUTPUT 707;":FUNCTION1:ADD CHANNEL1,CHANNEL2"
20  END
```

BWLimit

Command

```
:FUNCtion<number>:BWLimit <operand>,<operand>
```

Turns on the bandwidth limit filter function. This function may be performed on waveforms to limit the bandwidth of the signal. This function is useful for TDR/TDT applications where the accuracy of normalization is not required to simulate a bandlimited signal through a device under test.

<number>

An integer, 1 or 2, representing the selected function

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< >}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 1 to turn on the bandwidth limit filter function for channel 1.

```
10  OUTPUT 707;":FUNCTION1:BWLIMIT CHANNEL1"
20  END
```

DIFFerentiate

Command

```
:FUNCtion<number>:DIFFerentiate <operand>
```

Computes the discrete derivative of the defined operand's waveform.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to take the discrete derivative of the signal on channel 2.

```
10  OUTPUT 707;":FUNCTION2:DIFFERENTIATE CHANNEL2"
20  END
```

DISPlay

Command

```
:FUNCtion<number>:DISPlay {ON | 1 | OFF | 0}
```

Turns the selected function on and off.

<number>

An integer, 1 or 2, representing the selected function.

Example

The following example turns function 1 on.

```
10  OUTPUT 707;":FUNCTION1:DISPLAY ON"
20  END
```

Query

```
:FUNCtion<number>:DISPlay?
```

The query returns the state of the specified function.

Returned Format

```
[:FUNCtion<number>:DISPlay] {1 | 0}<NL>
```

Example

The following example places the current state of function 1 in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]              !Dimension variable
20  OUTPUT 707;":FUNCTION1:DISPLAY?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

DIVide

Command

```
:FUNCtion<number>:DIVide <operand>,<operand>
```

Defines a function that divides the first operand by the second operand.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to divide the signal on channel 1 by the signal in waveform memory 4.

```
10  OUTPUT 707;":FUNCTION2:DIVIDE CHANNEL1,WMEMORY4"
20  END
```

FFT:FREQuency

Command

```
:FUNCtion<number>:FFT:FREQuency <center_frequency_value>
```

Sets the center frequency for the FFT when magnify is on. Magnify is turned on and off with the FFT:MAGNify command.

<number> Function number 1 or 2.

<center_frequency_value> Value in Hertz.

Query

```
:FUNCtion<number>:FFT:FREQuency?
```

The query returns the center frequency value when magnify is on.

Returned Format

```
[:FUNCtion<number>:FFT:FREQuency] <center_frequency_value><NL>
```

FFT:MAGNify

Command

```
:FUNCtion<number>:FFT:MAGNify {{ON | 1} | {OFF | 0}}
```

Enables the magnification mode.

<number> Function number 1 or 2.

Query

```
:FUNCtion<number>:FFT:MAGNify?
```

The query returns the current state of the magnification mode.

Returned Format

```
[:FUNCtion<number>:FFT:MAGNify] {1 | 0}<NL>
```

FFT:MSPan

Command

```
:FUNCtion<number>:FFT:MSPan <magnify_span_value><NL>
```

Sets the span when magnify is on. Magnify is turned on and off with the FFT:MAGNify command.

<number> Function number 1 or 2.

<magnify_span_value> Span frequency in Hertz.

Query

```
:FUNCtion<number>:FFT:MSPan?
```

The query returns the current span value when magnify is on.

Returned Format

```
[:FUNCtion:FFT:MSPan] <magnify_span_value><NL>
```

FFT:WINDow

Command

```
:FUNCtion<number>:FFT:WINDow {RECTangular | HANNing | FLATtop}
```

Sets the window type for the FFT function.

The FFT function assumes that the time record repeats. Unless there is an integral number of cycles of the sampled waveform in the record, a discontinuity is created at the beginning of the record. This introduces additional frequency components into the spectrum about the actual peaks. This is referred to as spectral leakage. In order to minimize spectral leakage, windows that approach zero smoothly at the beginning and end of the record are employed as filters to the FFTs. Each window is useful for certain classes of input signals.

The RECTangular window is essentially no window; all points are multiplied by 1. The RECTangular window is useful for transient signals and signals where there are an integral number of cycles in the time record. The HANNing window is useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together or for making frequency measurements. The FLATtop window is best for making accurate amplitude measurements of frequency peaks.

<number>

Function number 1 or 2.

Example

```
10 OUTPUT 707;":FUNCtion<number>:FFT:WINDow RECTangular
20 END
```

Query

```
:FUNCtion<number>:FFT:WINDow?
```

The query returns the current selected window for the FFT function.

Returned Format

```
[:FUNCtion<number>:FFT:WINDow] {RECTangular | HANNing| FLATtop}<NL>
```

Example

```
10 DIM WND$[50]
20 OUTPUT 707;":FUNCTION1:FFT:WINDOW?
30 ENTER 707;WND$
40 PRINT WND$
50 END
```

FFTMagnitude

Command

```
:FUNCtion<number>:FFTMagnitude <operand>
```

Computes the fast Fourier transform of the specified channel or memory. The FFT takes the digitized time record of the specified channel or memory and transforms it to the frequency domain.

Query

```
:FUNCtion<number>:FFTMagnitude?
```

The query returns the frequency domain for the specified channel or memory.

HORizontal

Command

```
:FUNCtion<number>:HORizontal {TSOurce | MANual}
```

Sets the horizontal tracking to either Track SOurce or MANual.

The HORizontal command also includes a subsystem consisting of the following commands and queries:

- POSition
- RANGe

<number>

An integer, 1 or 2, representing the selected function.

Query

```
:FUNCtion<number>:HORizontal?
```

The query returns the current horizontal scaling mode of the specified function.

Returned Format

```
[:FUNCtion<number>:HORizontal] {TSOurce |MANual}<NL>
```

Example

The following example places the current state of horizontal tracking of function 1 in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]              !Dimension variable
20  OUTPUT 707;":FUNCTION1:HORIZONTAL?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

HORizontal:POSition

Command

```
:FUNCtion<number>:HORizontal:POSition<position_value>
```

Sets the time value at center screen for the selected function.

<number> Function number 1 or 2.

<position_value> Position value in time.

Query

```
:FUNCtion<number>:HORizontal:POSition?
```

The query returns the current time value at center screen of the selected function.

Returned Format

```
[:FUNCtion<number>:HORizontal:POSition]<position><NL>
```

Example The following example places the current horizontal position setting for function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:HORIZONTAL:POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

HORizontal:RANGe

Command

```
:FUNCtion<number>:HORizontal:RANGe<range_value>
```

Sets the current time range for the specified function.

<number> Function number 1 or 2.

<range_value> Width of screen in current x-axis units (usually seconds).

Query

```
:FUNCtion<number>:HORizontal:RANGe?
```

The query returns the current time range setting of the specified function.

Returned Format

```
[:FUNCtion<number>:HORizontal:RANGe]<range><NL>
```

Example

The following example places the current horizontal range setting of function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:HORIZONTAL:RANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

INTegrate

Command

```
:FUNCtion<number>:INTegrate <operand>
```

Defines a function that computes the integral of the defined operand's waveform.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 1 to compute the integral of waveform memory 3.

```
10  OUTPUT 707;":FUNCTION1:INTEGRATE WMEMORY3"
20  END
```

INVert

Command

```
:FUNCtion<number>:INVert <operand>
```

Defines a function that inverts the defined operand's waveform by multiplying by −1.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to invert the signal on channel 1.

```
10  OUTPUT 707;":FUNCTION2:INVERT CHANNEL1"
20  END
```

MAGNify

Command

```
:FUNCtion<number>: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, and memories with the RANGE and OFFSET commands in this subsystem.

Magnify performs the same operation as the "ONLY" operator and is the preferred operator of the two. "ONLY" is included in this instrument for compatibility with previous instruments.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example creates a function (function 1) that is a magnified copy of channel 1.

```
10  OUTPUT 707;":FUNCTION1:MAGNIFY CHANNEL1"
20  END
```

MAXimum

Command

```
:FUNCtion<number>:MAXimum <operand>
```

Defines a function that computes the maximum value of the operand waveform in each time bucket.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to compute the maximum of each time bucket for channel 2A.

```
10  OUTPUT 707;":FUNCTION2:MAXIMUM CHANNEL2A"
20  END
```

MINimum

Command

```
:FUNCtion<number>:MINimum <operand>
```

Defines a function that computes the minimum of each time bucket for the defined operand's waveform.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to compute the minimum of each time bucket for channel 4B.

```
10  OUTPUT 707;":FUNCTION2:MINIMUM CHANNEL4B"
20  END
```

MULTiply

Command

```
:FUNCtion<number>:MULTiply<operand>,<operand>
```

Defines a function that algebraically multiplies the first operand by the second operand.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example defines a function that multiplies channel 1 by waveform memory 1.

```
10  OUTPUT 707;":FUNCTION1:MULTIPLY CHANNEL1,WMEMORY1"
20  END
```

OFFSet

Command

```
:FUNCtion<number>:OFFSet <offset_value>
```

Sets the voltage represented at the center of the screen for the selected function. The offset is limited to being within the vertical range that can be represented by the function data.

<number> An integer, 1 or 2, representing the selected function.

<offset_value> The offset value is limited to being within the vertical range that can be represented by the function data.

Example The following example sets the offset voltage for function 1 to 2 mV.

```
10  OUTPUT 707;":FUNCTION1:OFFSET 2E-3"
20  END
```

Query

```
:FUNCtion<number>:OFFSet?
```

The query returns the current offset value for the selected function.

Returned Format

```
[:FUNCtion<number>:OFFSet]<offset_value><NL>
```

Example The following example places the current setting for offset on function 2 in the numeric variable, Value, then prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:OFFSET?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

NOTE

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

ONLY

Command

```
:FUNCtion<number>:ONLY <operand>
```

Defines a function that takes a copy of the original operand. It is similar to the MAGNIFY command. It is useful for scaling channels, another function, and memories with the RANGE and OFFSET commands in this subsystem.

NOTE

The ONLY command is provided for compatibility with previous instruments. MAGNIFY is the preferred method for new programs.

<number> An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example The following example creates a function (function 1) that is a copy of channel 1.

```
10  OUTPUT 707;":FUNCTION1:ONLY CHANNEL1"
20  END
```

RANGe

Command

```
:FUNCtion<number>:RANGe <full_scale_range>
```

Defines the full scale vertical axis of the selected function.

<number> An integer, 1 or 2, representing the selected function.

<full_scale_range> Full-scale vertical range.

Example The following example sets the full scale range for function 1 to 400 mV.

```
10  OUTPUT 707;":FUNCTION1:RANGE 400E-3"
20  END
```

Query

```
:FUNCtion<number>:RANGe?
```

The query returns the current full scale range setting for the specified function.

Returned Format

```
[:FUNCtion<number>:RANGe]<full_scale_range><NL>
```

Example The following example places the current range setting for function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:RANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

SUBTract

Command

```
:FUNCtion<number>:SUBTract<operand>,<operand>
```

Defines a function that algebraically subtracts the second operand from the first operand.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example defines a function that subtracts waveform memory 1 from channel 1.

```
10  OUTPUT 707;":FUNCTION1:SUBTRACT CHANNEL1,WMEMORY1"
20  END
```

VERSus

Command

```
:FUNCtion<number>:VERSus<operand>,<operand>
```

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 is initially equal to that of the first operand and can be adjusted with the RANGE and OFFSET commands in this subsystem.

<number>

An integer, 1 or 2, representing the selected function.

<operand>

```
{CHANnel< > | FUNCtion1 | WMEMory< > |<float_value>}
```

Where < > is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following 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"
20  END
```

VERTical

Command

```
:FUNCtion<number>:VERTical {TSOurce | MANual | AUTO}
```

Sets the vertical scaling mode of the specified function to either Track SOurce, MANual, or AUTO. Whether you use TSOurce or AUTO depends on the function definition. TSOurce should be used with the operators MAGNify and VERSus. AUTO should be used with all others. However, the instrument will accept either TSOurce or AUTO for functions that actually use "auto" on the front panel and will put the function into the autoscaling mode. The VERTical command also contains a subsystem consisting of the following commands and queries:

- OFFset
- RANge

<number>

An integer, 1 or 2, representing the selected function.

Query

```
:FUNCtion<number>:VERTical?
```

The query returns the current vertical scaling mode of the specified function.

Returned Format

```
[:FUNCtion<number>:VERTical] {TSOurce | MANual |AUTO}<NL>
```

Example

The following example places the current state of the vertical tracking of function 1 in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]              !Dimension variable
20  OUTPUT 707;":FUNCTION1:VERTICAL?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

VERTical:OFFSet

Command

```
:FUNCtion<number>:VERTical:OFFSet<offset_value>
```

Sets the voltage represented at center screen for the selected function.

<number> Function number 1 or 2.

<offset_value> The offset value is limited only to being within the vertical range that can be represented by the function data.

Query

```
:FUNCtion<number>:VERTical:OFFset?
```

The query returns the current offset value of the selected function.

Returned Format

```
[:FUNCtion<number>:VERTical:OFFset]<offset_value><NL>
```

Example The following example places the current offset setting for function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:VERTICAL:OFFSET?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VERTical:RANGe

Command

```
:FUNCtion<number>:VERTical:RANGe<full_scale_range>
```

Defines the full-scale vertical axis of the selected function.

<number> Function number 1 or 2.

<full_scale_range> The range may be expanded until there are no fewer than 32 q-levels on the screen.

Query

```
:FUNCtion<number>:VERTical:RANGe?
```

The query returns the current range setting of the specified function.

Returned Format

```
[:FUNCtion<number>:VERTical:RANGe]<range><NL>
```

Example The following example places the current vertical range setting of function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:VERTICAL:RANGe?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

12

Hardcopy Commands

Hardcopy Commands

The Hardcopy subsystem commands set various parameters for printing the screen. The print sequence is activated when the root level command PRINT is sent.

The Hardcopy subsystem contains the following commands:

- ADDRess
- AREA
- BACKground
- DESTination
- DEVice
- FACTors
- FFEed (Form FEed)
- FILename
- LENGth
- MEDia

ADDRess

Command

```
:HARDcopy:ADDRess <address>
```

Sets the address for the printer when printing to an HP-IB printer.

<address>

The address must be a legal HP-IB address, 0 to 30.

Example

The following example sets the hardcopy destination to HP-IB address 8.

```
10  OUTPUT 707;":HARDCOPY:DESTINATION HPIB"<R>
20  OUTPUT 707;":HARDCOPY:ADDRESS 8"
30  END
```

Query

```
:HARDcopy:ADDRess?
```

The query returns the current HP-IB printer address.

Returned Format

```
[:HARDcopy:ADDRess] <address>
```

Example

The following example sets the HP-IB address of the hardcopy destination and prints it on the controller screen:

```
10  OUTPUT 707;":HARDCOPY:ADDRESS?"<R>
20  ENTER 707;Hard_address
30  PRINT Hard_address
40  END
```

AREA

Command

```
:HARDcopy:AREA {GRATicule | SCReen | FACTors}
```

Selects which data from the screen is to be printed. When GRATicule is selected, the data in the graticule area of the screen is printed. When SCReen is selected, the entire screen is printed. When FACTors is selected, only the text description of the current setup is printed.

The setup factors can be turned on and off using the HARDcopy:FACTors command when either GRATicule or SCReen is selected as the area for printing.

Example

The following example selects the graticule data for printing.

```
10  OUTPUT 707;":HARDCOPY:AREA GRATICULE"
20  END
```

Query

```
:HARDcopy:AREA?
```

The query returns the current setting for the area of the screen to be printed.

Returned Format

```
[:HARDcopy:AREA] {GRATicule | SCReen | FACTors}<NL>
```

Example

The following example places the current selection for the area to be printed in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]        !Dimension variable
20  OUTPUT 707;":HARDCOPY:AREA?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

BACKground

Command

```
:HARDcopy:BACKground {WHITe | NORMal}
```

Controls the background color of the graticule area of an HP PaintJet printer. It is only valid when the destination printer is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper).

Example

The following example selects white as the backgound color.

```
10  OUTPUT 707;":HARDCOPY:BACKGROUND WHITE"
20  END
```

Query

```
:HARDcopy:BACKground?
```

The query returns the selected setting of the background color.

Returned Format

```
[:HARDcopy:BACKground] {WHITe | NORMal}<NL>
```

Example

The following example places the current selection for print destination in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":HARDCOPY BACKGROUND?
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

DESTination

Command

```
:HARDcopy:DESTination {CENTronics | HPIB | DISK}
```

Selects the destination for printing. The options are HPIB, CENTronics, and the internal DISK.

> **NOTE**
>
> When printing TIFF, GIF, or PCX files, the only choice of destination is DISK.

Example

The following example selects HPIB as the print destination.

```
10  OUTPUT 707;":HARDCOPY:DESTINATION HPIB"
20  END
```

Query

```
:HARDcopy:DESTination?
```

The query returns the current destination for printing.

Returned Format

```
[:HARDcopy:DESTination] {CENTronics | HPIB | DISK}<NL>
```

Example

The following example places the current selection for print destination in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":HARDCOPY:DESTINATION?
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

DEVice

Command

```
:HARDcopy:DEVice {THINkjet | PAINtjet | LASerjet | EPSon |
 TIFF | CTIFf | GIF | PCX | DJ500 | DJ540 | DJ550  | DJ560 |
 BWPaintjet | BWDeskjet}
```

Selects the output format. When TIFF, CTIFf, GIF, or PCX are selected, the only destination available is DISK. To get data in the TIFF, CTIFf, GIF, or PCX format, use the root level PRINt command. The HARDcopy:DESTination DISK command can be used to store the data on disk in any of the above formats.

Example

The following example sets up the output format for an HP LaserJet printer.

```
10  OUTPUT 707;":HARDCOPY:DEVICE LASERJET"
20  END
```

Query

```
:HARDcopy:DEVice?
```

The query returns the current output format selection.

Returned Format

```
[:HARDcopy:DEVice] {THINkjet | PAINtjet | LASerjet | EPSon |
TIFF | CTIFf | GIF | PCX | DJ500 | DJ540 | DJ550 | DJ560 |
BWPaintjet | BWDeskjet}<NL>
```

Example

The following example places the current selection for the hardcopy output format in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]        !Dimension variable
20  OUTPUT 707;":HARDCOPY:DEVICE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

FACTors

Command

```
:HARDcopy:FACTors {ON | 1} | {OFF | 0}
```

Determines whether the instrument setup factors will be appended to screen or graticule images.

Example

The following example turns on the setup factors.

```
10  OUTPUT 707;":HARDCOPY:FACTORS ON"
20  END
```

Query

```
:HARDcopy:FACTors?
```

The query returns the current setup factors setting.

Returned Format

```
[:HARDcopy:FACTors] {1 | 0}<NL>
```

Example

The following example places the current setting for the setup factors in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]              !Dimension variable
20  OUTPUT 707;":HARDCOPY:FACTORS?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

FFEed (Form FEed)

Command

```
:HARDcopy:FFEed {ON | 1} | {OFF | 0}
```

Sets the form feed option. If it is set to ON, a formfeed will occur at the end of the hardcopy; otherwise, the page will scroll up by 4 lines.

Example

The following example turns the form feed option on.

```
10  OUTPUT 707;":HARDCOPY:FFEED ON"
20  END
```

Query

```
:HARDcopy:FFEed?
```

The query returns the current setup of the form feed option setting.

Returned Format

```
[:HARDcopy:FFEed] {1 | 0}<NL>
```

Example

The following example places the current setting for the setup factors in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":HARDCOPY:FFEED?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

FILename

Command

```
:HARDcopy:FILename <string>
```

Specifies the filename when printing files to the disk.

<string>

Any series of ASCII characters (A to Z, 0 to 9, and, _ (underscore)) enclosed in quotes and up to 8 characters. The appropriate suffix will be automatically appended.

Example

The following example sets the filename for printing to "CAPTURE."

```
10  OUTPUT 707;":HARDCOPY:FILENAME ""CAPTURE""<R>
20  END
```

LENGth

Command

```
:HARDcopy:LENGth {ENGLish | 11} | {METRic | 12}
```

Sets the length of the paper to either ENGLish (11 inches) or METRic (12 inches). The longer paper is metric size A4.

Example

The following example sets the hardcopy length to 11 inches.

```
10  OUTPUT 707;":HARDCOPY:LENGTH ENGLISH"
20  END
```

Query

```
:HARDcopy:LENGth?
```

The query returns the current paper length setting.

Returned Format

```
[:HARDcopy:LENGth] {ENGLish | METRic}<NL>
```

Example

The following example places the current setting for the paper length in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":HARDCOPY:LENGTH?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

MEDia

Command

```
:HARDcopy:MEDia {PAPer | TRANsparency}
```

Sets the speed of the printer. When TRANSPARENCY is selected, the printer prints the data twice, which makes the contents of the paper look darker and slows down the printing process. This command applies only to HP PaintJet and color DeskJet printers.

Example

The following example sets up the printer for printing on paper.

```
10  OUTPUT 707;":HARDCOPY:MEDIA PAPER"
20  END
```

Query

```
:HARDcopy:MEDia?
```

The query returns the current media setting.

Returned Format

```
[:HARDcopy:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following example places the current setting for printer speed in the string variable, Speed$, then prints the contents of the variable to the controller's screen.

```
10  DIM Speed$[50]            !Dimension variable
20  OUTPUT 707;":HARDCOPY:MEDIA?"
30  ENTER 707;Speed$
40  PRINT Speed$
50  END
```

13

Histogram Commands

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.

The Histogram subsystem contains the following commands:

- AXIS
- MODE
- RRATe
- RUNTil
- SCALe
  - OFFSet
  - RANGe
  - SCALe
  - TYPE
- WINDow
  - DEFault
  - SOURce
  - X1Position
  - X2Position
  - Y1Position
  - Y2Position

Histograms and the database

The histograms, mask testing, and color-graded 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, which is 256 pixels high by 451 pixels wide. 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 DISPlay:CGRade:LEVels command 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 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. Even if the display is set to show only the most recent acquisition, the database keeps track of all pixel hits while the database is building.

Remember that color-graded display, mask testing, and histograms all use the same database. 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 DUT conditions and acquire new data before extracting measurement results.

AXIS

Command

```
:HISTogram:AXIS {VERTical | HORizontal}
```

Selects the axis of the histogram. A horizontal or vertical histogram may be created.

Example

The following example defines a vertical histogram.

```
10  OUTPUT 707;":HISTOGRAM:AXIS VERTICAL"
20  END
```

Query

```
:HISTogram:AXIS?
```

The query returns the currently selected histogram axis.

Returned Format

```
[:HISTogram:AXIS] {VERTical | HORizontal} <NL>
```

Example

The following example returns the result of the axis query and prints it to the controller's screen.

```
10  DIM Axis$[50]
20  OUTPUT 707;":HISTOGRAM:AXIS?"
30  ENTER 707;Axis$
40  PRINT Axis$
50  END
```

MODE

Command

```
:HISTogram:MODE {OFF | WAVeform}
```

Selects the histogram mode. The histogram may be off or set to track the waveform database.

Example

The following example sets the histogram mode to track the waveform database.

```
10  OUTPUT 707;":HISTOGRAM:MODE WAVEFORM"
20  END
```

Query

```
:HISTogram:MODE?
```

The query returns the currently selected histogram mode.

Returned Format

```
[:HISTogram:MODE] {OFF | WAVeform} <NL>
```

Example

The following example returns the result of the mode query and prints it to the controller's screen.

```
10  DIM Mode$[10]
20  OUTPUT 707;":HISTOGRAM:MODE?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

RRATe

Command

```
:HISTogram:RRATe <time>
```

Sets the refresh rate of the histogram measurements.

Example

The following example sets the refresh rate for 3 seconds.

```
10 OUTPUT 707;":HISTogram:RRATe 3"
20 END
```

Query

```
:HISTogram:RRATe?
```

The query returns the current refresh rate setting in seconds.

Example

The following example obtains the current refresh rate setting and prints it to the controller's screen:

```
10 OUTPUT 707;":HISTogram:RRATe?"
20 ENTER 707; Rate
30 PRINT Rate
40 END
```

RUNTil

Command

```
:HISTogram:RUNTil {FOREver | WAVeforms, <n waveforms> |SAMples,
<n samples>}
```

Selects the histogram run until mode. The histogram may be set to run until n waveforms or n samples have been acquired or will run forever. 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.

Example

The following example specifies that the histogram runs until 200 samples have been obtained.

```
10  OUTPUT 707;":HISTOGRAM:RUNTIL SAMPLES,200"
20  END
```

Query

```
:HISTogram:RUNTil?
```

The query returns the currently selected run until state.

Returned Format

```
[:HISTogram:RUNTil] {FOREver | WAVeform, <n waveforms>
| SAMPles, <n samples>}<NL>
```

Example

The following example returns the result of the run until query and prints it to the controller's screen.

```
10  DIM Runt$[50]
20  OUTPUT 707;":HISTOGRAM:RUNTIL?"
30  ENTER 707;Runt$
40  PRINT Runt$
50  END
```

SCALe

Command

```
:HISTogram:SCALe {AUTO | MANual}
```

Selects the histogram scaling mode. The scaling mode may be automatic or manual. In automatic scaling, the scale will be set to display the histogram using one half of the display. In manual scaling, the scale and offset may be set manually to show any portion of the histogram.

Example

The following example selects automatic scaling.

```
10  OUTPUT 707;":HISTOGRAM:SCALE AUTO"
20  END
```

Query

```
:HISTogram:SCALe?
```

The query returns the currently selected histogram scale mode.

Returned Format

```
[:HISTogram:SCALe]{AUTO | MANual} <NL>
```

Example

The following example returns the result of the scale query and prints it to the controller's screen.

```
10  DIM Scal$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE?"
30  ENTER 707;Scal$
40  PRINT Scal$
50  END
```

SCALe:OFFSet

Command

```
:HISTogram:SCALe:OFFSet <offset>
```

Sets the histogram offset. For horizontal histograms, this is the vertical offset in percentage of peak or decibels. For vertical histograms, this is the horizontal offset in percentage of peak or decibels.

For the linear scale type, the offset is the percentage of peak at the left or lower edge of the display. For example, with a horizontal histogram, an offset of 50% would place one half of the peak (50%) at the lower edge of the display. 50% of the histogram would be below the display and the other 50% would be above the lower edge of the display.

For the log scale type, the offset is in decibels at the left or lower edge of the display. The histogram is plotted according to the formula:

$$y = 20log_{10}\left(\frac{x}{peak}\right)$$

where x is the number of hits of a histogram column and *peak* is the number of hits of the largest histogram column. This means that 0 dB is at the histogram peak and that the offset can only contain negative values. For example, with a horizontal histogram, an offset of −20 dB would place 10% or $10^{(-20/20)}$ of the peak at the lower edge of the display.

The histogram scale should be in manual mode when the offset is modified.

Example

The following example sets the offset to 50%.

```
10  OUTPUT 707;":HISTOGRAM:SCALE:OFFSET 50"
20  END
```

Query

```
:HISTogram:SCALe:OFFSet?
```

The query returns the current offset in decibels or percentage of hits.

Returned Format

```
[:HISTogram:SCALe:OFFSet] <offset> <NL>
```

Example

The following example returns the result of the offset query and prints it to the controller's screen.

```
10  DIM Off$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:OFFSET?"
30  ENTER 707;Off$
40  PRINT Off$
50  END
```

SCALe:RANGe

Command

```
:HISTogram:SCALe:RANGe <range>
```

Selects the histogram scale across the entire display. For horizontal histograms, this is the vertical percentage of peak or decibels across the display. For vertical histograms, this is the horizontal percentage of peak or decibels across the display.

For the linear scale type, the range is in the percentage of the peak across the display. For example, with a horizontal histogram, 200% would place 25% of the histogram in each of 8 divisions with the top of the peak (100%) at the middle of the display.

For the log scale type, the range is in decibels full scale. The histogram is plotted according to the formula:

$$y = 20log_{10}\left(\frac{x}{peak}\right)$$

where x is the number of hits of a histogram column and $peak$ is the number of hits of the largest histogram column.

The histogram scale should be in manual mode when the range is modified.

Example

The following example sets the range of the histogram to 200%.

```
10  OUTPUT 707;":HISTOGRAM:SCALE:RANGE 200"
20  END
```

Query

```
:HISTogram:SCALe:RANGe?
```

The query returns the current range in decibels or percentage of hits across the display.

Returned Format

```
[:HISTogram:SCALe:RANGe] <range><NL>
```

Example

The following example returns the result of the range query and prints it to the controller's screen.

```
10  DIM Range$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:RANGE?"
30  ENTER 707;Range$
40  PRINT Range$
50  END
```

SCALe:SCALe

Command

```
:HISTogram:SCALe:SCALe <scale>
```

Selects the histogram scale per division. For horizontal histograms, this is the vertical percentage of peak or decibels per division. For vertical histograms, this is the horizontal percentage of peak or decibels per division.

For the linear scale type, the scale is in the percentage of peak per division. For example, with a horizontal histogram, 25% would place one quarter of the histogram in each of 8 divisions with the top of the peak (100%) at the middle of the display.

For the log scale type, the scale is in decibels per division. The histogram is plotted according to the formula:

$$y = 20log_{10}\left(\frac{x}{peak}\right)$$

where x is the number of hits of a histogram column and *peak* is the number of hits of the largest histogram column.

The histogram scale should be in manual mode when the scale is modified.

Example

The following example sets the histogram scale to 25% (per division).

```
10  OUTPUT 707;":HISTOGRAM:SCALE:SCALE 25"
20  END
```

Query

```
:HISTogram:SCALe:SCALe?
```

The query returns the current range in decibels or percentage of hits across the display.

Returned Format

```
[:HISTogram:SCALe:SCALe] <scale><NL>
```

Example

The following example returns the result of the scale query and prints it to the controller's screen.

```
10  DIM Scscale$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:SCALE?"
30  ENTER 707;Scscale$
40  PRINT Scscale$
50  END
```

SCALe:TYPE

Command

```
:HISTogram:SCALe:TYPE {LINear | LOGarithmic}
```

Selects the histogram scale type.

The histogram may be displayed according to a linear or a logarithmic scale.

Example

The following example sets the histogram scale to linear.

```
10  OUTPUT 707;":HISTOGRAM:SCALE:TYPE:LINEAR"
20  END
```

Query

```
:HISTogram:SCALe:TYPE?
```

The query returns the currently selected histogram scale type.

Returned Format

```
[:HISTogram:SCALe:TYPE] {LINear | LOGarithmic} <NL>
```

Example

The following example returns the result of the scale type query and prints it to the controller's screen.

```
10  DIM Type$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:TYPE?"
30  ENTER 707;Type$
40  PRINT Type$
50  END
```

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

The following example sets the histogram window to the default position.

```
10 OUTPUT 707;":HISTogram:WINDow:DEFault"
20 END
```

WINDow:SOURce

Command

```
:HISTogram:WINDow:SOURce {CHANnelN | FUNCtionN | WMEMoryN | FFT}
```

Selects the source of the histogram window. The histogram window will track the source's vertical and horizontal scale.

Example

The following example sets the histogram window's source to Channel 1.

```
10  OUTPUT 707;":HISTOGRAM:WINDOW:SOURCE CHANNEL1"
20  END
```

Query

```
:HISTogram:WINDow:SOURce?
```

The query returns the currently selected histogram window source.

Returned Format

```
[:HISTogram:WINDow:SOURce] {CHANnelN | FUNCtionN | WMEMoryN |
FFT}<NL>
```

Example

The following example returns the result of the window source query and prints it to the controller's screen.

```
10  DIM Winsour$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:SOURCE?"
30  ENTER 707;Winsour$
40  PRINT Winsour$
50  END
```

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"
20  END
```

Query

```
:HISTogram:WINDow:X1Position?
```

The query returns the value of the X1 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:X1Position ]<X1 position><NL>
```

Example

The following example returns the result of the X1 position query and prints it to the controller's screen.

```
10  DIM X1$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:X1POSITION?"
30  ENTER 707;X1$
40  PRINT X1$
50  END
```

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"
20  END
```

Query

```
:HISTogram:WINDow:X2Position?
```

The query returns the value of the X2 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:X2Position] <X2 position><NL>
```

Example

The following example returns the result of the X2 position query and prints it to the controller's screen.

```
10  DIM X2$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:X2POSITION?"
30  ENTER 707;X2$
40  PRINT X2$
50  END
```

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"
20  END
```

Query

```
:HISTogram:WINDow:Y1Position?
```

The query returns the value of the Y1 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:Y1Position] <Y1 position><NL>
```

Example

The following example returns the result of the Y1 position query and prints it to the controller's screen.

```
10  DIM Y1$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:Y1POSITION?"
30  ENTER 707;Y1$
40  PRINT Y1$
50  END
```

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"
20  END
```

Query

```
:HISTogram:WINDow:Y2Position?
```

The query returns the value of the Y2 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:Y2Position] <Y2 position><NL>
```

Example

The following example returns the result of the Y2 position query and prints it to the controller's screen.

```
10  DIM Y2$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:Y2POSITION?"
30  ENTER 707;Y2$
40  PRINT Y2$
50  END
```

14

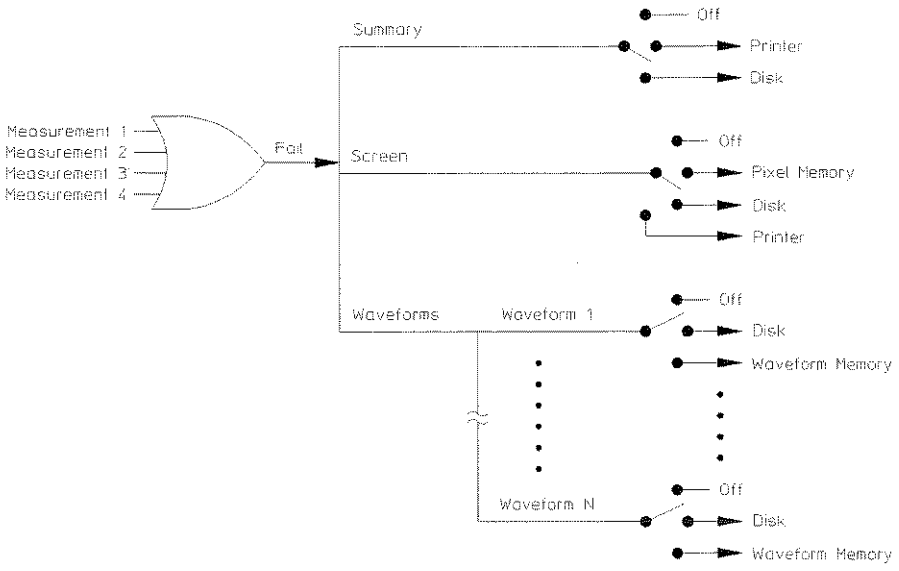
Limit Test Commands

Limit Test Commands

The Limit Test commands and queries control the limit test features of the HP 54750A digitizing oscilloscope and the HP 83480A digital communications analyzer. 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.

The Limit Test subsystem contains the following commands:

- FAIL
- LLIMit (Lower Limit)
- MNFound (Measurement Not Found)
- RUN or RUMode
- SOURce
- SSCReen (Store Screen)
  - DDISk
    - BACKground
    - MEDia
    - PFORmat
  - DPRinter
    - ADDRess
    - BACKground
    - MEDia
    - PFORmat
    - PORT
- SSUMmary (Store Summary)
  - ADDRess
  - FORMat
  - MEDia
  - PFORmat
  - PORT
- SWAVeform (Store Waveform)
- TEST
- ULIMit (Upper Limit)



54720m12

Limit Test Functional Diagram

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.

INSide

FAIL: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

FAIL:OUTside causes the instrument to fail a test when the measurement results exceed the parameters set by LTESt:LLIMit and LTESt:ULIMit commands.

ALWays

FAIL: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

FAIL: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.

Example

The following example causes the instrument to fail a test when the measurements are outside the lower and upper limits.

```
10  OUTPUT 707;":LTEST:FAIL OUTSIDE"
20  END
```

Query

```
:LTESt:FAIL?
```

The query returns the current set fail condition.

Returned Format

```
[:LTESt:FAIL] {INSide | OUTSide | ALWays | NEVer}<NL>
```

Example

The following example returns the current fail condition and prints the result to the controller's screen.

```
10  DIM FAIL$[50]
20  OUTPUT 707;":LTEST:FAIL?"
30  ENTER 707;FAIL$
40  PRINT FAIL$
50  END
```

LLIMit

Command

```
:LTESt:LLIMit <lower_value>
```

Sets the lower test limit for the active measurement currently selected by the :LTESt:SOURce command.

<lower_value>

A real number.

Example

The following example sets the lower test limit to 1.

```
10  OUTPUT 707;":LTEST:LLIMIT 1"
20  END
```

If, for example, you chose to measure volts peak-peak and want the smallest acceptable signal swing to be one volt, you could use the above command, 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

The following example returns the current lower test limit and prints the result to the controller's screen.

```
10  DIM LLIM$[50]
20  OUTPUT 707;":LTEST:LLIMIT?"
30  ENTER 707;LLIM$
40  PRINT LLIM$
50  END
```

MNFound

Command

```
:LTESt:MNFound {FAIL | PASS | IGNore}
```

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

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

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

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.

Example

The following example causes the instrument to pass the test when a measurement cannot be made.

```
10  OUTPUT 707;":LTEST:MNFOUND PASS"
20  END
```

Query

```
:LTESt:MNFound?
```

The query returns the current action set by the command.

Returned Format

```
[:LTESt:MNFound] {FAIL | PASS | IGNore}<NL>
```

Example

The following example gets the current setting of the measurement not found action and prints the result to the controller's screen.

```
10  DIM MNF$[50]
20  OUTPUT 707;":LTEST:MNFOUND?"
30  ENTER 707;MNF$
40  PRINT MNF$
50  END
```

RUN (RUMode)

Command

```
:LTESt:RUN {FORever | FAILures, <total_failures> | WAVeforms,
<maximum_waveforms>}
```

Determines the termination conditions for the test. The choices are FORever, FAILures, or WAVeforms.

If FAILures or WAVeforms are selected, a second parameter is required indicating the number of failures that can occur or the number of waveforms that are to be acquired.

The keyword RUMode (Run Until Mode) may be used instead of RUN.

FORever

FORever runs the limit test until the test is turned off. This mode is used when you want a measurement to run continually and not to stop after a fixed number of failures. For example, you may want the limit test to run overnight and not be limited by a number of failures.

FAILures

FAILures runs the limit test until a set number of failures occurs. 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.

WAVeforms

WAVeforms sets the maximum waveforms that are required. When any measurement reaches this number of waveforms, the test terminates. WAVeforms runs the limit test until a set number of waveforms are acquired.

Use the WAVeforms mode when you want the limit test to reach completion after a set number of waveforms are acquired. The test terminates with the measurement that first reaches the specified number of waveforms.

<maximum_waveforms>

An integer: 1 to 1,000,000,000.

Example

The following example causes limit test to run until 25 waveforms are acquired.

```
10  OUTPUT 707;":LTEST:RUN WAVEFORMS, 25"
20  END
```

Query

```
:LTESt:RUN?
```

The query returns the currently selected termination condition and value.

Returned Format

```
[:LTESt:RUN] {FOREver | FAILures, <total_failures> | WAVeforms,
<maximum_waveforms>}<NL>
```

Example

The following example returns the current condition under which the limit test terminates and prints the result to the controller's screen.

```
10  DIM RUN$[50]
20  OUTPUT 707;":LTEST:RUN?"
30  ENTER 707;RUN$
40  PRINT RUN$
50  END
```

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.

Example

The following example selects the first measurement as the source for the limit testing commands.

```
10  OUTPUT 707;":LTEST:SOURCE 1"
20  END
```

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$[50]
20  OUTPUT 707;":LTEST:SOURCE?"
30  ENTER 707;SOURCE$
40  PRINT SOURCE$
50  END
```

See Also

Measurements are started in the Measurement Subsystem.

SSCReen

Command

```
:LTESt:SSCReen  {OFF | PMEMory1 | PRINter | DISK [,<filename>]}
```

Saves a copy of the screen in the event of a failure.

A destination of OFF turns off the save action.

A destination of PMEMory1 adds the screen to the pixel memory.

When the destination is PRINter, a variety of printer-related controls are used to specify the printer.

When the destination is DISK, a different set of commands is provided to control the print to disk. The two printer specifications are set through the DPRinter (Destination Printer) and the DDISk (Destination Disk) commands.

<filename>

A file prefix of four characters. If no prefix is specified, the default prefix is SCRN.

The LTESt:SSCReen contains the following command sub-subsystems when either PRINter or DISK is specified:

- DDISk (Display Disk)
- DPRinter (Display Printer)

Example

The following example saves a copy of the screen to the printer in the event of a failure. Additional printer-related controls are set using the SSCReen:DPRinter command set.

```
10  OUTPUT 707;":LTEST:SSCREEN PRINTER"
20  END
```

Query

```
:LTESt:SSCReen?
```

The query returns the current state of the SSCreen command.

Returned Format

```
[:LTESt:SSCReen] {OFF | PMEMory1 | PRINter | DISK [,<filename>]}<NL>
```

Example

The following example returns the destination of the save screen when a failure occurs and prints the result to the controller's screen.

```
10  DIM SSCR$[50]
20  OUTPUT 707;":LTEST:SSCREEN?"
30  ENTER 707;SSCR$
40  PRINT SSCR$
50  END
```

SSCReen:DDISk

The LTESt:SSCReen:DDISk sub-subsystem commands are used to set up the disk drive when storing the limit test display screen to a disk. Use DDISk to store limit test display screens to a file in printer format for later printing. The disk drive setup consists of the following commands:

- BACKground
- MEDia
- PFORmat (printer format)

SSCReen:DDISk:BACKground

Command

```
:LTESt:SSCReen:DDISk:BACKground {WHITe | NORMal}
```

Controls the background color of the graticule area of an HP PaintJet printer. It is only valid when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the disk.

Example

The following example sets the background color of the HP PaintJet printer to white.

```
10  OUTPUT 707;":LTEST:SSCREEN:DDISK:BACKGROUND WHITE"
20  END
```

Query

```
:LTESt:SSCReen:DDISk:BACKground?
```

The query returns the current background for the disk store.

Returned Format

```
[:LTESt:SSCReen:DDISk:BACKground] {WHITe | NORMal}<NL>
```

Example

The following example returns the current background color of the HP PaintJet printer and prints the result to the controller's screen.

```
10  DIM DDBCK$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DDISK:BACKGROUND?"
30  ENTER 707;DDBCK$
40  PRINT DDBCK$
50  END
```

SSCReen:DDISk:MEDia

Command

```
:LTESt:SSCReen:DDISk:MEDia {PAPer | TRANsparency}
```

Specifies either paper or transparency to be used in the printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the media look darker and slows down the printing process. This control is used when the store screen is directed to the disk, and is valid only when printing to the HP PaintJet and color DeskJet printers.

Example

The following example specifies that paper will be used in the printer when a file saved to disk is later printed.

```
10  OUTPUT 707;":LTEST:SSCREEN:DDISK:MEDIA PAPER"
20  END
```

Query

```
:LTESt:SSCReen:DDISk:MEDia?
```

The query returns the current media for the disk store.

Returned Format

```
[:LTESt:SSCReen:DDISk:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following example returns the current media to be used in the printer and prints the result to the controller's screen.

```
10  DIM DDMED$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DDISK:MEDIA?"
30  ENTER 707;DDMED$
40  PRINT DDMED$
50  END
```

SSCReen:DDISk:PFORmat

Command

```
:LTESt:SSCReen:DDISk:PFORmat
{THINkjet | PAINtjet | LASerjet | EPSon | GIF | TIFF| CTIFf | PCX |
DJ500 | DJ540 | DJ550 | DJ560 | BWPaintjet | BWDeskjet}
```

Selects the file format for a stored file or the printer format to use when storing the screen to a disk for later printing. This includes all print formats supported by the instrument.

Example

The following example sets the format of the file stored to disk for later printing to an HP PaintJet.

```
10  OUTPUT 707;":LTEST:SSCREEN:DDISK:PFORMAT PAINTJET"
20  END
```

Query

```
:LTESt:SSCReen:DDISk:PFORmat?
```

The query returns the current printer format for the disk store.

Returned Format

```
[:LTESt:SSCReen:DDISk:PFORmat]
{ THINkjet | PAINtjet | LASerjet | EPSon | GIF | TIFF | CTIFf | PCX |
DJ500 | DJ540 | DJ550 | DJ560 | BWPaintjet | BWDeskjet}<NL>
```

Example

The following example returns the file format and prints the result to the controller's screen.

```
10  DIM DDPFORM$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DDISK:PFORMAT?"
30  ENTER 707;DDPFORM$
40  PRINT DDPFORM$
50  END
```

SSCReen:DPRinter

The LTESt:SSCReen:DPRinter sub-subsystem commands are used to set up the printer when storing the limit test display screen to a printer. The printer setup consists of the following commands:

- ADDRess
- BACKground
- MEDia
- PFORmat
- PORT

SSCReen:DPRinter:ADDRess

Command

```
:LTESt:SSCReen:DPRinter:ADDRess <address_value>
```

Allows the user to select the HP-IB address for the printer. This address is used only if the port is HP-IB.

<address_value> Any HP-IB address, 0 to 30.

Example The following example sets the HP-IB address for the printer to 1.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:ADDRESS 1"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:ADDRess?
```

The query returns the current HP-IB address of the printer.

Returned Format

```
[:LTESt:SSCReen:DPRint:ADDRess]<address_value><NL>
```

Example The following example returns the current HP-IB address for the printer and prints the result to the controller's screen.

```
10  DIM DPADD$[10]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:ADDRESS?"
30  ENTER 707;DPADD$
40  PRINT DPADD$
50  END
```

SSCReen:DPRinter:BACKground

Command

```
:LTESt:SSCReen:DPRinter:BACKground {WHITe | NORMal}
```

Controls the background color of the graticule area of an HP PaintJet printer. It is only valid when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the printer.

Example

The following example sets the background to white.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:BACKGROUND WHITE"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:BACKground?
```

The query returns the current background for the printer.

Returned Format

```
[:LTESt:SSCReen:DPRinter:BACKground] {WHITe | NORMal} <NL>
```

Example

The following example returns the current background setting and prints the result on the controller screen:

```
10  DIM FAIL$[10]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:BACKGROUND?"
30  ENTER 707;FAIL$
40  PRINT FAIL$
50  END
```

SSCReen:DPRinter:MEDia

Command

```
:LTESt:SSCReen:DPRinter:MEDia {PAPer | TRANsparency}
```

Specifies either paper or transparency in the printer. This control is used when the store screen is directed to a printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the media look darker and slows down the printing process. This command is valid only for the HP PaintJet and color DeskJet printers.

Example

The following example selects paper as the printer media.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:MEDIA PAPER"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:MEDia?
```

The query returns the current printer media.

Returned Format

```
[:LTESt:SSCReen:DPRinter:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following example gets the current media setting and prints the result to the controller's screen.

```
10  DIM DPMED$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:MEDIA?"
30  ENTER 707;DPMED$
40  PRINT DPMED$
50  END
```

SSCReen:DPRinter:PFORmat

Command

```
:LTESt:SSCReen:DPRinter:PFORmat
{THINkjet | PAINtjet | LASerjet | EPSon | DJ500 | DJ540 |
DJ550 | DJ560 | BWPaintjet | BWDeskjet}
```

Selects the printer format to use when storing the screen to a printer. This includes only the formats available from the setup print menu that refer to a physical printer.

Example

The following example sets the printer format to an HP PaintJet.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:PFORMAT PAINTJET"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:PFORmat?
```

The query returns the current printer format.

Returned Format

```
[:LTESt:SSCReen:DPRinter:PFORmat]
{THINkjet | PAINtjet | LASerjet | EPSon | DJ500 | DJ540
| DJ550 | DJ560 | BWPaintjet | BWDeskjet}<NL>
```

Example

The following example returns the current printer format and prints the result to the controller's screen.

```
10  DIM DPPFORM$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:PFORMAT?"
30  ENTER 707;DPPFORM$
40  PRINT DPPFORM$
50  END
```

SSCReen:DPRinter:PORT

Command

```
:LTESt:SSCReen:DPRinter:PORT {CENTronics | HPIB}
```

Selects the printer port for the screen when store screen is going to the printer.

Example

The following example selects the Centronics printer port.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:PORT CENT"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:PORT?
```

The query returns the current port type.

Returned Format

```
[:LTESt:SSCReen:DPRinter:PORT] {CENTronics | HPIB} <NL>
```

Example

The following example returns the current printer port and prints the result to the controller's screen.

```
10  DIM DPPORT$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:PORT?"
30  ENTER 707;DPPORT$
40  PRINT DPPORT$
50  END
```

SSUMmary

Command

```
:LTESt:SSUMmary {OFF | PRINter | DISK [,<filename>]}
```

Saves the summary in the event of a failure.

A destination of OFF turns off the summary save.

A destination of PRINter builds the summary file and sends it to the printer. In this case, a variety of printer-related controls are used to specify the printer configuration. The following printer commands are used when storing the summary to a printer:

- ADDRess
- FORMat
- MEDia
- PFORmat (printer format)
- PORT

When set to DISK, the summary is written to the disk drive using a user-specified base name with a machine-generated suffix. For example, the results of tests 1, 2, and 3 may be written to files SUMM0001.SUM, SUMM0002.SUM, and SUMM0003.SUM, respectively. 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 peruse on a computer, read into a spreadsheet, etc.

<filename> A file prefix of four characters defaulted to SUMM.

Example The following example saves the summary to a disk file named TEST0000.SUM.

```
10  OUTPUT 707;":LTEST:SSUMMARY DISK,TEST"
20  END
```

Query

```
:LTESt:SSUMmary?
```

The query returns the current specified destination for the summary.

Returned Format

```
[:LTESt:SSUMmary] {OFF | PRINter | DISK [,<filename>]}<NL>
```

Example The following example returns the current destination for the summary and prints the result to the controller's screen.

```
10  DIM SUMM$[50]
20  OUTPUT 707;":LTEST:SUMMARY?"
30  ENTER 707;SUMM$
40  PRINT SUMM$
50  END
```

SSUMmary:ADDRess

Command

```
:LTESt:SSUMmary:ADDRess <address_value>
```

Selects the HP-IB address for the printer. This address is used only when the summary is going to the printer and the Port command is set to HP-IB.

<address_value> Any HP-IB address from 0 to 30.

Example The following example sets the HP-IB address for the printer to 3.

```
10  OUTPUT 707;":LTEST:SSUMMARY:ADDRESS 3"
20  END
```

Query

```
:LTESt:SSUMmary:ADDRess?
```

The query returns the current address of the printer.

Returned Format

```
[:LTESt:SSUMmary:ADDRess]<address_value><NL>
```

Example The following example returns the current HP-IB address of the printer and prints the result to the controller's screen.

```
10  DIM SUMMADD$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:ADDRESS?"
30  ENTER 707;SUMMADD$
40  PRINT SUMMADD$
50  END
```

SSUMmary:FORMat

Command

```
:LTESt:SSUMmary:FORMat {BRIef | STAtistics}
```

Specifies whether the summary file is either brief or contains statistics. BRIef is a one line log of the failure. STATistics includes the measurement statistics, plus the one line log. It also changes the results of the MEASure:RESults? query.

Example

The following example specifies a brief summary file.

```
10  OUTPUT 707;":LTESt:SSUMmary:FORMat BRIef"
20  END
```

Query

```
:LTESt:SSUMmary:FORMat?
```

The query returns the current summary file format.

Returned Format

```
[:LTESt:SSUMmary:FORMat] {BRIef | STATistics}<NL>
```

Example

The following example returns the current summary file format and prints the result to the controller's screen.

```
10  DIM SUMMFORM$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:FORMAT?"
30  ENTER 707;SUMMFORM$
40  PRINT SUMMFORM$
50  END
```

SSUMmary:MEDia

Command

```
:LTESt:SSUMmary:MEDia {PAPer | TRANsparency}
```

Specifies whether paper or transparency film will be used in the printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the media look darker and slows down the printing process. This control is used when the store summary is directed to the printer, and is valid only for the HP PaintJet and color DeskJet printers.

Example

The following example specifies that transparency media should be in the printer when the summary is printed.

```
10  OUTPUT 707;":LTEST:SSUMMARY:MEDIA TRANSPARENCY"
20  END
```

Query

```
:LTESt:SSUMmary:MEDia?
```

The query returns the current specified media for the printer.

Returned Format

```
[:LTESt:SSUMmary:MEDia]{PAPer | TRANsparency}<NL>
```

Example

The following example returns the current media requirement for the printer and prints the result to the controller's screen.

```
10  DIM SUMMMED$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:MEDIA?"
30  ENTER 707;SUMMMED$
40  PRINT SUMMMED$
50  END
```

SSUMmary:PFORmat

Command

```
:LTESt:SSUMmary:PFORmat {THINkjet | PAINtjet | LASerjet | EPSon |
DJ540 | DJ550 | DJ560 | BWPaintjet | BWDeskjet}
```

Selects the printer format to use when storing the summary to a printer. This includes the physical printers supported by the instrument with the exception of the HP DeskJet 500C with the color cartridge installed. To set up the format for the HP DeskJet 500C, use the monochrome cartridge and the BWDeskjet parameter in the command.

Example

The following example sets the printer format for the summary file to HP PaintJet.

```
10  OUTPUT 707;":LTEST:SSUMMARY:PFORMAT PAINTJET"
20  END
```

Query

```
:LTESt:SSUMmary:PFORmat?
```

The query returns the current specified printer format.

Returned Format

```
[:LTESt:SSUMmary:PFORmat] {THINkjet | PAINtjet | LASerjet | EPSon
| DJ540 | DJ550 | DJ560 | BWPaintjet | BWDeskjet}<NL>
```

Example

The following example returns the current printer format for the summary file and prints the result to the controller's screen.

```
10  DIM SUMMPFORM$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:PFORMAT?"
30  ENTER 707;SUMMPFORM$
40  PRINT SUMMPFORM$
50  END
```

SSUMmary:PORT

Command

```
:LTESt:SSUMmary:PORT {CENTronics | HPIB}
```

Selects the printer port for the summary when the summary is sent to the printer.

Example

The following example sets the printer port for the summary to HP-IB.

```
10  OUTPUT 707;":LTEST:SSUMMARY:PORT HPIB"
20  END
```

Query

```
:LTESt:SSUMmary:PORT?
```

The query returns the current specified port for the printer.

Returned Format

```
[:LTESt:SSUMmary:PORT]{CENTronics | HPIB} <NL>
```

Example

The following example returns the current printer port for the summary file and prints the result to the controller's screen.

```
10  DIM SUMMPORT$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:PORT?"
30  ENTER 707;SUMMPORT$
40  PRINT SUMMPORT$
50  END
```

SWAVeform

Command

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

Saves waveforms from a channel, function, histogram, 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.

<source>

```
{CHANnelN | FUNCtionN | FFT  | HISTogram | WMEMoryN}
```

<destination>

```
{OFF | WMEMoryN | DISK}
```

<filename>

A descriptive file prefix consisting of up to four characters. If no filename is specified, the prefix defaults to CH1A . . . CH4B, FN1, FN2, FFT, HIST, MEM1 . . . MEM4.

<format>

```
{TEXT [,YVALues | VERBose] | INTernal}
```

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"
20  END
```

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$[50]
20  OUTPUT 707;":LTEST:SWAVEFORM? CHANNEL1"
30  ENTER 707;SWAV$
40  PRINT SWAV$
50  END
```

TEST

Command

```
:LTESt:TEST {ON | 1 | OFF | 0}
```

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.

Example

The following example turns off the limit test function.

```
10  OUTPUT 707;":LTEST:TEST OFF"
20  END
```

Query

```
:LTESt:TEST?
```

The query returns the state of the TEST control.

Returned Format

```
[:LTESt:TEST] {1 | 0} <NL>
```

Example

The following example returns the current state of the limit test (on or off, 1 or 0, respectively) and prints the result to the controller's screen.

```
10  DIM TEST$[50]
20  OUTPUT 707;":LTEST:TEST?"
30  ENTER 707;TEST$
40  PRINT TEST$
50  END
```

NOTE

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

ULIMit

Command

```
:LTESt:ULIMit <upper_value>
```

Sets the upper test limit for the active measurement currently selected by the last :LTESt:SOURce command.

<upper_value>

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"
20  END
```

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 and prints the result to the controller's screen.

```
10  DIM ULIM$[50]
20  OUTPUT 707;":LTEST:ULIMIT?"
30  ENTER 707;ULIM$
40  PRINT ULIM$
50  END
```

15

Marker Commands

Marker Commands

The Marker subsystem commands are used to specify and query the settings of the time (X-axis) and current measurement unit (volts, amps, watts, and so on, for the Y-axis) markers. The Y-axis measurement units are typically set using the CHANnel:UNITs command.

The Marker subsystem contains the following commands and queries:

- CURSor?
- MEASurement:READout
- MODE
- TDELta?
- TSTArt
- TSTOp
- VDELta?
- VSTArt
- VSTOp
- X1Position
- X2Position
- X1Y1source
- X2Y2source
- XDELta?
- XUNits?
- Y1Position
- Y2Position
- YDELta?
- YUNits?

CURSor?

Query

```
:MARKer:CURSor? {DELTa | STARt | STOP}
```

The query returns the time and current measurement unit values of the specified marker or cursor (if in the waveform mode) as an ordered pair of time and measurement unit values.

If DELTA is specified, the value of delta Y and delta X are returned. If START is specified, the position of X1 (or the + cursor) and measurement unit marker 1 (Y1) are returned. If STOP is specified, the positions of X2 (or the X cursor) and measurement unit marker 2 (Y2) are returned.

Returned Format

```
[:MARKer:CURSor] {DELTa | STARt | STOP} {<X1, Y1> | <X2,Y2>
 | <deltaX, deltaY>}<NL>
```

Example

The following example returns the current position of the + cursor and measurement unit marker 1 to the string variable, Position$, then prints the contents of the variable to the controller's screen.

```
10  DIM Position$[50]              !Dimension variable
20  OUTPUT 707;":MARKER:CURSOR? START"
30  ENTER 707;Position$
40  PRINT Position$
50  END
```

MEASurement:READout

Command

```
:MARKer:MEASurement:READout {{ON | 1} | {OFF | 0}}
```

Controls the state of the marker readout area on the screen when markers are on in measurement mode.

Query

```
:MARKer:MEASurement:READout?
```

The query returns the current state of the marker readout display.

Returned Format

```
{:MARKer:MEASurement:READout] {1 | 0}<NL>
```

MODE

Command

```
:MARKer:MODE {OFF | MANual | WAVeform | MEASurement | HISTogram |
 TDR/TDT}
```

The TDR/TDT function is only available with the HP 54750A, or if the HP 54755K option is installed in an HP 83480A.

Sets the marker mode.

Example

The following example sets the marker mode to waveform.

```
10  OUTPUT 707;":MARKER:MODE WAVEFORM"
20  END
```

Query

```
:MARKer:MODE?
```

The query returns the current marker mode.

Returned Format

```
[:MARKer:MODE] {OFF | MANual | WAVeform | MEASurement | HISTogram |
 TDR/TDT}<NL>
```

Example

The following example places the current marker mode in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":MARKER:MODE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

TDELta?

Query

```
:MARKer:TDELta?
```

The query returns the time difference between X1 and X2 time markers or between the X cursor and + cursor, depending on the current marker mode.

NOTE

The MARKer:TDELta? query performs the same function as the MARKer:XDELta? query. MARKer:TDELta? is provided for compatibility with older instruments. MARKer:XDELta? should be used for new programs.

Returned Format

```
[:MARKer:TDELta] <time><NL>
```

<time>

The time difference between X1 and X2 time markers or between the X cursor and + cursor, depending on the current marker mode.

Example

The following example places the time difference between the X1 and X2 markers in the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:TDELTA?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

NOTE

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TSTArt

Command

```
:MARKer:TSTArt <X1 position>
```

Sets the X1 position and moves the X1 marker or X cursor, depending on the current marker mode.

NOTE

The MARKer:TSTArt command and query perform the same function as the MARKer:X1Position command and query. MARKer:TSTArt is provided for compatibility with previous instruments. MARKer:X1Position should be used for new programs.

<X1 position>

Time at X1 marker or X cursor in seconds.

Example

The following example sets the X1 marker at 90 ns.

```
10  OUTPUT 707;":MARKER:TSTART 90E-9"
20  END
```

Query

```
:MARKer:TSTArt?
```

The query returns the time at X1 or X cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:TSTArt] <X1 position><NL>
```

Example

The following example places the current setting of the X cursor in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:TSTART?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

NOTE

The short form of the TSTArt command and query do not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP. Sending "TST" produces an error.

TSTOp

Command

```
:MARKer:TSTOp <X2 position>
```

Sets the X2 position and moves the X2 marker or + cursor, depending on the current marker mode.

NOTE

The MARKer:TSTOp command and query perform the same function as the MARKer:X2Position command and query. MARKer:TSTOp is provided for compatibility with previous instruments. MARKer:X2Position should be used for new programs.

<X2 position> Time at X2 marker or + cursor in seconds.

Example The following example sets the X2 marker at 90 ns.

```
10  OUTPUT 707;":MARKER:TSTOP 90E-9"
20  END
```

Query

```
:MARKer:TSTOp?
```

The query returns the time at X2 or + cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:TSTOp] <X2 position><NL>
```

Example The following example places the current setting of the + cursor in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:TSTOP?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

NOTE

The short form of the TSTOp command and query do not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP. Sending "TST" produces an error.

VDELta?

Query

```
:MARKer:VDELta?
```

The query returns the current measurement unit difference between Y1 and Y2.

NOTE

The MARKer:VDELta query performs the same function as the MARKer:YDELta query. MARKer:VDELta is provided for compatibility with previous instruments. MARKer:YDELta should be used for new programs.

Returned Format

```
[:MARKer:VDELta] <value><NL>
```

<value>

Current measurement unit difference between Y1 and Y2.

Example

The following example returns the voltage difference between Y1 and Y2 to the numeric variable, Volts, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:VDELTA?"
30  ENTER 707;Volts
40  PRINT Volts
50  END
```

VSTArt

Command

```
:MARKer:VSTArt <Y1 position>
```

Sets the Y1 Position and moves Y1 to the specified measurement unit value on the specified source.

NOTE

The MARKer:VSTArt command and query perform the same function as the MARKer:Y1Position command and query. MARKer:VSTArt is provided for compatibility with previous instruments. MARKer:Y1Position should be used for new programs.

<Y1 position> Current measurement unit value at Y1.

Example The following example sets Y1 to −10 mV.

```
10  OUTPUT 707;":MARKER:VSTART -10E-3"
20  END
```

Query

```
:MARKer:VSTArt?
```

The query returns the current measurement unit level of Y1.

Returned Format

```
[:MARKer:VSTArt] <Y1 position><NL>
```

Example The following example returns the voltage setting for Y1 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:VSTART?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

NOTE

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP. Sending "VST" produces an error.

VSTOp

Command

```
:MARKer:VSTOp <Y2 position>
```

Sets the Y2 position and moves Y2 to the specified measurement unit value on the specified source.

NOTE

The MARKer:VSTOp command and query perform the same function as the MARKer:Y2Position command and query. MARKer:VSTOp is provided for compatibility with previous instruments. MARKer:Y2Position should be used for new programs.

<Y2 position>

Current measurement unit value at Y2.

Example

The following example sets Y2 to −100 mV.

```
10  OUTPUT 707;":MARKER:VSTOP -100E-3"
20  END
```

Query

```
:MARKer:VSTOp?
```

The query returns the current measurement unit level at Y2.

Returned Format

```
[:MARKer:VSTOp] <Y2 position><NL>
```

Example

The following example returns the voltage at Y2 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:VSTOP?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

NOTE

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP. Sending "VST" produces an error.

X1Position

Command

```
:MARKer:X1Position <X1 position>
```

Sets the X1 position and moves the X1 marker or X cursor, depending on the current marker mode, to the specified time with respect to the trigger time.

<X1 position>

Time at X1 marker or X cursor in seconds.

Example

The following example sets the X1 marker at 90 ns.

```
10  OUTPUT 707;":MARKER:X1POSITION 90E-9"
20  END
```

Query

```
:MARKer:X1Position?
```

The query returns the time at X1 or the X cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:X1Position] <X1 position><NL>
```

Example

The following example returns the current setting of the X cursor to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:X1POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

MARKer:TSTArt

X2Position

Command

```
:MARKer:X2Position <X2 position>
```

Sets the X2 position and moves the X2 marker or + cursor, depending on the current marker mode, to the specified time with respect to the trigger time.

<X2 position>

Time at X2 marker or + cursor in seconds.

Example

The following example sets the X2 marker to 90 ns.

```
10  OUTPUT 707;":MARKER:X2POSITION 90E-9"
20  END
```

Query

```
:MARKer:X2Position?
```

The query returns the time at X2 or the + cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:X2Position] <X2 position><NL>
```

Example

The following example returns the current position of the + cursor to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:X2POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

X1Y1source

Command

```
:MARKer:X1Y1source {CHANnel<number> | FUNCtion<number> |
WMEMory<number> | HISTogram | FFT}
```

Sets the source for the X1 and Y1 markers.

<number>

For channels: an integer 1 through 4.

For functions: 1 or 2.

For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example selects channel 1 as the X1Y1 source.

```
10  OUTPUT 707;":MARKER:X1Y1SOURCE CHANNEL1"
20  END
```

Query

```
:MARKer:X1Y1source?
```

The query returns the current X1Y1 source.

Returned Format

```
[:MARKer:X1Y1source] {CHANnel<number> | FUNCtion<number>
| WMEMory<number>  | HISTogram | FFT}
```

Example

The following example returns the current X1Y1 source selection to the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":MARKER:X1Y1SOURCE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

X2Y2source

Command

```
:MARKer:X2Y2source {CHANnel<number> | FUNCtion<number> |
WMEMory<number>  | HISTogram | FFT}
```

Sets the source for the X2 and Y2 markers.

<number>

For channels: an integer 1 through 4.

For functions: 1 or 2.

For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example selects channel 1 as the X2Y2 source.

```
10  OUTPUT 707;":MARKER:X2Y2SOURCE CHANNEL1"
20  END
```

Query

```
:MARKer:X2Y2source?
```

The query returns the current X2Y2 source.

Returned Format

```
[:MARKer:X2Y2source] {CHANnel<number> | FUNCtion<number>
| WMEMory<number> | HISTogram | FFT}
```

Example

The following example returns the current X2Y2 source selection to the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":MARKER:X2Y2SOURCE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

XDELta?

Query

```
:MARKer:XDELta?
```

The query returns the time difference between X1 and X2 time markers or between the X cursor and + cursor, depending on the current marker mode.

Manual Mode

Xdelta = time at X2 – time at X1

Tracking Mode

Xdelta = time at + cursor – time at X cursor

Returned Format

```
[:MARKer:XDELta] <time><NL>
```

<time>

The time difference between X1 and X2 time markers or between the X cursor and + cursor, depending on the current marker mode.

Example

The following example returns the current time between the X1 and X2 time markers to the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:XDELTA?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

XUNits?

Query

```
:MARKer:XUNits?
```

The query returns the x-axis units of the currently selected marker mode.

Returned Format

```
[:MARKer:XUNits] {SECond | METer | FEET}
```

Y1Position

Command

```
:MARKer:Y1Position <Y1 position>
```

Sets the Y1 Position and moves Y1 to the specified measurement unit value on the specified source.

<Y1 position>

Current measurement unit value at Y1.

Example

The following example sets Y1 to 10 mV.

```
10  OUTPUT 707;":MARKER:Y1POSITION 10E-3"
20  END
```

Query

```
:MARKer:Y1Position?
```

The query returns the current measurement unit level of Y1.

Returned Format

```
[:MARKer:Y1Position] <Y1 position>
```

Example

The following example returns the voltage setting for Y1 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:Y1POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

Y2Position

Command

```
:MARKer:Y2Position <Y2 position>
```

Sets the Y2 position and moves Y2 to the specified measurement unit value on the specified source.

<Y2 position>

Current measurement unit value at Y2.

Example

The following example sets Y2 to −100 mV.

```
10  OUTPUT 707;":MARKER:Y2POSITION -100E-3"
20  END
```

Query

```
:MARKer:Y2Position?
```

The query returns the current measurement unit level at Y2.

Returned Format

```
[:MARKer:Y2Position] <Y2 position><NL>
```

Example

The following example returns the voltage at Y2 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:Y2POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

YDELta?

Query

```
:MARKer:YDELta?
```

Returns the current measurement unit difference between Y1 and Y2.

Vdelta = value at Y2 − value at Y1

Returned Format

```
[:MARKer:YDELta] <value><NL>
```

<value> Current measurement unit difference between Y1 and Y2.

Example The following example returns the voltage difference between Y1 and Y2 to the numeric variable, Volts, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:YDELTA?"
30  ENTER 707;Volts
40  PRINT Volts
50  END
```

YUNits?

Query

```
:MARKer:YUNits?
```

The query returns the y-axis units of the currently selected marker mode.

Returned Format

```
[:MARKer:YUNits] {VOLT | OHM | REFlect}
```

16

Mask Test Commands

Mask Test Commands

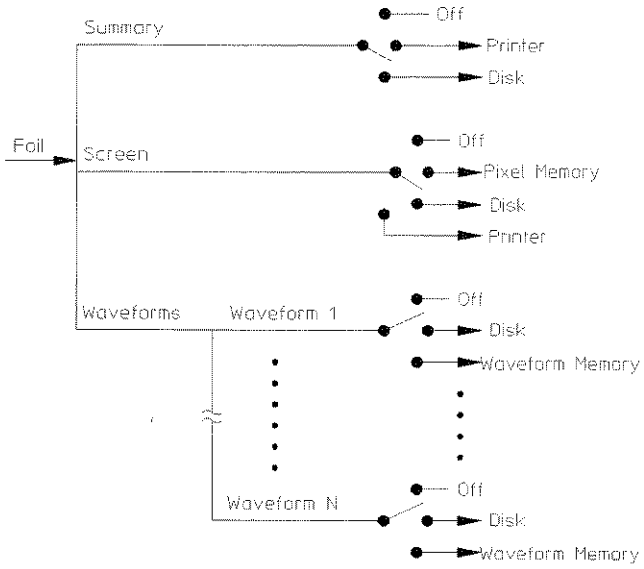
The Mask Test commands and queries control the mask test features. Mask testing automatically compares measurement results with the boundaries of a set of polygons that you define. Any waveform or sample that falls within the boundaries of one or more polygons is recorded as a failure.

The Mask Test subsystem contains the following commands:

- ALIGn (*HP 83480A Only*)
- AMASk
  - CReate
  - SOURce
  - UNITs
  - XDELta
  - YDELta
- AMODe (*HP 83480A Only*)
- COUNt
  - FAILures?
  - FSAMples
  - FWAVeforms?
  - SAMPles?
  - WAVeforms?
- FENable (*HP 83480A Only*)
- MASK
  - DEFine
  - DELete
- MMARgin
  - PERCent
  - STATe
- POLYgon
  - DEFine (*HP 83480A Only*)
  - DELete POLYgon
  - MOVE
- RECall
- RUMode
- SAVE

- SCALe
  - DEFault
  - SOURce
  - X1
  - XDELta
  - Y1
  - Y2
- SSCReen
  - DDISk
    - BACKground
    - MEDia
    - PFORmat
  - DPRinter
    - ADDRess
    - BACKground
    - MEDia
    - PFORmat
    - PORT
- SSUMmary
  - ADDRess
  - MEDia
  - PFORmat
  - PORT
- STANdard (*HP 83480A Only*)
- SWAVeform
  - RESet
- TEST

The HP 83480A and 54750A with Option 83480K provide a series of standard masks defined according to telecom and datacom standards. For a complete list of masks and templates refer to the *HP 83480A, HP 54750A User's Guide*.



54720m12b

Mask Test Subsystem Functional Diagram

Building Polygon Masks

The instrument allows you to build a mask using either polygons or a reference waveform or to use a predefined mask (HP 83480A only). With the polygon method, you use polygons to mask off failure regions on the graticule area. You can position up to eight polygons on the graticule area, and each polygon can have from 3 to 512 sides. You use the :MTESt:POLYgon:DEFine and :MTESt:MASk:DEFine commands to define polygons.

The polygon method is typically used for telecommunications applications because of the flexibility polygons give in designing a mask. For example, you can construct very complicated masks with polygons, or you can place polygons within polygons. Placing polygons within polygons allows you to test waveform failure rates to different tolerances because you can obtain separate failure statistics for each polygon, using the :MTESt:COUNt:FAILures? query. For example, an outer polygon could represent a 1% tolerance, and two inner polygons could represent a 5% tolerance and 10% tolerance, respectively.

With the reference waveform method, you construct a mask by adding a ΔX and ΔY tolerance around your reference waveform, using the :MTESt:AMASk subsystem commands. The reference waveform method is simpler to use but less flexible than the polygon method.

Mask handling

The instrument has three features that use a specific database. This database uses a different memory area than the waveform record for each channel. The three features that use the database are histograms, mask testing, and color-graded display. When any one of these three features is turned on, the instrument starts building the database. The database is the size of the graticule area, which is 256 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 DISPlay:CGRade ON command to enable the color-graded display feature. 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-graded display, mask testing, and histograms) are turned off. The instrument stops acquiring data when the power is cycled, the Stop/Single hardkey is pressed, or the Run Until softkey in the mask or histograms menu is set to stop acquiring data, after a specified number of waveforms or samples are acquired.

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. 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. You can separate waveforms by setting the display to two graphs or by positioning the waveforms vertically with the channel offset. By separating the waveforms, you can avoid overlapping data in the database caused by multiple waveforms.

To avoid erroneous data, clear the display after you change instrument setup conditions or DUT conditions and acquire new data before extracting measurement results.

Mask File Format

Because polygon masks can be complicated, it's usually easier to define them and save them in a disk file, then retrieve them from the disk when needed for a measurement. You load a mask file using the DISK:LOAD command. Mask files have the .MSK extension and have four parts:

- A mask title up to twenty characters (optional). This title will be displayed below the graticule when the mask is loaded.
- A polygon identifier, 1 through 8.
- The number of vertices for the polygon being defined.
- A series of X-Y coordinates defining the vertices. These are floating point numbers. The special values MIN and MAX automatically set a vertex X or Y coordinate to the boundary of the graticule, even if the scaling is changed with the :MTESt:SCALe commands.

  Mask files can contain "C" style comments, which are enclosed in /* and */. Commas, tabs, spaces, or newlines are valid separators between parameters.

NOTE

The HP 83480A and HP 54750A with Option 83480K provide a series of standard masks defined according to telecom and datacom standards. For a complete list of mask and templates, refer to the user's guide.

The following is a sample mask file for the DS1 E telecommunications waveform:

```
"DS1Eur 2048 kbit/s"
/*
     Physical/Electrical Characteristics of Heirarchical
     Digital Interface. (Geneva 1972 : further ammended)
     Recommendation G.703
*/
/*   Level1 Peak 2.37V   */
/*   Level2      0.00V   */
/*   Delta X     244 ns  */

/* Top Polygon           */   1
/* Number of vertices    */   9
     -0.5,      Max      /* Top of screen left side     */
     -0.5,     +0.1
     -0.0512,  +0.5
     -0.0512,  +1.2
     +0.5,     +1.1
     +1.0512,  +1.2
     +1.0512,  +0.5
     +1.5,     +0.1
     +1.5,      Max      /* Top of screen right side    */

/* Bottom Polygon        */   2
/* Number of vertices    */   11
     -0.5,      Min      /* Bottom of screen left side  */
     -0.5,     -0.1
     +0.05123, -0.2
     +0.05123, +0.5
     +0.10246, +0.8
     +0.5,     +0.9
     +0.8975,  +0.8
     +0.94877, +0.5
     +0.94877, -0.2
     +1.5,     -0.1
     +1.5,      Min     /* Bottom of screen right side */
```

ALIGn (HP 83480A Only)

Command

```
:MTESt:ALIGn
```

Automatically aligns and scales the mask to the current waveform on the display or scales the waveform and scales max to fill the display depending on alignment mode. (See AMODe.)

Example

The following example aligns the current mask to the current waveform.

```
10  OUTPUT 707;":MTEST:ALIGN"
20  END
```

AMASk:CReate

Command

```
:MTESt:AMASk:CReate
```

Automatically constructs a mask around the current waveform on the display, using the tolerance parameters defined by the AMASk:XDELta, AMASk:YDELta, and AMASK:UNITs commands. The mask only encompasses the portion of the waveform visible on the display, so your measurement must ensure that the waveform is acquired and displayed consistently to obtain repeatable results.

Example

The following example defines an automask given the current ΔX, ΔY, and units settings.

```
10  OUTPUT 707;":MTEST:AMASK:CREATE"
20  END
```

AMASk:SOURce

Command

```
:MTESt:AMASk:SOURce {WMEMory<number> | FUNCtion<number>
| CHANnel<number> | FFT}
```

Selects the source for the interpretation of the AMASk:XDELta and AMASk:YDELta parameters when AMASk:UNITs is set to CURRent. When UNITs are CURRent, the XDELta and YDELta parameters are defined in terms of the measurement system of the selected source. Suppose that UNITs are CURRent and that you set SOURce to CHANNEL1, which is using volts as a measurement system. Then you can define AMASk:XDELta in terms of volts and AMASk:YDELta in terms of seconds.

<number>

For channels: the number represents an integer, 1 through 4.

For waveform memories: 1, 2, 3, or 4.

For functions: 1 or 2.

Example

The following program sets the automask measurement system source to Channel 1.

```
10  OUTPUT 707;":MTEST:AMASK:SOURCE CHANNEL1"
20  END
```

Query

```
:MTESt:AMASk:SOURce?
```

The query returns the currently set source.

Returned Format

```
[:MTESt:AMASk:SOURce] {WMEMory<number> | FUNCtion<number>
| CHANnel<number> | FFT} <NL>
```

Example

The following program gets the source setting for automask and prints the result on the controller display.

```
10  DIM AMASK_SOURCE$[30]
20  OUTPUT 707;":MTEST:AMASK:SOURCE?"
30  ENTER 707;AMASK_SOURCE$
40  PRINT AMASK_SOURCE$
50  END
```

AMASk:UNITs

Command

```
:MTESt:AMASk:UNITs {CURRent | DIVisions}
```

Alters the way the mask test subsystem interprets the ΔX and ΔY tolerance parameters for automasking, defined by AMASk:XDELta and AMASk:YDELta, respectively.

CURRent

When set to CURRent, the mask test subsystem uses the measurement system currently in use, usually time for ΔX and voltage for ΔY.

DIVisions

When set to DIVisions, the mask test subsystem uses the graticule as the measurement system, so tolerance settings are specified as parts of a screen division. The mask test subsystem maintains separate XDELta and YDELta settings for CURRent and DIVisions. Thus, XDELta and YDELta are not converted to new values when the UNITs setting is changed.

Example

The following example sets the measurement units for automasking to the current measurement system.

```
10  OUTPUT 707;":MTEST:AMASK:UNITS CURRENT"
20  END
```

Query

```
:MTESt:AMASk:UNITs?
```

The query returns the current measurement units setting for the mask test automask feature.

Returned Format

```
[:MTESt:AMASk:UNITs] {CURRent | DIVision} <NL>
```

Example

The following example gets the current automask units setting, then prints the setting on the screen of the controller.

```
10  DIM AUTOMASK_UNITS$[10]
20  OUTPUT 707;":MTEST:AMASK:UNITS?"
30  ENTER 707;AUTOMASK_UNITS$
40  PRINT AUTOMASK_UNITS$
50  END
```

AMASk:XDELta

Command

```
:MTESt:AMASk:XDELta <xdelta_value>
```

Sets the tolerance in the X direction around the waveform for the automasking feature. The absolute value of the tolerance will be added and subtracted to X components of the waveform to determine the boundaries of the mask.

<xdelta_value>

A value for the X tolerance. This value is interpreted based on the setting specified by the AMASk:UNITs command; thus, if you specify 250-E3, the setting for AMASk:UNITs is CURRent, and the current measurement system specifies time in the X direction, the tolerance will be ±250 ms. If the setting for AMASk:UNITs is DIVisions, the same xdelta_value will set the tolerance to ±250 millidivisions, or 1/4 of a division.

Example

The following example sets the current measurement system to divisions and sets the ΔX tolerance to one-eighth of a division.

```
10  OUTPUT 707;":MTEST:AMASK:UNITS DIVISIONS"
20  OUTPUT 707;":MTEST:AMASK:XDELTA 125E-3"
30  END
```

Query

```
:MTESt:AMASk:XDELta?
```

The query returns the current setting of the ΔX tolerance for automasking. If your controller program will interpret this value, it should also request the current measurement system using the AMASk:UNITs query.

Returned Format

```
[:MTESt:AMASk:XDELta] <xdelta_value> <NL>
```

Example

The following example gets the measurement system units and ΔX settings for automasking from the instrument and prints the results on the controller screen.

```
10  DIM AUTOMASK_UNITS$[10]
20  DIM AUTOMASK_XDELTA$[20]
30  OUTPUT 707;":MTEST:AMASK:UNITS?"
40  ENTER 707;AUTOMASK_UNITS$
50  OUTPUT 707;":MTEST:AMASK:XDELTA?"
60  ENTER 707;AUTOMASK_XDELTA$
70  PRINT AUTOMASK_UNITS$
80  PRINT AUTOMASK_XDELTA$
90  END
```

AMASk:YDELta

Command

```
:MTESt:AMASk:YDELta <ydelta_value>
```

Sets the tolerance in the Y direction around the waveform for the automasking feature. The absolute value of the tolerance will be added and subtracted to Y components of the waveform to determine the boundaries of the mask.

<ydelta_value>

A value for the Y tolerance. This value is interpreted based on the setting specified by the AMASk:UNITs command; thus, if you specify 250-E3, the setting for AMASk:UNITs is CURRent, and the current measurement system specifies voltage in the Y direction, the tolerance will be ±250 mV. If the setting for AMASk:UNITs is DIVisions, the same ydelta_value will set the tolerance to ±250 millidivisions, or 1/4 of a division.

Example

The following example sets the current measurement system to current and sets the ΔY tolerance to 30 mV, assuming that the current measurement system specifies volts in the Y direction.

```
10  OUTPUT 707;":MTEST:AMASK:UNITS CURRENT"
20  OUTPUT 707;":MTEST:AMASK:YDELTA 30E-3"
30  END
```

Query

```
:MTESt:AMASk:YDELta?
```

The query returns the current setting of the ΔY tolerance for automasking. If your controller program will interpret this value, it should also request the current measurement system using the AMASk:UNITs query.

Returned Format

```
[:MTESt:AMASk:YDELta] <ydelta_value> <NL>
```

Example

The following example gets the measurement system units and ΔY settings for automasking from the instrument and prints the results on the controller screen.

```
10  DIM AUTOMASK_UNITS$[10]
20  DIM AUTOMASK_YDELTA$[20]
30  OUTPUT 707;":MTEST:AMASK:UNITS?"
40  ENTER 707;AUTOMASK_UNITS$
50  OUTPUT 707;":MTEST:AMASK:YDELTA?"
60  ENTER 707;AUTOMASK_YDELTA$
70  PRINT AUTOMASK_UNITS$
80  PRINT AUTOMASK_YDELTA$
90  END
```

MTEST:AMODe (HP 83480A Only)

Requires firmware revision A.02.00 and above.

Command

```
:MTEST:AMODe {MASK | FILL}
```

Sets the mode for automated mask alignment. The Mask mode aligns the mask to the waveform. The Fill mode scales the waveform to optimize the display of the mask.

Example

The following example sets the alignment mode to align mask to waveform.

```
10  OUTPUT 707;":MTEST:AMODE MASK"
20  END
```

Query

```
:MTEST:AMODe?
```

The query returns the current setting of alignment mode.

Returned Format

```
[:MTEST:AMODe] {MASK | FILL}
```

Example

The following example gets the current setting of the mask alignment mode and prints it on the screen.

```
10  DIM ALIGN_MODE$[4]
20  OUTPUT 707;":MTEST:AMODE?"
30  ENTER 707;ALIGN_MODE$
40  PRINT ALIGN_MODE$
50  END
```

COUNt:FAILures?

Query

```
:MTESt:COUNt:FAILures? POLYgon<number>
```

The query returns the number of failures that occurred within a particular polygon. By defining polygons within polygons, then counting the failures for each individual polygon, 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 polygon number that is not used.

<number> An integer, 1 through 8, designating the polygon for which you want to determine the failure count.

Returned Format

```
[:MTESt:COUNt:FAILures] POLYgon<number><number_of_failures><NL>
```

<number_of_failures> The number of failures that have occurred for the designated polygon.

Example The following example determines the current failure count for polygon 3 and prints it on the controller screen.

```
10  DIM MASK_FAILURES$[50]
20  OUTPUT 707;":MTEST:COUNT:FAILURES? POLYGON3"
30  ENTER 707;MASK_FAILURES$
40  PRINT MASK_FAILURES$
50  END
```

COUNt:FSAMples?

Query

```
:MTESt:COUNt:FSAMples?
```

The query returns the total number of failed samples in the current mask test run. This count is for all polygons and all waveforms, so if you wish to determine failures by polygon number, use the COUNt:FAILures? query.

The count value returned is not the sum of the failure counts for each polygon. For example, assume a polygon 2 enclosed completely by polygon 1. If polygon 1 has 100 failures, the value returned is 100, regardless of how many failures are in polygon 2. Because polygon 2 is completely enclosed, the failure count for polygon 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>

The total number of failed samples for the current test run.

Example

The following example determines the number of failed samples and prints the result on the controller screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"
20  OUTPUT 707;":MTEST:COUNT:FSAMPLES?"
30  ENTER 707;MASK_FSAMPLES
40  PRINT MASK_FSAMPLES
50  END
```

COUNt:FWAVeforms?

Query

```
:MTESt:COUNt:FWAVeforms?
```

The query returns the total number of failed waveforms in the current mask test run. This count is for all polygons and all waveforms, so if you wish to determine failures by polygon number, use the COUNt:FAILures? query.

This count may not always be available. It is available only when the following conditions are true:

- mask testing was turned on before the histogram or color-graded display, and
- no mask changes have occurred, including scaling changes, editing, or new masks.

The value 9.999E37 is returned if mask testing is not enabled, or if you have modified the mask.

Returned Format

```
[:MTESt:COUNt:FWAVeforms] <number_of_failed_waveforms><NL>
```

<number_of_failed_waveforms>

The total number of failed waveforms for the current test run.

Example

The following example determines the number of failed waveforms and prints the result on the controller screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"
20  OUTPUT 707;":MTEST:COUNT:FWAVEFORMS?"
30  ENTER 707;MASK_FWAVEFORMS
40  PRINT MASK_FWAVEFORMS$
50  END
```

COUNt:SAMPles?

Query

```
:MTESt:COUNt:SAMPles?
```

The query 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>

The total number of samples for the current test run.

Example

The following example determines the number of samples gathered in the current test run and prints the result on the controller screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"
20  OUTPUT 707;":MTEST:COUNT:SAMPLES?"
30  ENTER 707;Mask_samples
40  PRINT Mask_samples
50  END
```

COUNt:WAVeforms?

Query

```
:MTESt:COUNt:WAVeforms?
```

The query 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>

The total number of waveforms for the current test run.

Example

The following example determines the number of waveforms gathered in the current test run and prints the result on the controller screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"
20  OUTPUT 707;":MTEST:COUNT:WAVEFORMS?"
30  ENTER 707;Mask_waveforms
40  PRINT Mask_waveforms
50  END
```

FENable

Command

```
:MTESt:FENable {ON | 1 | OFF | 0}
```

Enables the fail action when a mask hit occurs.

Example

The following example enables the fail action.

```
10 OUTPUT 707;":MTEST:FENABLE ON"
20 END
```

Query

```
:MTESt:FENable?
```

The query returns the current failure enable setting.

Returned Format

```
[:MTESt:FENable] {1 | 0}<NL>
```

Example

The following example determines the failure enable setting and prints the result on the controller screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"
20  OUTPUT 707;":MTEST:FENable?"
30  ENTER 707;Fail_enable$
40  PRINT Fail_enable$
50  END
```

MASK:DEFine

Command

```
:MTESt:MASK:DEFine #6NNNNNN [<mask_label>]<polygon_id>,
<number_of_vertices>,<x_coordinate>,<y_coordinate>[,<x_coordinate>,
<y_coordinate>]*[,<polygon_id>,<number_of_vertices>,<x_coordinate>,
<y_coordinate>[,<x_coordinate>,<y_coordinate>]*]*
```

Defines a set of up to eight polygons to be used for mask testing. When you use the MASK:DEFine command, you can specify polygon definitions for up to eight polygons. You can also use the MTESt:POLYgon:DEFine command to define a single polygon at a time.

The set can be named with an optional mask label. Each polygon in the mask is described by a polygon identifier and a series of vertices, specified as X-Y coordinates. An initial block identifier, starting with #6 and followed immediately by a six-digit integer, specifies the number of bytes of polygon data to follow. You may include "C"-style comments in the polygon data; these comments are enclosed with /* and */. You must include the byte count for the comments in the byte count supplied with the command.

Items in the mask definition must be separated by spaces. As used here, spaces are defined to be character space (decimal 32 ASCII), commas, tabs, or newline characters.

#6NNNNNN The #6 indicates that the immediately following 6-digit integer, represented by NNNNNN, will contain the byte count for the mask data.

<mask_label> An optional string naming this mask definition.

<polygon_id> An integer, 1 through 8, identifying the polygon you wish to define.

<number_of_vertices> The number of vertices that you will supply as X-Y coordinates.

<x_coordinate> A value specifying the X-coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

<y_coordinate> A value specifying the Y-coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

Example The following example shows how to define polygons 1 and 2 as rectangles below and above the Y1 and Y2 boundaries.

```
10  OUTPUT 707 USING "-K";":MTEST:MASK:DEFINE #6000048
    1,4,0,MIN,1,MIN,1,0,0,0,2,4,0,1,1,1,1,MAX,0,MAX"
20  END
```

The "-K" is an image specifier that outputs the string in compact form with no leading or trailing blanks.

Query

```
:MTESt:MASK:DEFine?
```

The query gets the parameters for all currently defined polygons from the instrument.

Returned Format

```
[:MTESt:MASK:DEFine] #6NNNNNN [<mask_label>]
<polygon_id>,<number_of_vertices>,<x_coordinate>,<y_coordinate>
[,<x_coordinate>,<y_coordinate>]*[,<polygon_id>, <number_of_vertices>,
<x_coordinate>,<y_coordinate>[,<x_coordinate>,<y_coordinate>]*]*<NL>
```

Example This example gets the definition of the mask and prints it on the controller screen.

```
10  DIM MASK_DEF$[400]
20  OUTPUT 707;":MTEST:MASK:DEFINE?"
30  ENTER 707 USING "-K";POLY_DEF$
40  PRINT MASK_DEF$
50  END
```

The "-K" in the ENTER statement is an image specifier that forces entered characters to be placed in the string. Carriage returns, without line-feeds, are also placed in the string without terminating the entry.

MASK:DELete

Command

```
:MTESt:MASK:DELete
```

Deletes the complete currently defined mask.

Example

The following example deletes the currently defined mask.

```
10 OUTPUT 707;":MTEST:MASK:DELETE"
20 END
```

MMARgin:PERCent

Command

```
:MTESt:MMARgin:PERCent <margin_percent>
```

Sets the amount of mask margin to apply to the selected mask.

<margin_percent>

An integer, −100 to 100, expressing the mask margin in percent.

Example

The following example sets the mask margin to 50 percent.

```
10 OUTPUT 707;":MTEST:MMARGIN:PERCENT 50"
20 END
```

Query

```
:MTESt:MMARgin:PERCent?
```

The query returns the current mask margin.

Returned Format

```
[:MTESt:MMARgin:PERCent] <margin_percent>
```

Example

The following example determines the mask margin and prints the result on the controller screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"
20  OUTPUT 707;":MTEST:MMARgin:PERCent?"
30  ENTER 707;Margin
40  PRINT Margin
50  END
```

MMARgin:STATe

Command

```
:MTESt:MMARgin:STATe {ON | 1 | OFF | 0}
```

Controls the activation of the mask margin.

Example

The following example activates the mask margin.

```
10 OUTPUT 707;":MTEST:MMARgin:STATe ON"
20 END
```

Query

```
:MTESt:MMARgin:STATe?
```

The query returns the current mask margin state.

Returned Format

```
[:MTESt:MMARgin:STATe] {1 | 0}
```

Example

The following example determines the mask margin state and prints the result on the controller screen.

```
10  DIM Margin_state$[50]
20  OUTPUT 707;":MTEST:MMARgin:STATe?"
30  ENTER 707;Margin_state$
40  PRINT Margin_state$
50  END
```

POLYgon:DEFine

Command

```
:MTESt:POLYgon:DEFine #6NNNNNN <polygon_id>,
<number_of_vertices>,<x_coordinate>,<y_coordinate>
[,<x_coordinate>,<y_coordinate>]*
```

Defines a polygon to be used for mask testing. You can define one polygon at a time by using this command. If you want to define several polygons at once, you can use the DISK:LOAD command to load a mask file, or use the MTESt:MASK:DEFine command to define a series of polygons.

Each polygon is described by a polygon identifier and a series of vertices, specified as X-Y coordinates. An initial block identifier, starting with #6 and followed immediately by a six-digit integer, specifies the number of bytes of polygon data to follow. You may include "C"-style comments in the polygon data; these comments are enclosed with /* and */. You must include the byte count for the comments in the byte count supplied with the command.

Items in the polygon definition must be separated by spaces. As used here, spaces are defined to be character space (decimal 32 ASCII), commas, tabs, or newline characters.

#6NNNNNN The #6 indicates that the immediately following 6-digit integer, represented by NNNNNN, will contain the byte count for the polygon data.

<polygon_id> An integer, 1 through 8, identifying the polygon you wish to define.

<number_of_vertices> The number of vertices that you will supply as X-Y coordinates.

<x_coordinate> A value specifying the X-coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

<y_coordinate> A value specifying the Y-coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

Example

The following example shows how to define polygon 3 as a triangle.

```
10  OUTPUT 707 USING "-K";":MTEST:POLYGON:DEFINE
    #6000028 3,3,1.0,1.0,0.5,0.7,0.7,0.7"
20  END
```

The "-K" is an image specifier that outputs the string in compact form with no leading or trailing blanks.

Query

```
:MTESt:POLYgon:DEFine? POLYGON <polygon_number>
```

The query gets the parameters for the selected polygon from the instrument. To get the parameters for all defined polygons, use the :MTESt:MASK:DEFine? query.

<polygon_number>

A number, 1 through 8, specifying the polygon for which you want information.

Returned Format

```
[:MTESt:POLYgon:DEFine] #6NNNNNN <polygon_id>,<number_of_vertices>,
<x_coordinate>,<y_coordinate>[,<x_coordinate>,<y_coordinate>]*<NL>
```

Example

This example gets the definition of polygon one and prints it on the controller screen.

```
10  DIM POLY_DEF$[200]
20  OUTPUT 707;":MTEST:POLYGON:DEFINE? POLYGON1"
30  ENTER 707 USING "-K";POLY_DEF$
40  PRINT POLY_DEF$
50  END
```

The "-K" in the ENTER statement is an image specifier that forces entered characters to be placed in the string. Carriage returns, without line-feeds, are also placed in the string without terminating the entry.

POLYgon:DELete POLYgon

Requires firmware revision A.04.00 and above.

Command

```
:MTESt:POLYgon:DELete POLYgon <number>
```

Deletes the specified polygon.

POLYgon <number> Polygon to be moved.

Example The following example deletes Polygon 2.

```
10  OUTPUT 707;":MTEST:POLYGON:DELETE POLYGON2
20  END
```

POLYgon:MOVE

Requires firmware revision A.04.00 and above.

Command

```
:MTESt:POLYgon:MOVE <POLYgon><number>,<xdelta_value>,<ydelta_value>
```

Moves the specified polygon by the X and Y amounts indicated.

POLYgon<number> Polygon to be moved.

<xdelta_value> Amount to move polygon on the X-axis.

<ydelta_value> Amount to move polygon on the Y-axis.

Example The following example moves Polygon 1 +0.1 on the X-axis and −0.2 on the Y-axis.

```
10  OUTPUT 707;":MTEST:POLYGON:MOVE POLYGON1, 0.1, -0.2
20  END
```

RECall

Command

```
:MTESt:RECall <mask_memory>
```

Recalls from the mask memory the mask associated with the mask memory number.

<mask_memory> An integer in the range of 1 to 5.

Example

```
10  OUTPUT 707;":MTEST:RECALL 1"
20  END
```

RUMode

Command

```
:MTESt:RUMode {FORever | FSAMples, <number_of_failed_samples> |
FWAVeforms, <number_of_failed_waveforms> | SAMPles,
<number_of_samples> | WAVeforms, <number_of_waveforms>}
```

Determines the termination conditions for the test. The choices are FORever, FSAMples (FailedSAMples), FWAVeforms (FailedWAVeforms), SAMPles, or WAVeforms.

If FSAMples, FWAVeforms, SAMPles, or WAVeforms are selected, a second parameter is required indicating the number of failures that can occur or the number of samples or waveforms that are to be acquired.

FORever FORever runs the Mask Test until the test is turned off. This is used when you want a measurement to run continually and to not stop after a fixed number of failures. For example, you may want the Mask Test to run overnight and not be limited by a number of failures.

FSAMples FSAMples runs the Mask Test until at least a set number of failed samples occur, that is, samples which fell within the boundaries of one or more polygons. When FSAMples is sent, the test executes until the selected total failures are obtained. After each complete acquisition, the number of failures are compared against this number to test for termination.

Use the FSAMples or FWAVeforms mode when you want the Mask Test to complete after a set number of failures.

<number_of_failed_samples> An integer: 1 to 2,000,000,000.

FWAVeforms FWAVeforms runs the Mask Test until a set number of failed waveforms occur, that is, waveforms which fell within the boundaries of one or more polygons. When FWAVeforms 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 FSAMples or FWAVeforms mode when you want the Mask Test to complete after a set number of failures.

<number_of_failed_waveforms> An integer: 1 to 1,000,000,000.

SAMPles SAMPles sets the minimum number of samples that are required to terminate the test. After each acquisition, the number of samples acquired is compared against this value. SAMPles runs the Mask Test until at least a set number of samples are acquired.

Use the SAMPles or WAVeforms mode when you want the Mask Test to reach completion after a set number of samples waveforms are acquired. The test terminates when the system reaches the specified number of samples or waveforms.

<number_of_samples> An integer: 1 to 2,000,000,000.

WAVeforms

WAVeforms sets the maximum waveforms that are required. When any measurement reaches this number of waveforms, the test terminates. WAVeforms runs the Mask Test until a set number of waveforms are acquired.

Use the SAMPles or WAVeforms mode when you want the Mask Test to reach completion after a set number of samples waveforms are acquired. The test terminates when the system reaches the specified number of samples or waveforms.

<number_of_waveforms>

An integer: 1 to 1,000,000,000.

Example

The following example sets the mask test subsystem Run Until mode to continue testing until 500,000 samples have been gathered.

```
10  OUTPUT 707;":MTEST:RUMODE SAMPLES,500E3"
20  END
```

Query

```
:MTESt:RUMode?
```

The query returns the currently selected termination condition and value.

Returned Format

```
[:MTESt:RUMode] {FOREver | FSAMples, <number_of_failed_samples> |
FWAVeforms, <number_of_failed_waveforms> | SAMPles,
<number_of_samples> | WAVeforms, <number_of_waveforms>}
```

Example

The following example gets the current setting of the mask test Run Until mode from the instrument and prints it on the controller screen.

```
10  DIM MTEST_Runmode$[50]
20  OUTPUT 707;":MTEST:RUMODE?"
30  ENTER 707;MTEST_Runmode$
40  PRINT MTEST_Runmode$
50  END
```

SAVE

Command

```
:MTEST:SAVE <mask_memory>
```

Saves the mask to mask memory.

<mask_memory> An integer in the range of 1 to 5.

Example

```
10  OUTPUT 707;":MTEST:SAVE 1"
20  END
```

SCALe:DEFault

Command

```
:MTESt:SCALe:DEFault
```

Sets the scaling markers to default values. The X1, XDELta, Y1, and Y2 markers are set to values corresponding to graticule positions that are one division in from the left, right, top, and bottom of the graticule, respectively.

Example The following example selects the default scale.

```
10  OUTPUT 707;":MTEST:SCALE:DEFAULT"
20  END
```

SCALe:SOURce

Command

```
:MTESt:SCALe:SOURce {WMEMory<number> | FUNCtion<number>
| CHANnel<number>}
```

Sets the source used by the mask subsystem for interpretation of the SCALe:Y1 and SCALe:Y2 parameters. SCALe:Y1 and SCALe:Y2 set the vertical boundaries of the coordinate system for mask testing, and are affected by the scaling of the selected source. For example, suppose that Y1 was set to −1 V and Y2 was set to +1 V. If Channel 1 was selected as the scaling source, and was set to a vertical scale factor of 100 mV per division, the Y1 and Y2 markers will be below and above the graticule, respectively. If Channel 2 was selected as the scaling source, and was set to a vertical scale factor of 500 mV per division, the Y1 and Y2 markers will be two divisions below and above the center of the graticule, respectively.

Interpretation of the X1 and ΔX settings is done using the time/div setting in the time base subsystem, if the source is a channel. The setting can be queried using the TIMEbase:SCALe? command. Functions and waveform memories can have independent scale.

<number>

For channels: the number represents an integer, 1 through 4.

For waveform memories: 1, 2, 3, or 4.

For functions: 1 or 2.

Example

The following example selects waveform memory 1 as the source for interpretation of the Y1 and Y2 scaling values.

```
10  OUTPUT 707;":MTEST:SCALE:SOURCE WMEMORY1"
20  END
```

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] {WMEMory<number> | FUNCtion<number>
| CHANnel<number>} <NL>
```

Example

The following example gets the current scale source setting from the instrument and prints it on the controller screen.

```
10  DIM Scale_Source$[30]
20  OUTPUT 707;":MTEST:SCALE:SOURCE?"
30  ENTER 707;Scale_source$
40  PRINT Scale_source$
50  END
```

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 polygon masks are defined with respect to this value, according to the equation:

$X = (X \times \Delta X) + X1$

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.

<x1_value>

A time value specifying the location of the X1 coordinate, which will then be treated as X=0 for polygon vertex coordinates.

Example

The following example sets the X1 coordinate at 150 μs.

```
10  OUTPUT 707;":MTEST:SCALE:X1 150E-6"
20  END
```

Query

```
:MTESt:SCALe:X1?
```

The query returns the current X1 coordinate setting.

Returned Format

```
[:MTESt:SCALe:X1] <x1_value> <NL>
```

Example

The following example gets the current setting of the X1 coordinate from the instrument and prints it on the controller screen.

```
10  DIM Scale_x1$[50]
20 OUTPUT 707;":MTEST:SCALE:X1?"
30 ENTER 707;Scale_x1$
40 PRINT Scale_x1$
50 END
```

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 polygons 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 polygon vertices are normalized using the equation:

$X = (X \times \Delta X) + X1$

<xdelta_value>

A time value specifying the distance of the X2 marker with respect to the X1 marker.

Example

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"
20  END
```

Query

```
:MTESt:SCALe:XDELta?
```

The query returns the current value of ΔX.

Returned Format

```
[:MTESt:SCALe:XDELta] <xdelta_value> <NL>
```

Example

The following example gets the value of ΔX from the instrument and prints it on the controller screen.

```
10  DIM Scale_xdelta$[50]
20  OUTPUT 707;":MTEST:SCALE:XDELTA?"
30  ENTER 707;Scale_xdelta$
40  PRINT Scale_xdelta$
50  END
```

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 1 V, a Y value of .100 in a vertex is at 190 mV.

<y1_value>

A voltage value specifying the point at which Y=0.

Example

The following example sets the Y1 marker to −150 mV.

```
10  OUTPUT 707;":MTEST:SCALE:Y1 -150E-3"
20  END
```

Query

```
:MTESt:SCALe:Y1?
```

The query returns the current setting of the Y1 marker.

Returned Format

```
[:MTESt:SCALe:Y1] <y1_value><NL>
```

Example

The following example gets the setting of the Y1 marker from the instrument and prints it on the controller screen.

```
10  DIM Scale_y1$[50]
20  OUTPUT 707;":MTEST:SCALE:Y1?"
30  ENTER 707;Scale_y1$
40  PRINT Scale_y1$
50  END
```

SCALe:Y2

Command

```
:MTESt:SCALe:Y2 <y2_value>
```

Defines the Y2 marker 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 1 V, a Y value of .100 in a vertex is at 190 mV.

<y2_value>

A voltage value specifying the location of the Y2 marker.

Example

The following example sets the Y2 marker to 2.5 V.

```
10  OUTPUT 707;":MTEST:SCALE:Y2 2.5"
20  END
```

Query

```
:MTESt:SCALe:Y2?
```

The query returns the current setting of the Y2 marker.

Returned Format

```
[:MTESt:SCALe:Y2] <y2_value> <NL>
```

Example

The following example gets the setting of the Y2 marker from the instrument and prints it on the controller screen.

```
10  DIM Scale_y2$[50]
20  OUTPUT 707;":MTEST:SCALE:Y2?"
30  ENTER 707;Scale_y2$
40  PRINT Scale_y2$
50  END
```

SSCReen

Command

```
:MTESt:SSCReen  {OFF | PMEMory1 | PRINter | DISK [,<filename>]}
```

Saves a copy of the screen in the event of a failure. A destination of OFF turns off the save action. A destination of PMEMory1 adds the screen to the pixel memory.

When the destination is PRINter, a variety of printer-related controls are used to specify the printer. When the destination is DISK, a different set of commands is provided to control the print to disk. The printer specifications are set through the DPRinter (Destination Printer) and the DDISk (Destination Disk) commands.

<filename>

A file prefix of four characters. If no prefix is specified, the default prefix is SCRN.

Example

The following program sends the screen image to the printer when a failure occurs.

```
10  OUTPUT 707;":MTEST:SSCREEN:PRINTER"
20  END
```

Query

```
:MTESt:SSCReen?
```

The query returns the current state of the SSCreen command.

Returned Format

```
[:MTESt:SSCReen] {OFF | PMEMory1 | PRINter | DISK [,<filename>]}<NL>
```

Example

The following program gets the current destination for the save screen command and prints it on the controller.

```
10  DIM Screen_out$[30]
20  OUTPUT 707;":MTEST:SSCREEN?"
30  ENTER 707;Screen_out$
40  PRINT Screen_out$
50  END
```

The MTESt:SSCReen command contains the following command subsystems, used for handling printer or disk output format:

- DDISk (Display Disk)
- DPRinter (Display Printer)

SSCReen:DDISk

The MTESt:SSCReen:DDISk sub-subsystem commands are used to set up the disk drive when storing the Mask Test display screen to a disk.

The disk drive setup consists of the following commands:

- BACKground
- MEDia
- PFORmat (printer format)

SSCReen:DDISk:BACKground

Command

```
:MTESt:SSCReen:DDISk:BACKground {WHITe | NORMal}
```

Controls the background color of the graticule area of an HP PaintJet printer. It is valid only when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the disk.

Example

The following program selects the HP PaintJet as the printer format and sets the background color to white.

```
10  OUTPUT 707;":MTEST:SSCREEN:DDISK:PFORMAT PAINTJET"
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:BACKGROUND WHITE"
30  END
```

Query

```
:MTESt:SSCReen:DDISk:BACKground?
```

The query returns the current background for the disk store.

Returned Format

```
[:MTESt:SSCReen:DDISk:BACKground] {WHITe | NORMal} <NL>
```

Example

The following example gets the current background color setting and prints the setting on the controller display.

```
10  DIM Screen_back$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:BACKGROUND?"
30  ENTER 707;Screen_back$
40  PRINT Screen_back$
50  END
```

SSCReen:DDISk:MEDia

Command

```
:MTESt:SSCReen:DDISk:MEDia {PAPer | TRANsparency}
```

Specifies whether paper or transparency is to be used in the printer. This control is used when the store screen is directed to the disk. If TRANsparency is selected, the printer makes two passes over each data row, putting more ink on the page. This darkens the page for better transparency results. The command applies only to the HP PaintJet and color DeskJet printers, and will slow print speed.

Example

The following program sets the media type to TRANsparency.

```
10  OUTPUT 707;":MTEST:SSCREEN:DDISK:MEDIA TRANSPARENCY"
20  END
```

Query

```
:MTESt:SSCReen:DDISk:MEDia?
```

The query returns the current media for the disk store.

Returned Format

```
[:MTESt:SSCReen:DDISk:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following program gets the current media setting and prints the result on the controller screen.

```
10  DIM Screen_media$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:MEDIA?"
30  ENTER 707;Screen_media$
40  PRINT Screen_media$
50  END
```

SSCReen:DDISk:PFORmat

Command

```
:MTESt:SSCReen:DDISk:PFORmat {THINkjet | PAINtjet |
LASerjet | EPSon | GIF | TIFF | CTIFf | PCX | DJ500 | DJ540 |
DJ550 | DJ560 | BWPaintjet | BWDeskjet}
```

Selects the printer format to use when storing the screen to a disk. This includes all print formats supported by the instrument.

Example

The following example sets the output format for the save screen disk file to TIFF (Tag Image File Format).

```
10  OUTPUT 707;":MTEST:SSCREEN:DDISK:PFORMAT TIFF"
20  END
```

Query

```
:MTESt:SSCReen:DDISk:PFORmat?
```

The query returns the current printer format for the disk store.

Returned Format

```
[:MTESt:SSCReen:DDISk:PFORmat] {THINkjet | PAINtjet | LASerjet |
 EPSon | GIF | TIFF | CTIFf | PCX | DJ500 | DJ540 | DJ550 | DJ560 |
 BWPaintjet | BWDeskjet}<NL>
```

Example

The following program gets the current setting for the disk file output format and prints the result on the controller screen.

```
10  DIM Disk_format$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:PFORMAT?"
30  ENTER 707;Disk_format$
40  PRINT Disk_format$
50  END
```

SSCReen:DPRinter

The MTESt:SSCReen:DPRinter sub-subsystem commands are used to set up the printer when storing the Mask Test display screen to a printer. The printer setup consists of the following commands:

- ADDRess
- BACKground
- MEDia
- PFORmat
- PORT

SSCReen:DPRinter:ADDRess

Command

```
:MTESt:SSCReen:DPRinter:ADDRess <address_value>
```

Allows the user to select the HP-IB address for the printer. This address is used only if the port is HP-IB.

<address_value> Any HP-IB address, 0 to 30.

Example The following example sets the port to HP-IB for save screen printer output, and tells the instrument that the printer is at HP-IB address 8.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PORT HPIB"
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:ADDRESS 8"
30  END
```

Query

```
:MTESt:SSCReen:DPRinter:ADDRess?
```

The query returns the current address of the printer.

Returned Format

```
[:MTESt:SSCReen:DPRint:ADDRess] <address_value><NL>
```

Example The following example gets the HP-IB address of the printer from the instrument and prints the result on the controller screen.

```
10  DIM Screen_address$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:ADDRESS?"
30  ENTER 707;Screen_address$
40  PRINT Screen_address$
50  END
```

SSCReen:DPRinter:BACKground

Command

```
:MTESt:SSCReen:DPRinter:BACKground {WHITe | NORMal}
```

Controls the background color of the graticule area of an HP PaintJet printer. It is only valid when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the printer.

Example

The following program selects the HP PaintJet as the printer format and sets the background color to white.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PFORMAT PAINTJET"
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:BACKGROUND WHITE"
30  END
```

Query

```
:MTESt:SSCReen:DPRinter:BACKground?
```

The query returns the current background for the printer.

Returned Format

```
[:MTESt:SSCReen:DPRinter:BACKground] {WHITe | NORMal}<NL>
```

Example

The following example gets the current background color setting and prints the setting on the controller display.

```
10  DIM Screen_back$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:BACKGROUND?"
30  ENTER 707;Screen_back$
40  PRINT Screen_back$
50  END
```

SSCReen:DPRinter:MEDia

Command

```
:MTESt:SSCReen:DPRinter:MEDia {PAPer | TRANsparency}
```

Specifies either paper or transparency in the printer. If TRANsparency is selected, the printer makes two passes over each data row, putting more ink on the page. This darkens the page for better transparency results. The command applies only to the HP PaintJet and color DeskJet printers, and will slow print speed.

Example

The following program sets the media type to TRANsparency.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:MEDIA TRANSPARENCY"
20  END
```

Query

```
:MTESt:SSCReen:DPRinter:MEDia?
```

The query returns the current printer media.

Returned Format

```
[:MTESt:SSCReen:DPRinter:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following program gets the current media setting and prints the result on the controller screen.

```
10  DIM Screen_media$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:MEDIA?"
30  ENTER 707;Screen_media$
40  PRINT Screen_media$
50  END
```

SSCReen:DPRinter:PFORmat

Command

```
:MTESt:SSCReen:DPRinter:PFORmat {THINkjet | PAINtjet |
 LASerjet | EPSon | DJ500 | DJ540 | DJ550 | DJ560 | BWPaintjet |
 BWDeskjet}
```

Selects the printer format to use when storing the screen to a printer. This includes only the formats available from the setup print menu that refer to a physical printer.

Example

The following program sets the save screen printer format to the HP DeskJet 550.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PFORMAT DJ550"
20  END
```

Query

```
:MTESt:SSCReen:DPRinter:PFORmat?
```

The query returns the current printer format.

Returned Format

```
[:MTESt:SSCReen:DPRinter:PFORmat] {THINkjet | PAINtjet |
 LASerjet | EPSon | DJ500 | DJ540 | DJ550 | DJ560 | BWPaintjet |
 BWDeskjet}<NL>
```

Example

The following example gets the current save screen printer format setting and prints it on the controller display.

```
10  DIM Screen_pformat$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PFORMAT?"
30  ENTER 707;Screen_pformat$
40  PRINT Screen_pformat$
50  END
```

SSCReen:DPRinter:PORT

Command

```
:MTESt:SSCReen:DPRinter:PORT {CENTronics | HPIB}
```

Selects the printer port for the screen when store screen is going to the printer.

Example

The following example sets the Centronics port for printer output when the mask test saves a screen to the printer.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PORT CENTRONICS"
20  END
```

Query

```
:MTESt:SSCReen:DPRinter:PORT?
```

The query returns the current port type.

Returned Format

```
[:MTESt:SSCReen:DPRinter:PORT] {CENTronics | HPIB} <NL>
```

Example

The following example determines which port is selected for save screen printer output and prints the result on the controller screen.

```
10  DIM Screen_port$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PORT?"
30  ENTER 707;Screen_port$
40  PRINT Screen_port$
50  END
```

SSUMmary

Command

```
:MTESt:SSUMmary {OFF | PRINter | DISK [,<filename>]}
```

Saves the summary in the event of a failure.

A destination of OFF turns off the summary save. A destination of PRINter builds the summary file and sends it to the printer. In this case, a variety of printer-related controls are used to specify the printer configuration. The following printer commands are used when storing the summary to a printer:

- ADDRess
- MEDia
- PFORmat (printer format)
- PORT

When set to DISK, the summary is written to the disk drive using a user-specified base name with a machine-generated suffix. The results of tests 1, 2, and 3, for example, may be written to files SUMM0001.SUM SUMM0002.SUM, and SUMM0003.SUM, respectively. The summary is a logging method where the user can get an overall view of the test results. The summary is an ASCII file which the user can peruse on a computer, read into a spreadsheet, and so on.

<filename>

A file prefix of four characters, defaulted to MSUM.

Example

The following program disables the save summary feature.

```
10  OUTPUT 707;":MTEST:SSUMMARY OFF"
20  END
```

Query

```
:MTESt:SSUMmary?
```

The query returns the current specified destination for the summary.

Returned Format

```
[:MTESt:SSUMmary] {OFF | PRINter | DISK [,<filename>]}<NL>
```

Example

The following example gets the setting of the save summary destination from the instrument and prints the result on the controller display.

```
10  DIM Summary_dest$[30]
20  OUTPUT 707;":MTEST:SSUMMARY?"
30  ENTER 707;Summary_dest$
40  PRINT Summary_dest$
50  END
```

SSUMmary:ADDRess

Command

```
:MTESt:SSUMmary:ADDRess <address_value>
```

Selects the HP-IB address for the printer. This address is used only when the summary is going to the printer and the port is set to HP-IB.

<address_value>

Any HP-IB address from 0 to 30.

Example

The following program sets the printer output port to HP-IB at address 9.

```
10  OUTPUT 707;":MTEST:SSUMMARY:PORT HPIB"
20  OUTPUT 707;":MTEST:SSUMMARY:ADDRESS 9"
30  END
```

Query

```
:MTESt:SSUMmary:ADDRess?
```

The query returns the current address of the printer.

Returned Format

```
[:MTESt:SSUMmary:ADDRess]<address_value><NL>
```

Example

The following example gets the current setting of the HP-IB address and prints it on the controller display.

```
10  DIM Summary_addr$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:ADDRESS?"
30  ENTER 707;Summary_addr$
40  PRINT Summary_addr$
50  END
```

SSUMmary:MEDia

Command

```
:MTESt:SSUMmary:MEDia {PAPer | TRANsparency}
```

Specifies either paper or transparency in the printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the media look darker and slows down the printing process. This control is used when the store summary is directed to the printer, and is only valid for the HP PaintJet and color DeskJet printers.

Example

The following program sets the media type to paper for the store summary results.

```
10  OUTPUT 707;":MTEST:SSUMMARY:MEDIA PAPER"
20  END
```

Query

```
:MTESt:SSUMmary:MEDia?
```

The query returns the current specified media for the printer.

Returned Format

```
[:MTESt:SSUMmary:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following example gets the media type setting and prints the result on the controller display.

```
10  DIM Summary_media$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:MEDIA?"
30  ENTER 707;Summary_media$
40  PRINT 707;Summary_media$
50  END
```

SSUMmary:PFORmat

Command

```
:MTESt:SSUMmary:PFORmat {THINkjet | PAINtjet |
LASerjet | EPSon | DJ540 | DJ550 | DJ560 | BWPaintjet | BWDeskjet}
```

Selects the printer format to use when storing the summary to a printer. This includes the physical printers supported by the instrument, with the exception of the HP DeskJet 500C with the color cartridge installed. To set up the format for the HP DeskJet 500C, use the monochrome cartridge and the BWDeskjet parameter in the command.

Example

The following program sets the save screen printer format to Epson.

```
10  OUTPUT 707;":MTEST:SSUMMARY:PFORMAT EPSON"
20  END
```

Query

```
:MTESt:SSUMmary:PFORmat?
```

The query returns the current specified printer format.

Returned Format

```
[:MTESt:SSUMmary:PFORmat] {THINkjet | PAINtjet |
LASerjet | EPSon | DJ540 | DJ550 | DJ560 | BWPaintjet |
BWDeskjet}<NL>
```

Example

The following example gets the current printer format setting and prints it on the controller screen.

```
10  DIM Summary_format$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:PFORMAT?"
30  ENTER 707;Summary_format$
40  PRINT Summary_format$
50  END
```

SSUMmary:PORT

Command

```
:MTESt:SSUMmary:PORT {CENTronics | HPIB}
```

Selects the printer port for the summary when the summary is sent to the printer.

Example

The following program sets the printer port for summary output to HP-IB.

```
10  OUTPUT 707;":MTEST:SSUMMARY:PORT HPIB"
20  END
```

Query

```
:MTESt:SSUMmary:PORT?
```

The query returns the current specified port for the printer.

Returned Format

```
[:MTESt:SSUMmary:PORT] {CENTronics | HPIB}<NL>
```

Example

The following program gets the current setting of the summary output port and prints the setting on the controller display.

```
10  DIM Summary_port$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:PORT?"
30  ENTER 707;Summary_port$
40  PRINT Summary_port$
50  END
```

STANdard (HP 83480A Only)

Command

```
:MTESt:STANdard <standard>
```

Loads a predefined standard mask as the current mask.

<standard>

```
{ DS1 | DS1C |DS2 | DS3 | FC133 | FC266 | FC531 | FC1063 | FDDI |
PRImary | SECondary | TERtiary | OC1 | OC3  |OC12 | OC24 | OC48 |
QUAT0 | QUAT1 | STM0 | STM1 | STM4 | STM16 | STS1E | STS1P | STS3E |
STS3P0 | STS3P1 | PDH2 | PDH8 | PDH34 | PDH1390 | PDH 1391}
```

The standards PDH2, PDH8, PDH34, PDH1390, and PDH1391 are available in firmware revision A.02.00 and above.

Example

The following program sets the mask to OC48.

```
10  OUTPUT 707;":MTEST:STANDARD:OC48"
20  END
```

Query

```
:MTESt:STANdard?
```

The query returns the currently selected standard mask.

Returned Format

```
[:MTESt:STANdard] <standard><NL>
```

Example

The following program determines the current standard and prints the setting on the controller display.

```
10  DIM Standard$[30]
20  OUTPUT 707;":MTEST:STANdard?"
30  ENTER 707;Standard$
40  PRINT Standard$
50  END
```

SWAVeform

Command

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

Saves waveforms from a channel, function, histogram, or waveform memory in the event of a failure detected by the Mask 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.

<source>

```
{CHANnelN | FUNCtionN | FFT  | HISTogram | WMEMoryN | RESet}
```

<destination>

```
{OFF | WMEMoryN | DISK}
```

<filename>

A descriptive file prefix consisting of up to four characters. If no filename is specified the prefix defaults to CH1A . . . CH4B, FN1, FN2, FFT, HIST, MEM1 . . . MEM4.

<format>

```
{TEXT [,YVALues | VERBose] | INTernal}
```

Example

The following example saves the waveform from channel 1 to waveform memory 1 when a failure occurs.

```
10  OUTPUT 707;":MTEST:SWAVEFORM CHANNEL1,WMEMORY1"
20  END
```

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 gets the current save waveform configuration and prints it on the controller display.

```
10  DIM Save_wave$[200]
20  OUTPUT 707;":MTEST:SWAVEFORM? CHANNEL1"
30  ENTER 707;Save_wave$
40  PRINT Save_wave$
50  END
```

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.

TEST

Command

```
:MTESt:TEST {ON | 1 | OFF | 0}
```

Controls the execution of the Mask Test function. ON allows the Mask Test to run for all of the active sources. When the Mask Test is turned on, the Mask Test results are displayed on screen in a window below the graticule in the mask test window. OFF disables mask testing.

Example

The following example turns on the mask test function.

```
10  OUTPUT 707;":MTEST:TEST ON"
20  END
```

Query

```
:MTESt:TEST?
```

The query returns the state of the mask test subsystem, whether on or off.

Returned Format

```
[:MTESt:TEST] {1 | 0}<NL>
```

Example

The following example determines whether the mask test subsystem is on or off and prints the result on the controller screen.

```
10  DIM Mtest_state$[30]
20  OUTPUT 707;":MTEST:TEST?"
30  ENTER 707;Mtest_state$
40  PRINT Mtest_state$
50  END
```

17

Measure Commands

Measure Commands

The Measure subsystem commands are used to make parametric measurements on displayed waveforms.

The Measure subsystem contains the following commands and queries:

- APOWer (*HP 83480A Only*)
- CGRade
  - COMPlete
  - CROSsing
  - DCDistortion
  - EHEight
  - ERATio (*HP 83480A Only*)
  - ERCalibrate (*HP 83480A Only*)
  - ERFactor (*HP 83480A Only*)
  - EWIDth
  - JITTer
  - PEAK?
  - QFACtor (*HP 83480A Only*)
- CLEar
- DEFine
- DELTatime
- DUTYcycle
- FALLtime
- FFT
  - DFRequency (delta frequency)
  - DMAGnitude (delta magnitude)
  - FREQuency
  - MAGNitude
  - PEAK1
  - PEAK2
  - THReshold
- FREQuency

- HISTogram
  - HITS
  - MEAN
  - MEDian
  - M1S
  - M2S
  - M3S
  - OFFSet?
  - PEAK
  - PP
  - SCALe?
  - STDDev
- NWIDth
- OVERshoot
- PERiod
- PREShoot
- PWIDth
- RESults?
- RISetime
- SCRatch
- SENDvalid
- SOURce
- STATistics
- TEDGe
- TMAX
- TMIN
- TVOLt
- VAMPlitude
- VAVerage
- VBASe
- VLOWer
- VMAX
- VMIDdle
- VMIN
- VPP
- VRMS
- VTIMe
- VTOP
- VUPper

Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must be displayed on the instrument.

- 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.
- For eye measurements, the complete eye (two crossing points) must be displayed.

User-Defined Measurements

When user-defined measurements are made, the defined parameters must be set before actually sending the measurement command or query.

Measurement Error

If a measurement cannot be made because of the lack of data, because the source signal is not displayed, or some other reason, the following results are returned:

- 9.99999E+37 is returned as the measurement result.
- If SENDVALID is ON, the error code is also returned.

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 instrument first determines the top (100%) and base (0%) amplitudes of the waveform. From this information, the instrument determines the other important amplitude values (10%, 90%, and 50% amplitude values) for making measurements.

The 10% and 90% amplitude values are used in the rise time and fall time measurements when standard measurements are selected.

The 50% amplitude value is used for measuring frequency, period, pulse width, and duty cycle with standard measurements selected.

Measurements can also be made using user-defined parameters instead of the standard measurement values.

When the command form of a measurement is used, the instrument is placed in the continuous measurement mode. The measurement result will be displayed on the front panel. There may be up to four measurements running continuously. The SCRATCH command is used 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 instrument 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 instrument 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 VAVERAGE and VRMS which allow measurements on a single cycle. 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.

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 rescaling the vertical to prevent the signal from being clipped.

Measurement Considerations with the HP 83480A or 54750A opt.& 83480K

The HP 83480A provides several eye diagram measurements, such as extinction ratio and crossing percent. These measurements are only available when the color grade database has been activated through the :DISPlay:CGRade:ON command. Measurements are valid only when one channel is ON with the database activated. If two channels are ON and the color grade database is active, a measurement command will return an error.

Before conducting eye diagram measurements, the signal type should be specified through the :MEASure:DEFine:CGRade NRZ command. This tells the instrument that the signal under analysis is an NRZ eye diagram waveform.

Several of the eye diagram measurements depend upon the definition of the vertical histogram window. This window is defined by time markers positioned within the eye as a percent of the bit period. The window can be defined through the :MEASure:DEFine EWINdow, <ewind1pct>,<ewind2pct> command.

To avoid using erroneous data, clear the display after you change instrument setup conditions or DUT conditions and acquire new data before extracting measurement results.

APOWer (HP 83480A Only)

Command

```
:MEASure:APOWer <units> [,<source>]
```

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.

<units>

```
{WATT | DECibel}
```

<source>

```
{CHANnel<number>}
```

<number>

For channels, this value is dependant on the type of plug-in and its location in the instrument. It will work only on optical channels.

Example

The following example measures the average power of the last specified signal.

```
10 OUTPUT 707;"MEASURE:APOWER WATT"
20 END
```

Query

```
:MEASure:APOWer? <units> [<,source>]
```

The query returns the measured power of the specified source.

Returned Format

```
[:MEASure:APOWer]<value>[,<result_state>]<NL>
```

<value> The average power.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the current power of the specified signal in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:APOWER? WATT"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:COMPlete

Command

```
:MEASure:CGRade:COMPlete <comp_hits>
```

Sets the color grade measurement completion criterion.

Example

The following example sets the completion criterion to 10 hits.

```
10  OUTPUT 707;":MEASURE:CGRADE:COMPLETE 10"
20  END
```

Query

The query returns the current setting for color grade completion.

```
:MEASure:CGRade:COMPlete?
```

Returned Format

```
[:MEASure:CGRade:COMPlete]<comp_hits><NL>
```

<comp_hits>

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 complete.

A color grade measurement query will return 9.99999E+37 until the measurement is complete.

Example

The following example sets the color grade complete value, then starts a Vmax measurement with the color grade database as the source.

```
10  OUTPUT 707;":MEASURE:CGRADE:COMPLETE? 8"
20  OUTPUT 707;":DISPLAY:CGRADE ON"
30  OUTPUT 707;":MEASURE:VMAX CGRADE"
40  END
```

CGRade:CROSsing

Command

```
:MEASure:CGRade:CROSsing
```

Measures the crossing level percent of the current eye diagram on the color grade display.

Example

The following example measures the crossing level.

```
10 OUTPUT 707;"MEASURE:CGRade:CROSsing"
20 END
```

Query

```
:MEASure:CGRade:CROSsing?
```

The query returns the crossing level percent of the current eye diagram on the color grade display.

Returned Format

```
[:MEASure:CGRade:CROSsing]<value>[,<result_state>]<NL>
```

<value>

The crossing level.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current crossing level in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:CROSSING?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:DCDistortion

Command

```
:MEASure:CGRade:DCDistortion <format>
```

Measures the duty cycle distortion on the eye diagram of the current color grade display. The parameter specifies the format for reporting the measurement.

<format>

```
{TIME | PERCent}
```

Example

The following example measures the duty cycle distortion.

```
10 OUTPUT 707;"MEASURE:CGRADE:DCDistortion TIME"
20 END
```

Query

```
:MEASure:CGRade:DCDistortion? <format>
```

This query returns the duty cycle distortion of the color grade display.

Returned Format

```
[:MEASure:CGRade:DCDistortion]<value>[,<result_state>] <NL>
```

<value>

The duty cycle distortion.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current duty cycle distortion in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:DCDISTORTION? PERCENT"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:EHEight

Command

```
:MEASure:CGRade:EHEight <format>
```

Measures the eye height on the eye diagram of the current color grade display. The parameter specifies the format for reporting the measurement.

<format>

```
{TIME | PERCent}
```

Example

The following example measures the eye height.

```
10 OUTPUT 707;"MEASURE:CGRADE:EHEight TIME"
20 END
```

Query

```
:MEASure:CGRade:EHEight? <format>
```

The query returns the eye height of the color grade display.

Returned Format

```
[:MEASure:CGRade:EHEight]<value>[,<result_state>]<NL>
```

<value>

The eye height.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current eye height in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:EHEIGHT?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:ERATio (HP 83480A Only)

Command

```
:MEASure:CGRade:ERATio <format>
```

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. See the :MEASure:CGRade:ERCalibrate command.

<format>

```
{RATio | DECibel | PERCent}
```

Example

The following example measures the extinction ratio.

```
10 OUTPUT 707;"MEASURE:CGRADE:ERATIO RATIO"
20 END
```

Query

```
:MEASure:CGRade:ERATio? <format>
```

The query returns the extinction ratio of the color grade display.

Returned Format

```
[:MEASure:CGRade:ERATio]<value>[,<result_state>]<NL>
```

<value>

The extinction ratio.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current extinction ratio in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:ERATIO? RATIO"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:ERCalibrate (HP 83480A Only)

Command

```
:MEASure:CGRade:ERCalibrate
```

Calibrates the dark level or DC offset of the input channel for use in extinction ratio measurements. Extinction ratio cal only prompts when executed manually.

Example

Before initiating this command, the source must be turned off. The user will be prompted to turn off the laser. The following example calibrates the dark noise level.

```
10 OUTPUT 707;"MEASURE:CGRADE:ERCALIBRATE"
20 END
```

CGRade:ERFactor (HP 83480A Only)

Requires firmware revision A.03.00 and above.

Command

```
:MEASure:CGRade:ERFactor {CHANnel<number>} {ON | OFF}, <value>
```

Corrects for the frequency response of the measurement system. This is usually a function of the bit rate and the receiver frequency response.

<number> For channels: This value is dependent on the type of plug-in and its location in the instrument.

<value> % correction factor.

Example

The following example sets the extinction ratio correction factor to 1.5% and turns on correction.

```
10  OUTPUT 707;":MEASURE:CGRADE:ERFACTOR CHANNEL1, ON, 1.5
20  END
```

CGRade:EWIDth

Command

```
:MEASure:CGRade:EWIDth
```

Measures the eye width on the eye diagram of the current color grade display.

Example

The following example measures the eye width.

```
10 OUTPUT 707;"MEASURE:CGRADE:EWIDTH"
20 END
```

Query

```
:MEASure:CGRade:EWIDth?
```

The query returns the eye width of the color grade display.

Returned Format

```
[:MEASure:CGRade:EWIDth]<value>[,<result_state>]<NL>
```

<value>

The eye width.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current eye width in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:EWIDTH?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:JITTer

Command

```
:MEASure:CGRade:JITTer <format>
```

Measures the jitter at the eye diagram crossing point. The parameter specifies the format, peak-to-peak or RMS, in which the results are reported.

<format>

```
{PP | RMS}
```

Example

The following example measures the jitter.

```
10 OUTPUT 707;"MEASURE:CGRADE:JITTER RMS"
20 END
```

Query

```
:MEASure:CGRade:JITTer? <format>
```

The query returns the jitter of the color grade display.

Returned Format

```
[:MEASure:CGRade:JITTer]<value>[,<result_state>]<NL>
```

<value>

The jitter.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current jitter in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:JITTER? RMS"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:PEAK?

Query

```
:MEASure:CGRade:PEAK?
```

The query returns the maximum number of hits of the color grade display.

Returned Format

```
[:MEASure:CGRade:PEAK]<value>[,<result_state>]<NL>
```

<value> The number of hits.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the current number of hits in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:PEAK?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CGRade:QFACtor (HP 83480A Only)

Command

```
:MEASure:CGRade:QFACtor
```

Measures the Q factor.

Example

The following example measures the Q factor.

```
10 OUTPUT 707;"MEASURE:CGRADE:QFACTOR"
20 END
```

Query

```
:MEASure:CGRade:QFACtor?
```

The query returns the Q factor of the color grade display.

Returned Format

```
[:MEASure:CGRade:QFACTOR]<value>[,<result_state>]<NL>
```

<value>

The Q factor.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the Q factor in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:CGRADE:QFACTOR?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

CLEar

Command

```
:MEASure:CLEar
```

Clears the measurement results from the screen. It is identical to the :MEASure:SCRatch command.

Example

The following example clears the current measurement results from the screen.

```
10 OUTPUT 707;"MEASURE:CLEAR"
20 END
```

DEFine

Command

```
:MEASure:DEFine <meas_spec>
```

Sets up the definition for measurements by specifying the delta time, threshold, or top-base values. In addition, the color grade, measurement signal type, and the eye measurement window position can be set. Changing these values may affect other Measure commands. The following table identifies the relationships between user-defined values and other Measure commands.

<meas_spec>

```
{CGRade,DELTatime,EWINdow,THResholds,TOPBase}
```

CGRade and EWINdow are HP 83480A and HP 54750A (Option 83480K) only commands.

:MEASure:DEFine Interactions

MEASure commands	DELTatime	THResholds	TOPBase
RISEtime		•	•
FALLtime		•	•
PERiod		•	•
FREQuency		•	•
VMIN			
VMAX			
VPP			
VTOP			•
VBASe			•
VAMPlitude			•
PWIDth		•	
NWIDth		•	
OVERshoot		•	
DUTYcycle		•	
DELTatime	•	•	
VRMS			

Command

```
MEASure:DEFine CGRade, <signal_type>
```

<signal_type>

```
{PULSe | NRZ | AMI | CMI}
```

Command

```
:MEASure:DEFine DELTatime,<start edge_direction>,
<start edge_number>,<start edge_position>,
<stop edge_direction>,<stop edge_number>,
<stop edge_position>
```

<edge_direction> {RISing | FALLing | EITHer}

<edge_number> An integer from 1 to 65534.

<edge_position> {UPPer | MIDDle | LOWer}

Command

```
:MEASure:DEFine EWINDOW, <ewid1PCT>,<ewind2PCT>
```

<ewid1PCT>, <ewind2PCT> An integer, 0 to 100, specifying an eye window as a percentage of the bit period unit interval.

Command

```
:MEASure:DEFine THResholds,{{STANdard}| {PERCent,
<upper_pct>, <middle_pct>,<lower_pct>} | T1090 | T2080
{VOLTage,<upper_volts>,<middle_volts>,<lower_volts>|}}
```

<upper_pct>, <middle_pct>, <lower_pct> An integer, + 25 to 125, specifying a percentage form base voltage to top voltage.

<upper_volts>, <middle_volts>, <lower_volts> A real number specifying a threshold voltage.

Command

```
:MEASure:DEFine TOPBase,{{STANdard}|{<top_volts>,<base_volts>}}
```

<top_volts>, <base_volts> A real number specifying an amplitude.

Example The following example sets the parameters for a time measurement from the first positive edge at the upper threshold level to the second negative edge at the middle threshold.

```
10  OUTPUT 707;":MEASURE:DEFINE DELTATIME,RISING,1,UPPER,FALLING,2,MIDDLE"
20  END
```

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.

Source is specified either using MEASure:SOURce, or using the optional <source> parameter when the DELTatime measurement is started.

Query

```
:MEASure:DEFine? {CGRade | DELTatime | EWINdow | THResholds |
 TOPBase}
```

The query returns the current setup for the specified parameter.

Returned Format

```
[:MEASure:DEFine] CGRade,<signal_type>

[:MEASure:DEFine] DELTATIME <start edge_direction>,
<start edge_number>,<start edge_position>,
<stop edge_direction>,<stop edge_number>,
<stop edge_position>

[:MEASure:DEFine] EWINdow,<ewind1pct>,<ewind2pct>

[:MEASure:DEFine] THRESHOLDS {{STANDARD} |
{PERCENT,<upper_pct>,<middle_pct>,<lower_pct>} |
{VOLTAGE,<upper_volts>,<middle_volts>,
<lower_volts>}}

[:MEASure:DEFine] TOPBASE {{STANDARD} | {<top_volts>,<base_volts>}}
```

Example

The following example returns the current setup for the measurement thresholds to the string variable, Setup$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setup$[50]       !Dimension variable
20  OUTPUT 707;":MEASURE:DEFINE? THRESHOLDS"
30  ENTER 707;Setup$
40  PRINT Setup$
50  END
```

DELTatime

Command

```
:MEASure:DELTatime [<source>[,<source>]]
```

Measures the delta time between two edges. If one source is specified, the delta time from the leading edge of the specified source to the trailing edge of the specified source is measured. If two sources are specified, the delta time from the leading edge on the first source to the trailing edge on the second source is measured.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the MEASure:DELTatime command. The rest of the parameters for this command are specified with the MEASure:DEFine command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the delta time between channel 1 and channel 2.

```
10  OUTPUT 707;":MEASURE:DELTATIME CHANNEL1,CHANNEL2"
20  END
```

Query

```
:MEASure:DELTatime? [<source>[,<source>]]
```

The query returns the measured delta time value.

Returned Format

```
[:MEASure:DELTatime]<value>[,<result_state>]<NL>
```

<value>

Delta time from the first specified edge on one source to the next specified edge on another source.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the current value of delta time in the numeric variable, Value, then prints the contents of the variable to the controller's screen. This example assumes the source was set using MEASure:SOURce.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:DELTATIME?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

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 [<source>]
```

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.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the duty cycle of the last specified signal.

```
10  OUTPUT 707;":MEASURE:DUTYCYCLE"
20  END
```

Query

```
:MEASure:DUTYcycle? [<source>]
```

The query returns the measured duty cycle of the specified source.

Returned Format

```
[:MEASure:DUTYcycle]<value>[,<result_state>]<NL>
```

<value>

The ratio of the positive pulse width to the period.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current duty cycle of the specified signal in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:DUTYCYCLE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

FALLtime

Command

```
:MEASure:FALLtime [<source>]
```

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.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the fall time of the last specified signal.

```
10  OUTPUT 707;":MEASURE:FALLTIME"
20  END
```

Query

```
:MEASure:FALLtime? [<source>]
```

The query returns the fall time of the specified source.

Returned Format

```
[:MEASure:FALLtime]<value>[,<result_state>] <NL>
```

<value>

Time at lower threshold − time at upper threshold.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result.

Example The following example places the current value for fall time in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:FALLTIME?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

FFT

The MEASure:FFT commands control the FFT measurements accessible through the Measure subsystem. The FFT sub-subsystem consists of the following commands:

- DFRequency (delta frequency)
- DMAGnitude (delta magnitude)
- FREQuency
- MAGNitude
- PEAK1
- PEAK2
- THReshold

The following Measure commands also operate on FFT functions:

Measure Command	Measurement Performed
:TMAX	The frequency of the maximum value in the spectrum.
:TMIN	The frequency of the minimum value in the spectrum.
:VMAX	The maximum value in the spectrum.
:VMIN	The minimum value in the spectrum.
:VPP	The range of values in the spectrum.
:VTIM	The value at a specified frequency.

FFT:DFRequency

Command

```
:MEASure:FFT:DFRequency [<source>]
```

Enables the delta frequency measurement. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

<source>

```
{FUNCtion<number> | FFT | WMEMory<number>}
```

<number>

For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:DFRequency? [<source>]
```

The query returns the FFT delta frequency of the specified peaks.

Returned Format

```
[:MEASure:FFT:DFRequency]<delta_frequency><NL>
```

FFT:DMAGnitude

Command

```
:MEASure:FFT:DMAGnitude [<source>]
```

Enables the delta magnitude measurement. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

<source>

```
{FUNCtion<number> | FFT | WMEMory<number>}
```

<number>

For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:DMAGnitude? [<source>]
```

The query returns the delta magnitude of the specified peaks.

Returned Format

```
[:MEASure:FFT:DMAGnitude]<delta_magnitude><NL>
```

FFT:FREQuency

Command

```
:MEASure:FFT:FREQuency [<source>]
```

Enables the frequency measurement. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

<source>

```
{FUNCtion<number> | FFT | WMEMory<number>}
```

<number> For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:FREQuency? [<source>]
```

The query returns the frequency measurement.

Returned Format

```
[:MEASure:FFT:FREQuency]<frequency><NL>
```

FFT:MAGNitude

Command

```
:MEASure:FFT:MAGNitude [<source>]
```

Measures the magnitude of the FFT. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

<source>

```
{FUNCtion<number> | FFT | WMEMory<number>}
```

<number> For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:MAGNitude? [<source>]
```

The query returns the magnitude value of the FFT.

Returned Format

```
[:MEASure:FFT:FMAGNitude] <magnitude><NL>
```

FFT:PEAK1

Command

```
:MEASure:FFT:PEAK1 <1st_peak_number>[<source>]
```

Sets the peak number of the first peak for FFT measurements. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

<1st_peak_number> An integer specifying the number of the first peak.

<source>

```
{FUNCtion<number> | FFT | WMEMory<number>}
```

<number> For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:PEAK1? [<source>]
```

The query returns the peak number currently set as the first peak.

Returned Format

```
[:MEASure:FFT:PEAK1] <1st_peak_number><NL>
```

FFT:PEAK2

Command

```
:MEASure:FFT:PEAK2 <2nd_peak_number>[<source>]
```

Sets the peak number of the second peak for FFT measurements. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

<2nd_peak_number> An integer specifying the number of the second peak.

<source>

```
{FUNCtion<number> | FFT | WMEMory<number>}
```

<number> For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:PEAK2? [<source>]
```

The query returns the peak number currently set as the second peak.

Returned Format

```
[:MEASure:FFT:PEAK1] <2nd_peak_number><NL>
```

FFT:THReshold

Command

```
:MEASure:FFT:THReshold  <threshold_value>
```

Sets the peak search threshold value.

<threshold_value> An integer specifying the peak search threshold value.

Query

```
:MEASure:FFT:THReshold?
```

The query returns the peak search threshold value.

Returned Format

```
[:MEASure:FFT:THReshold]<threshold_value><NL>
```

FREQuency

Command

```
:MEASure:FREQuency [<source>]
```

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).

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> |
 CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the frequency of the last specified signal.

```
10  OUTPUT 707;":MEASURE:FREQUENCY"
20  END
```

Query

```
:MEASure:FREQuency? [<source>]
```

The query returns the measured frequency.

Returned Format

```
[:MEASure:FREQuency]<value>[,<result_state>]<NL>
```

<value>

The frequency value, in Hertz, of the first complete cycle on the screen using the mid-threshold levels of the waveform.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current frequency of the signal in the numeric variable, Freq, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:FREQUENCY?"
30  ENTER 707;Freq
40  PRINT Freq
50  END
```

HISTogram:HITS

Command

```
:MEASURE:HISTogram:HITS [<source>]
```

Measures the number of hits within the histogram. The source is specified with the MEASure:SOURce command or with the optional parameter following the HITS command. The HISTogram:HITS measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Example The following example measures the number of hits within the histogram stored in WMEMory1.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:HITS WMEMORY1"
20  END
```

Query

```
:MEASure:HISTogram:HITS? [<source>]
```

The query returns the number of hits within the histogram.

Returned Format

```
[:MEASure:HISTogram:HITS]<value>[,<result_state>]<NL>
```

<value> The number of hits in the histogram.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the number of hits within the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:HITS?"
30  ENTER 707;Histhits
40  PRINT Histhits
50  END
```

HISTogram:MEAN

Command

```
:MEASURE:HISTogram:MEAN [<source>]
```

Measures the mean of the histogram. The mean of the histogram is the average value of all the points in the histogram. The source is specified with the MEASure:SOURce command or with the optional parameter following the MEAN command. The HISTogram:MEAN measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number>

For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the mean of the histogram source.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:MEAN HISTOGRAM"
20  END
```

Query

```
:MEASure:HISTogram:MEAN? [<source>]
```

The query returns the mean of the histogram.

Returned Format

```
[:MEASure:HISTogram:MEAN]<value>[,<result_state>]<NL>
```

<value> The mean of the histogram.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example returns the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:MEAN?"
30  ENTER 707;Histmean
40  PRINT Histmean
50  END
```

HISTogram:MEDian

Command

```
:MEASURE:HISTogram:MEDian [<source>]
```

Measures 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 is specified with the MEASure:SOURce command or with the optional parameter following the MEDian command. The HISTogram:MEDian measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Example The following example measures the median of the histogram whose source has been defined with the MEASure:SOURce command.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:MEDIAN"
20  END
```

Query

```
:MEASure:HISTogram:MEDian? [<source>]
```

The query returns the median of the histogram.

Returned Format

```
[:MEASure:HISTogram:MEDian]<value>[,<result_state>]<NL>
```

<value> The median of the histogram.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example returns the median of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:MEDIAN?"
30  ENTER 707;Histmed
40  PRINT Histmed
50  END
```

HISTogram:M1S

Command

```
:MEASURE:HISTogram:M1S [<source>]
```

Measures the percentage of points that are within one standard deviation of the mean of the histogram. The source is specified with the MEASure:SOURce command or with the optional parameter following the M1S command. The HISTogram:M1S measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Example The following example measures the percentage of points that are within one standard deviation of the mean of the histogram of the data stored in waveform memory 3.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:M1S WMEMORY3"
20  END
```

Query

```
:MEASure:HISTogram:M1S? [<source>]
```

The query returns the percentage of points within one standard deviation of the mean of the histogram.

Returned Format

```
[:MEASure:HISTogram:M1S]<value>[,<result_state>]<NL>
```

<value> The percentage of points within one standard deviation of the mean of the histogram.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example returns the percentage of points within one standard deviation of the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:M1S?"
30  ENTER 707;Histm1s
40  PRINT Histm1s
50  END
```

HISTogram:M2S

Command

```
:MEASURE:HISTogram:M2S [<source>]
```

Measures the percentage of points that are within two standard deviations of the mean of the histogram. The sources is specified with the MEASure:SOURce command or with the optional parameter following the M2S command. The HISTogram:M2S measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Example The following example measures the percentage of points within two standard deviations of the mean of the histogram whose source is specified using the MEASure:SOURce command.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:M2S"
20  END
```

Query

```
:MEASure:HISTogram:M2S? [<source>]
```

The query returns the percentage of points within two standard deviations of the mean of the histogram.

Returned Format

```
[:MEASure:HISTogram:M2S]<value>[,<result_state>]<NL>
```

<value> The percent of points within two standard deviations of the mean of the histogram.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the percentage of points within two standard deviations of the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:M2S?"
30  ENTER 707;Histm2s
40  PRINT Histm2s
50  END
```

HISTogram:M3S

Command

```
:MEASURE:HISTogram:M3S [<source>]
```

Measures the percentage of points that are within three standard deviations of the mean of the histogram. The source is specified with the MEASure:SOURce command or with the optional parameter following the M3S command. The HISTogram:M3S measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number>

For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the percentage of points that are within three standard deviations of the mean of the histogram.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:M3S HISTOGRAM"
20  END
```

Query

```
:MEASure:HISTogram:M3S? [<source>]
```

The query returns the percentage of points within three standard deviations of the mean of the histogram.

Returned Format

```
[:MEASure:HISTogram:M3S]<value>[,<result_state>]<NL>
```

<value>

The percentage of points within three standard deviations of the mean of the histogram.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the percentage of points within three standard deviations of the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:M3S?"
30  ENTER 707;Histm3s
40  PRINT Histm3s
50  END
```

HISTogram:OFFSet?

Query

```
:MEASURE:HISTogram:OFFSet? [<source>]
```

The query returns the offset of the histogram in hits or decibels. The offset is in hits for linear-scaled histograms and in decibels for logarithmically-scaled histograms. The source is specified with the MEASure:SOURce command or with the optional parameter following the OFFSet query. The HISTogram:OFFSet measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number>

For waveform memories (WMEMory): 1, 2, 3, or 4.

Returned Format

```
[:MEASure:HISTogram:OFFSet]<value>[,<result_state>]<NL>
```

<value>

The offset of the histogram in hits.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the offset of the current linear-scaled histogram in hits and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:OFFSET?"
30  ENTER 707;Histoff
40  PRINT Histoff
50  END
```

HISTogram:PEAK

Command

```
:MEASURE:HISTogram:PEAK [<source>]
```

Measures the number of hits in the greatest peak of the histogram. The source is specified with the MEASure:SOURce command or with the optional parameter following the PEAK command. The HISTogram:PEAK measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number>

For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the number of hits in the greatest peak of the histogram stored in waveform memory 2.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:PEAK WMEMORY2"
20  END
```

Query

```
:MEASure:HISTogram:PEAK? [<source>]
```

The query returns the number of hits in the greatest peak of the histogram.

Returned Format

```
[:MEASure:HISTogram:PEAK]<value>[,<result_state>]<NL>
```

<value>

The number of hits in the histogram peak.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the number of hits in the greatest peak of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:PEAK?"
30  ENTER 707;Histpeak
40  PRINT Histpeak
50  END
```

HISTogram:PP

Command

```
:MEASURE:HISTogram:PP [<source>]
```

Measures 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 is specified with the MEASure:SOURce command or with the optional parameter following the PP command. The HISTogram:PP measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number>

For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the width of the histogram.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:PP HISTOGRAM"
20  END
```

Query

```
:MEASure:HISTogram:PP? [<source>]
```

The query returns the width of the histogram.

Returned Format

```
[:MEASure:HISTogram:PP]<value>[,<result_state>]<NL>
```

<value> The width of the histogram.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example returns the width of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:PP?"
30  ENTER 707;Histpp
40  PRINT Histpp
50  END
```

HISTogram:SCALe?

Query

```
:MEASURE:HISTogram:SCALe? [<source>]
```

The query returns the scale of the histogram in hits or decibels per division. The scale is in hits for linear-scaled histograms and in decibels per division for logarithmically-scaled histograms. The source is specified with the MEASure:SOURce command or with the optional parameter following the SCALe query. The HISTogram:SCALe measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number>

For waveform memories (WMEMory): 1, 2, 3, or 4.

Returned Format

```
[:MEASure:HISTogram:SCALe]<value>[,<result_state>]<NL>
```

<value>

The scale of the histogram in hits.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the scale of the histogram whose source is specified in MEASure:SOURce and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:SCALE?"
30  ENTER 707;Histscal
40  PRINT Histscal
50  END
```

HISTogram:STDDev

Command

```
:MEASURE:HISTogram:STDDev [<source>]
```

Measures the standard deviation of the histogram. The source is specified with the MEASure:SOURce command or with the optional parameter following the STDDev command. The HISTogram:STDDev measurement only applies to the histogram waveform or memories containing histograms.

<source>

```
{WMEMory<number> | HISTogram}
```

<source> only available in firmware revision A.03.00 and above.

<number>

For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the standard deviation of the histogram whose source is specified using the MEASure:SOURce command.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:STDDEV"
20  END
```

Query

```
:MEASure:HISTogram:STDDev? [<source>]
```

The query returns the standard deviation of the histogram.

Returned Format

```
[:MEASure:HISTogram:STDDev]<value>[,<result_state>]<NL>
```

<value>

The standard deviation of the histogram.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the standard deviation of the histogram whose source is specified using the MEASure:SOURce command, and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:STDDEV?"
30  ENTER 707;Histstdd
40  PRINT Histstdd
50  END
```

NWIDth

Command

```
:MEASure:NWIDth [<source>]
```

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.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the width of the first negative pulse on screen.

```
10  OUTPUT 707;":MEASURE:NWIDTH"
20  END
```

Query

```
:MEASure:NWIDth? [<source>]
```

The query returns the measured width of the first negative pulse of the specified source.

Returned Format

```
[:MEASure:NWIDth]<value>[,<result_state>]<NL>
```

<value>

The width of the first negative pulse on the screen using the mid-threshold levels of the waveform.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current width of the first negative pulse on screen in the numeric variable, Width, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:NWIDTH?"
30  ENTER 707;Width
40  PRINT Width
50  END
```

OVERshoot

Command

```
:MEASure:OVERshoot [<source>]
```

Measures the overshoot of the first edge on the screen. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the OVERSHOOT command.

The algorithm is:

If the first edge on screen is rising, then

overshoot = (Local V – V) / V

else,

overshoot = (V – Local V) / V.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the overshoot of the first edge on screen.

```
10  OUTPUT 707;":MEASURE:OVERSHOOT"
20  END
```

Query

```
:MEASure:OVERshoot? [<source>]
```

The query returns the measured overshoot of the specified source.

Returned Format

```
[:MEASure:OVERshoot]<value>[,<result_state>]<NL>
```

<value>

Ratio of overshoot to amplitude, in percent.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current value of overshoot in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:OVERSHOOT?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

PERiod

Command

```
:MEASure:PERiod [<source>]
```

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.

The algorithm is:

If the first edge on screen 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.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the period of the waveform.

```
10  OUTPUT 707;":MEASURE:PERIOD"
20  END
```

Query

```
:MEASure:PERiod? [<source>]
```

The query returns the measured period of the specified source.

Returned Format

```
[:MEASure:PERiod]<value>[,<result_state>]<NL>
```

<value>

Period of the first complete cycle on screen.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current period of the waveform in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:PERIOD?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

PREShoot

Command

```
:MEASure:PREShoot [<source>]
```

Measures the preshoot of the first edge on the screen. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the PRESHOOT command.

The algorithm is:

If the first edge on screen is rising, then

preshoot = (V – Local V) / V

else,

preshoot = (Local V – V) / V.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number>}
```

<number>

For channels: Value 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.

Example

The following example measures the preshoot of the signal on screen.

```
10  OUTPUT 707;":MEASURE:PRESHOOT"
20  END
```

Query

```
:MEASure:PREShoot? [<source>]
```

The query returns the measured preshoot of the specified source.

Returned Format

```
[:MEASure:PREShoot] <value>[,<result_state>]<NL>
```

<value>

Ratio of preshoot to amplitude, in percent.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current preshoot in the numeric variable, Preshoot, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:PRESHOOT?"
30  ENTER 707;Preshoot
40  PRINT Preshoot
50  END
```

PWIDth

Command

```
:MEASure:PWIDth [<source>]
```

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.

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.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the width of the first positive pulse on the screen.

```
10  OUTPUT 707;":MEASURE:PWIDTH"
20  END
```

Query

```
:MEASure:PWIDth? [<source>]
```

The query returns the measured width of the first positive pulse of the specified source.

Returned Format

```
[:MEASure:PWIDth]<value>[,<result_state>]<NL>
```

<value> Width of the first positive pulse on the screen in seconds.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the value of the width of the first positive pulse on the screen in the numeric variable, Width, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:PWIDTH?"
30  ENTER 707;Width
40  PRINT Width
50  END
```

RESults?

Query

```
:MEASure:RESults?
```

The query returns the results of the continuous measurements. The measurement results always include only the current results. If SENDVALID is ON, then the measurement results state is returned immediately following the measurement result. If statistics are computed, the measurement results include the current, minimum, maximum, mean, standard deviation, and statistical sample size of each measurement. If the limit test is on, then limit test fields of failures, total waveforms, and status are returned. The following tables show the relationship of the values returned and STATISTICS, SENDVALID, and LIMITTEST commands. If more than one measurement is running continuously, then the returned values will be duplicated for each continuous measurement from the first to last (top to bottom) of the display. There may be up to four continuous measurements at a time.

Results Values Returned

	Statistics OFF		Statistics ON	
	Sendvalid OFF	**Sendvalid ON**	**Sendvalid OFF**	**Sendvalid ON**
Limit test OFF	current result	current result	current result	current result
		result state	minimum	result state
			maximum	minimum
			mean	maximum
			standard deviation	mean
			sample size	standard deviation
				sample size
Limit test ON	current result	current result	current result	current result
		result state	minimum	result state
		minimum	maximum	minimum
		maximum	mean	maximum
		standard deviation	standard deviation	mean
		sample size	sample size	standard deviation
				sample size
				failures
				total
				status

Returned Format

```
[:MEASure:RESults] <result list><NL>
```

<result list>

List of the measurement results, as listed in Table 17-1, separated with commas.

Example

The following example places the current results of the measurements in the string variable, Result$, then prints the contents of the variable to the controller's screen.

```
10  DIM Result$[200]          !Dimension variable
20  OUTPUT 707;":MEASURE:RESULTS?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

Table 17-1. Result States

Result State Code	Result	Description
0	RESULT_CORRECT	Result correct. No problem found.
1	RESULT_QUESTIONABLE	Result questionable but could be measured.
2	RESULT_LESS_EQ	Result less than or equal to value returned.
3	RESULT_GTR_EQ	Result greater than or equal to value returned.
4	RESULT_INVALID	Result returned is invalid.
5	EDGE_NOT_FOUND	Result invalid. Required edge not found.
6	MAX_NOT_FOUND	Result invalid. Max not found.
7	MIN_NOT_FOUND	Result invalid. Min not found.
8	TIME_NOT_FOUND	Result invalid. Requested time not found.
9	VOLT_NOT_FOUND	Result invalid. Requested voltage not found.
10	TOP_EQUALS_BASE	Result invalid. Top and base are equal.
11	MEAS_ZONE_SMALL	Result invalid. Measurement zone too small.
12	LOWER_INVALID	Result invalid. Lower threshold not on waveform.
13	UPPER_INVALID	Result invalid. Upper threshold not on waveform.
14	UPPER_LOWER_INVALID	Result invalid. Upper and lower thresholds are too close.
15	TOP_INVALID	Result invalid. Top not on waveform.
16	BASE_INVALID	Result invalid. Base not on waveform.
17	INCOMPLETE	Result invalid. Completion criteria not reached.

Table 17-1. Result States (continued)

Result State Code	Result	Description
18	INVALID_SIGNAL	Result invalid. Measurement invalid for this type of signal.
19	SIGNAL_NOT_DISPLAYED	Result invalid. Signal is not displayed.
20	CLIPPED_HIGH	Result invalid. Waveform is clipped high.
21	CLIPPED_LOW	Result invalid. Waveform is clipped low.
22	CLIPPED_HIGH_LOW	Result invalid. Waveform is clipped high and low.
23	ALL_HOLES	Result invalid. Data contains all holes.
24	NO_DATA	Result invalid. No data on screen.
25	CURSOR_OFF_SCREEN	Result invalid. Cursor is not on screen.
26	MEASURE_CANCELLED	Result invalid. Measurement aborted.
27	MEASURE_TIMEOUT	Result invalid. Measurement timed-out.
28	NO_MEAS	Result invalid. No measurement to track.
29	PEAK_NOT_FOUND	Result invalid. FFT peak not found.
30	INVALID_EYE[1]	Result invalid. Eye pattern not found.
31	BAD_NRZ[1]	Result invalid. NRZ eye pattern not found.
32	BAD_ER_CAL[1]	Invalid extinction ratio calibration.
33	NOT_1_CHANNEL[1]	Color grade database has more than one source.
34	NO_REF_PLANE[2]	Result invalid. Reference plane not defined.
35	TDR_TDT_CAL_REQD[2]	Result invalid. A TDR/TDT normalization and reference plane calibration is required.

1 HP 83480A or HP 54750A with option 83480K only.

2 HP 54750A only.

RISetime

Command

```
:MEASure:RISetime [<source>]
```

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.

The source is specified with the MEASURE:SOURCE command or with the optional parameter following the RISETIME command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

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.

Example

The following example measures the rise time of the displayed signal.

```
10  OUTPUT 707;":MEASURE:RISETIME"
20  END
```

Query

```
:MEASure:RISetime? [<source>]
```

The query returns the rise time of the specified source.

Returned Format

```
[:MEASure:RISetime]<value>[,<result_state>]<NL>
```

<value>

Rise time in seconds.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to MEASure:RESults command, for a list of the result states.

Example The following example places the current value of rise time in the numeric variable, Rise, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:RISETIME?"
30  ENTER 707;Rise
40  PRINT Rise
50  END
```

SCRatch

Command

```
:MEASure:SCRatch
```

Clears the measurement results from the screen.

Example The following example clears the current measurement results from the screen.

```
10  OUTPUT 707;":MEASURE:SCRATCH"
20  END
```

SENDvalid

Command

```
:MEASure:SENDvalid {OFF | 0 | ON | 1}
```

Enables the result state code to be returned with the :MEASure:RESults? query.

Example

The following example turns the send valid function on.

```
10  OUTPUT 707;":MEASURE:SENDVALID ON"
20  END
```

Query

```
:MEASure:SENDvalid?
```

The query returns the state of the Sendvalid control.

Returned Format

```
[:MEASure:SENDvalid] {0 | 1}<NL>
```

Example

The following example places the current mode for sendvalid in the string variable, Mode$, then prints the contents of the variable to the controller's screen.

```
10  DIM Mode$[50]           !Dimension variable
20  OUTPUT 707;":MEASURE:SENDVALID?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

NOTE

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.

SOURce

Command

```
:MEASure:SOURce <source>[,<source>]
```

Selects the source for measurements. Two sources can be specified with this command. All measurements, except MEASURE: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.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | HISTogram |
FFT | CGRade}
```

<number>

For channels: the number represents an integer, 1 through 4, followed by an optional letter, A or B (chan1 = chan1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

For functions: 1 or 2.

For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example selects channel 1 as the source for measurements.

```
10  OUTPUT 707;":MEASURE:SOURCE CHANNEL1"
20  END
```

Query

```
:MEASure:SOURce?
```

The query returns the current source selection.

Returned Format

```
[:MEASure:SOURce] <source>[,<source>]<NL>
```

Example

The following example places the currently specified sources in the string variable, Source$, then prints the contents of the variable to the controller's screen.

```
10  DIM Source$[50]       !Dimension variable
20  OUTPUT 707;":MEASURE:SOURCE?"
30  ENTER 707;Source$
40  PRINT Source$
50  END
```

STATistics

Command

```
:MEASure:STATistics {OFF | 0 | ON | 1}
```

Turns the statistics measurements on and off. The statistics state also affects the MEASure:RESults query.

There are three modes for statistics on the front panel: off, mean/standard deviation, and min/max. When statistics is turned on over the bus, the last mode (mean/standard deviation or min/max) selected from the front panel is displayed on screen along with the current measurements.

Example

The following example turns the statistics function on.

```
10  OUTPUT 707;":MEASURE:STATISTICS ON"
20  END
```

Query

```
:MEASure:STATistics?
```

The query returns the current statistics mode.

Returned Format

```
[:MEASure:STATistics] {0 | 1}<NL>
```

Example

The following example places the current mode for statistics in the string variable, Mode$, then prints the contents of the variable to the controller's screen.

```
10  DIM Mode$[50]              !Dimension variable
20  OUTPUT 707;":MEASURE:STATISTICS?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

NOTE

Refer to the MEASure:RESults? query for information on the result returned and how they are affected by the STATistics command.

TEDGe

Command

```
:MEASure:TEDGe <meas_thres_txt>,<slope><occurrence>[,<source>]
```

Measures the time interval between the trigger event and the specified threshold level and transition. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the TEDGe command.

<meas_thres_txt>

```
{UPPer | MIDDle | LOWer | CROSsing}
```

CROSsing threshold is applicable only when the source is color grade.

<slope>

```
{- (minus) for falling | + (plus) or optional for rising}
```

<occurrence> Numeric value representing which edge of the occurrence.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number> For channels: Value 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.

Query

```
:MEASure:TEDGe? <meas_thres_txt>,<slope><occurrence> [,<source>]
```

The query returns the time interval between the trigger event and the specified threshold level and transition.

Returned Format

```
[:MEASure:TEDGe]<time>[,<result_state>]<NL>
```

<time> The time interval between the trigger event and the specified voltage level and transition.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following 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. The contents of the variable are then printed to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TEDGE? UPPER,+2"
30  ENTER 707;Time
40  PRINT Time
50  END
```

NOTE

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TMAX

Command

```
:MEASure:TMAX [<source>]
```

Measures the first time at which the maximum voltage of the source waveform occurred. When FFT is the specified source, the frequency at which the first maximum value occurred is measured. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the TMAX command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT | CGRade}
```

<number>

For channels: Value 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.

Query

```
:MEASure:TMAX? [<source>]
```

The query returns the time at which the first maximum voltage occurred.

Returned Format

```
[:MEASure:TMAX]<time>[,<result_state>]<NL>
```

<time> Time at which the first maximum voltage occurred.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example returns the time at which the first maximum voltage occurred to the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TMAX?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

NOTE

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TMIN

Command

```
:MEASure:TMIN [<source>]
```

Measures the time at which the first minimum voltage occurred. When FFT is the specified source, the frequency at which the first minimum value occurred is measured. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the TMIN command.

Query

```
:MEASure:TMIN? [<source>]
```

The query returns the time at which the first minimum voltage occurred.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT | CGRade}
```

<number>

For channels: Value 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.

Returned Format

```
[:MEASure:TMIN]<time>[,<result_state>]<NL>
```

<time>

Time at which the first minimum voltage occurred.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the time at which the first minimum voltage occurred to the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TMIN?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

NOTE

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TVOLt

Command

```
:MEASure:TVOLt <voltage>,<slope><occurrence>[,<source>]
```

Measures the time interval between the trigger event and the defined voltage level and transition. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the TVOLt command.

<voltage> Voltage level at which time will be measured.

<slope> The direction of the waveform change when the specified voltage is crossed, rising (+) or falling (−).

<occurrence> 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).

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number>}
```

<number> For channels: Value 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.

Query

```
:MEASure:TVOLt? <voltage>,<slope><occurrence>[,<source>]
```

The query returns the time interval between the trigger event and the specified voltage level and transition.

Returned Format

```
[:MEASure:TVOLt]<time>[,<result_state>]<NL>
```

<time> The time interval between the trigger event and the specified voltage level and transition.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

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. The contents of the variable are then printed to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TVOLT? -.250,+3"
30  ENTER 707;Time
40  PRINT Time
50  END
```

NOTE

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

VAMPlitude

Command

```
:MEASure:VAMPlitude [<source>]
```

Calculates the difference between the top and base voltage of the specified source. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VAMPlitude command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example calculates the difference between the top and base voltage of the specified source.

```
10  OUTPUT 707;":MEASURE:VAMPLITUDE"
20  END
```

Query

```
:MEASure:VAMPlitude? [<source>]
```

The query returns the calculated difference between the top and base voltage of the specified source.

Returned Format

```
[:MEASure:VAMPlitude]<value>[,<result_state>]<NL>
```

<value>

Calculated difference between the top and base voltage.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the current amplitude value in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VAMPLITUDE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VAVerage

Command

```
:MEASure:VAVerage  {CYCLe | DISPlay} [,<source>]
```

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.

CYCLe

The CYCLe parameter instructs the average measurement to measure the average voltage across the first period on the display.

DISPlay

The DISPlay parameter instructs the average measurement to measure all the data on the display.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example calculates the average voltage over the displayed waveform.

```
10  OUTPUT 707;":MEASURE:VAVERAGE DISPLAY"
20  END
```

Query

```
:MEASure:VAVerage? {CYCLe | DISPlay},[<source>]
```

The query returns the calculated average voltage of the specified source.

Returned Format

```
[:MEASure:VAVerage]<value>[,<result_state>]<NL>
```

<value> The calculated average voltage.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the current average voltage in the numeric variable, Average, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VAVERAGE? DISPLAY"
30  ENTER 707;Average
40  PRINT Average
50  END
```

VBASe

Command

```
:MEASure:VBASe [<source>]
```

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.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number> For channels: Value 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.

Example The following example measures the voltage at the base of the waveform.

```
10  OUTPUT 707;":MEASURE:VBASE"
20  END
```

Query

```
:MEASure:VBASe? [<source>]
```

The query returns the measured voltage value at the base of the specified source.

Returned Format

```
[:MEASure:VBASe]<value>[,<result_state>]<NL>
```

<value> Voltage at the base of the waveform.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example returns the current voltage at the base of the waveform to the numeric variable, Voltage, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VBASE?"
30  ENTER 707;Voltage
40  PRINT Voltage
50  END
```

VLOWer

Command

```
:MEASure:VLOWer [<source>]
```

Measures the voltage value at the lower threshold of the waveform. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VLOWer command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Query

```
:MEASure:VLOWer?
```

The query returns the measured lower threshold of the selected source.

Returned Format

```
[:MEASure:VLOWer]<value>[,<result_state>]<NL>
```

<value>

Voltage value at the lower threshold.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the measured voltage at the lower threshold of the waveform to the numeric variable, Vlower, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VLOW?"
30  ENTER 707;Vlower
40  PRINT Vlower
50  END
```

VMAX

Command

```
:MEASure:VMAX [<source>]
```

Measures the absolute maximum voltage present on the selected source waveform. When FFT is the specified source, the maximum value in the spectrum is measured. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VMAX command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the absolute maximum voltage on the waveform.

```
10  OUTPUT 707;":MEASURE:VMAX"
20  END
```

Query

```
:MEASure:VMAX? [<source>]
```

The query returns the measured absolute maximum voltage present on the selected source waveform.

Returned Format

```
[:MEASure:VMAX]<value>[,<result_state>]<NL>
```

<value>

Absolute maximum voltage present on the waveform.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the measured absolute maximum voltage on the waveform to the numeric variable, Maximum, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VMAX?"
30  ENTER 707;Maximum
40  PRINT Maximum
50  END
```

VMIDdle

Command

```
:MEASure:VMIDdle [<source>]
```

Measures the voltage level at the middle threshold of the waveform. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VMID command.

Query

```
:MEASure:VMIDdle? [<source>]
```

The query returns the voltage value at the middle threshold of the waveform.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Returned Format

```
[MEASure:VMIDdle]<value>[,<result_state>]<NL>
```

<value>

The middle voltage present on the waveform.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the measured middle voltage on the waveform to the numeric variable, Middle, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VMID?"
30  ENTER 707;Middle
40  PRINT Middle
50  END
```

VMIN

Command

```
:MEASure:VMIN [<source>]
```

Measures the absolute minimum voltage present on the selected source waveform. When FFT is the specified source, the minimum value in the spectrum is measured. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VMIN command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the absolute minimum voltage on the waveform.

```
10  OUTPUT 707;":MEASURE:VMIN"
20  END
```

Query

```
:MEASure:VMIN? [<source>]
```

The query returns the measured absolute minimum voltage present on the selected source waveform.

Returned Format

```
[:MEASure:VMIN]<value>[,<result_state>]<NL>
```

<value>

Absolute minimum voltage present on the waveform.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example returns the measured absolute minimum voltage on the waveform to the numeric variable, Minimum, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VMIN?"
30  ENTER 707;Minimum
40  PRINT Minimum
50  END
```

VPP

Command

```
:MEASure:VPP [<source>]
```

Measures the maximum and minimum voltages on the selected source, then calculates the peak-to-peak voltage as the difference between the two voltages. When FFT is the specified source, the range of values in the spectrum are measured. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VPP command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the peak-to-peak voltage of the previously selected source.

```
10  OUTPUT 707;":MEASURE:VPP"
20  END
```

Query

```
:MEASure:VPP? [<source>]
```

The query returns the peak-to-peak voltage of the specified source.

Returned Format

```
[:MEASure:VPP]<value>[,<result_state>]<NL>
```

<value> Peak-to-peak voltage of the selected source.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the current peak-to-peak voltage in the numeric variable, Voltage, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VPP?"
30  ENTER 707;Voltage
40  PRINT Voltage
50  END
```

VRMS

Command

```
:MEASure:VRMS {CYCLe | DISPlay}, {AC | DC} [,<source>]
```

Measures the RMS voltage of the selected waveform by subtracting the average value of the waveform from each data point on the display. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VRMS command.

CYCLe The CYCLE parameter instructs the RMS measurement to measure the RMS voltage across the first period of the display.

DISPlay The DISPLAY parameter instructs the RMS measurement to measure all the data on the display. Generally, RMS voltage is measured across one period, however, measuring multiple periods may be accomplished with the DISPLAY option. The DISPlay parameter is also useful when measuring noise.

AC The AC parameter is used to measure the RMS voltage subtracting out the DC component.

DC 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:

DCVRMS = ACVRMS + VAVG

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number>}
```

<number> For channels: Value 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.

Example The following example measures the RMS voltage of the previously selected waveform.

```
10  OUTPUT 707;":MEASURE:VRMS CYCLE,AC"
20  END
```

Query

```
:MEASure:VRMS? {CYCLe | DISplay},{AC | DC} [,<source>]
```

The query returns the RMS voltage of the specified source.

Returned Format

```
[:MEASure:VRMS]<value>[,<result_state>]<NL>
```

<value> RMS voltage of the selected waveform.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the current AC RMS voltage over one period of the waveform in the numeric variable, Voltage, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VRMS? CYCLE,AC"
30  ENTER 707;Voltage
40  PRINT Voltage
50  END
```

VTIMe

Command

```
:MEASure:VTIMe <time> [,<source>]
```

Measures the voltage at the specified time. The time is referenced to the trigger event and must be on screen. When FFT is the specified source, the value at the specified frequency is measured. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VTIMe command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT}
```

<number> For channels: Value 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.

<time> Displayed time from trigger in seconds, or frequency in Hertz for FFT.

Query

```
:MEASure:VTIMe? <time> [,<source>]
```

The query returns the measured voltage.

Returned Format

```
[:MEASure:VTIMe]<value>[,<result_state>]<NL>
```

<value> Voltage at the specified time. When the source is FFT, the returned value is the vertical value at the horizontal setting passed in the VTIMe <time> parameter. The time parameter, when FFT is the source, is in Hertz.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example The following example places the voltage at 500 ms in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VTIME? 500E-3"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VTOP

Command

```
:MEASure:VTOP [<source>]
```

Measures the statisical top of the selected source waveform. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VTOP command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the voltage at the top of the waveform.

```
10  OUTPUT 707;":MEASURE:VTOP"
20  END
```

Query

```
:MEASure:VTOP? [<source>]
```

The query returns the measured voltage at the top of the specified source.

Returned Format

```
[:MEASure:VTOP]<value>[,<result_state>]<NL>
```

<value>

Voltage at the top of the waveform.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. Refer to the MEASure:RESults command, for a list of the result states.

Example

The following example places the value of the voltage at the top of the waveform in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VTOP?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VUPper

Command

```
:MEASure:VUPper [<source>]
```

Measures the voltage value at the upper threshold of the waveform. The source is specified with the MEASURE:SOURCE command or with the optional parameter following the VUPper command.

<source>

```
{CHANnel<number> | FUNCtion<number> | WMEMory<number> | CGRade}
```

<number>

For channels: Value 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.

Example

The following example measures the voltage at the upper threshold of the waveform.

```
10  OUTPUT 707;":MEASURE:VUPper"
20  END
```

Query

```
:MEASure:VUPper? [<source>]
```

The query returns the measured upper threshold value of the selected source.

Returned Format

```
[:MEASure:VUPper]<value>[,<result_state>]<NL>
```

<value>

Voltage at the upper threshold.

<result_state>

If SENDVALID is ON, the result state is returned with the measurement result. See Table 17-1, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the value of the voltage at the upper threshold of the waveform in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VUPPER?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

18

Pixel Memory Commands

Pixel Memory Commands

The Pixel Memory subsystem commands control the functions associated with the pixel memory. This allows merging (add to memory), clearing and turning the pixel memory display on and off. The Pixel subsystem consists of the following commands and queries:

- ADD
- CLEar
- DISPlay
- ERASe
- MERGe
- RECall
- STORe

Refer to DISPlay:DATa for information on reading and writing pixel memory.

ADD

Command

```
:PMEMory1:ADD
```

Merges the current waveform(s) to the pixel memory. This is identical to the :PMEMory1:MERGe command.

CLEar

Command

```
:PMEMory1:CLEar
```

Clears the display memory. This is identical to the :PMEMory1:ERASe command.

DISPlay

Command

```
:PMEMory1:DISPlay {ON| 1 | OFF | 0}
```

Determines whether or not the pixel memory is viewable.

Query

```
:PMEMory1:DISPlay?
```

The query returns the state of the pixel memory.

Returned format:

```
[PMEMory1:DISPlay]  {1 | 0}
```

ERASe

Command

```
:PMEMory1:ERASe
```

Clears the pixel memory.

MERGe

Command

```
:PMEMory1:MERGe
```

Merges the current waveform(s) to the pixel memory.

RECall

Command

```
:PMEMory1:RECall
```

Recalls the current pixel memory to the screen.

STORe

Command

```
:PMEMory1:STORe
```

Stores the current waveform(s) to the pixel memory.

19

Service Commands

Service Commands

The Service subsystem commands control the service instrument functions.

The Service subsystem contains the following commands:

- COMMents
- DECLassify

COMMents

Command

```
:SERVice:COMMents[?]PLUGin <number><comments>
```

Sets the comments field for the plug-in. The comments field is used to describe options included in the plug-in or for user comments about the plug-in.

<number> For plug-in slot 1 through 4.

<comments> String unquoted.

Example The following example returns the comments field associated with the plug-in using the SERVice:COMMents command:

```
10  OUTPUT 707;"SERVICE:COMMENTS"
20  END
```

DECLassify

Command

```
:SERVice:DECLassify
```

Purges all instrument memory and resets the instrument to a default state.

Example The following example sets the instrument to a default state using the :SERVice:DECLassify command.

```
10 OUTPUT 707;":SERVice:DECLassify"
20 END
```

20

Timebase Commands

Timebase Commands

The Timebase subsystem commands control the horizontal (x axis) instrument functions. The Timebase subsystem contains the following commands and queries:

- BRATe (*HP 83480 Only*)
- DELay
- POSition
- RANGe
- REFerence
- SCALe
- UNITs (*HP 83480 Only*)
- VIEW
- WINDow
  - DELay
  - POSition
  - RANGe
  - SCALe
  - SOURce

BRATe (HP 83480 Only)

Command

```
:TIMebase:BRATe <bit_rate>
```

Sets the bit rate used when the time base units are bit period.

<bit_rate>

The bit rate (in bits-per-second).

Example

The following example sets the bit rate to 155.520 MHz.

```
10  OUTPUT 707;":TIMEBASE:BRATe 155.520E6"
20  END
```

Query

```
:TIMebase:BRATe?
```

The query returns the bit rate setting.

Returned Format

```
[:TIMebase:BRATe] <bit_rate><NL>
```

Example

The following example places the current bit rate in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:BRATe?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

DELay

Command

```
:TIMebase:DELay <delay_value>
```

Sets the time interval between the trigger event and the on-screen delay reference point. The delay reference point is set with the TIMEBASE:REFERENCE command.

> **NOTE**
>
> This command is the same as the TIMebase:POSition command and is provided for compatibility with previous instruments. The preferred command for compatibility with future instruments is TIMebase:POSition.

<delay_value>

Time in seconds (or bits for the HP 83480), from trigger to the on-screen delay reference point. The maximum value depends on the time/division setting.

Example

The following example sets the delay to 2 ms.

```
10  OUTPUT 707;":TIMEBASE:DELAY 2E-6"
20  END
```

Query

```
:TIMebase:DELay?
```

The query returns the current delay value.

Returned Format

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

RANGe

Command

```
:TIMebase:RANGe <full_scale_range>
```

Sets the full-scale horizontal time in seconds (or bits for the HP 83480). The range value is ten times the time per division value and covers the range of 1 ns (100 ps/div) to 10 seconds (1 s/div).

<full_scale_range> 1 ns to 100 seconds.

Example The following example sets the full-scale horizontal range to 10 ms.

```
10  OUTPUT 707;":TIMEBASE:RANGE 10E-6"
20  END
```

Query

```
:TIMebase:RANGe?
```

The query returns the current full-scale horizontal time.

Returned Format

```
[:TIMebase:RANGe] <full_scale_range><NL>
```

Example The following example places the current full-scale horizontal range value in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:RANGE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

<delay_value>

Time in seconds (or bits for the HP 83480A) from trigger to the on-screen delay reference point. The maximum value depends on the time/division setting.

Example

The following example sets the delay position to 2 ms.

```
10  OUTPUT 707;":TIMEBASE:POSITION 2E-6"
20  END
```

Query

```
:TIMebase:POSition?
```

The query returns the current delay value.

Returned Format

```
[:TIMebase:POSition] <delay_value><NL>
```

Example

The following example places the current delay value in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

SCALe

Command

```
:TIMebase:SCALe <time>
```

Sets the time base scale.

<time> Is the time value (in seconds per division).

Example The following example sets the scale to 10 ms.

```
10  OUTPUT 707;":TIMEBASE:SCALE 10E-6"
20  END
```

Query

```
:TIMebase:SCALe?
```

The query returns the current scale time setting.

Returned Format

```
[:TIMebase:SCALe] <time><NL>
```

Example The following example places the current scale value in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:RANGE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

UNITs (HP 83480 Only)

Command

```
:TIMebase:UNITs {TIME | BITS}
```

Sets the time base units.

Example

The following example sets the time base units to bits.

```
10  OUTPUT 707;":TIMEBASE:UNITs BITS"
20  END
```

Query

```
:TIMebase:UNITs?
```

The query returns the time base units.

Returned Format

```
[:TIMebase:UNITs] {TIME | BITS}
```

Example

The following example places the current bit rate in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:UNITs?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

VIEW

Command

```
:TIMebase:VIEW {MAIN | WINDow | INTensified}
```

Turns the windowed view on and off. If the windowed view is on, the main waveforms are not shown.

INTensified displays the main waveform with the windowed view intensified.

Example

The following example turns the windowed view on.

```
10  OUTPUT 707;":TIMEBASE:VIEW WINDOW"
20  END
```

Query

```
:TIMebase:VIEW?
```

The query returns the current state of the TIMebase:VIEW command.

Returned Format

```
[:TIMebase:VIEW] {MAIN | WINDow | INTensified}<NL>
```

Example

The following example places the current state of windows in the string variable, State$, then prints the contents of the variable to the controller's screen.

```
10  DIM State$[50]            !Dimension variable
20  OUTPUT 707;":TIMEBASE:VIEW?"
30  ENTER 707;State$
40  PRINT State$
50  END
```

WINDow:DELay

Command

```
:TIMebase:WINDow:DELay <delay_position>
```

Sets the position of the time-base window on the main sweep. The range for this command is determined by the main sweep range (or size) and the main sweep delay value. The value for this command must keep the time-base window within the main sweep range.

<delay_position> Time in seconds from trigger to on-screen delay reference point. Maximum depends on the main sweep range (or size) and the main sweep delay value.

Example

```
10  OUTPUT 707;":TIMEBASE:WINDOW:DELAY 50E-9"
20  END
```

Query

```
:TIMebase:WINDow:DELay?
```

The query returns the current position of the time base window.

Returned Format

```
[:TIMebase:WINDow:DELay]<delay_position><NL>
```

Example The following example places the current position of the time base window in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:WINDOW:DELAY?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

See Also The TIMEbase:WINDow:POSition command performs the same function as this command and is the preferred command for new programs.

WINDow:POSition

Command

```
:TIMebase:WINDow:POSition <delay position>
```

Sets the position of the time-base window on the main sweep. The range for this command is determined by the main sweep range (or size) and the main delay value. The value for this command must keep the time-base window on the main sweep range.

<delay_position> Time in seconds from trigger to onscreen delay reference point. Maximum depends on the main sweep range (or size) and the main sweep delay value.

Example

```
10  OUTPUT 707;":TIMEBASE:WINDOW:POSITION 50E-9"
20  END
```

Query

```
:TIMebase:WINDow:POSition?
```

The query returns the current position of the time-base window.

Returned Format

```
[:TIMebase:WINDow:POSition]<delay_position><NL>
```

Example The following example places the current position of the time-base window in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:WINDOW:POSITION?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

WINDow:RANGe

Command

```
:TIMebase:WINDow:RANGe <full_scale_range>
```

Sets the full-scale range of the time-base window. The range value is ten times the time per division of the window. The maximum range of the window is the current main range. The minimum window range is 10 ps (1 ps/div).

<full_scale_range> The full-scale range of the time base window.

Example The following example sets the full-scale range of the time base window to 100 ns.

```
10  OUTPUT 707;":TIMEBASE:WINDOW:RANGE 100E-9"
20  END
```

Query

```
:TIMebase:WINDow:RANGe?
```

The query returns the current full-scale range of the time base window.

Returned Format

```
[:TIMebase:WINDow:RANGe]<full_scale_range><NL>
```

Example The following example reads the current full-scale range of the time base window into the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:WINDOW:RANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

WINDow:SCALe

Command

```
:TIMebase:WINDow:SCALe <scale_value>
```

Sets the seconds-per-division scale of the time base window. The scale value is one tenth of the window's value. The minimum window scale is 1 ps/div.

<scale_value> The full-scale range of the time base window.

Example The following example sets the per-division scale of the time base window to 10 ns.

```
10  OUTPUT 707;":TIMEBASE:WINDOW:SCALE 10E-9"
20  END
```

Query

```
:TIMebase:WINDow:SCALe?
```

The query returns the current scale value.

Returned Format

```
[:TIMebase:WINDow:SCALe]<scale_value><NL>
```

Example The following example reads the current full-scale range of the time base window into the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:WINDOW:SCALE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

WINDow:SOURce

Command

```
:TIMebase:WINDow:SOURce ALL
```

Selects the source for the time base window. The only parameter available for this command is ALL, which selects all sources displayed on the main screen.

Example

The following example selects all of the sources displayed on the main screen for the sources for the current window.

```
10  OUTPUT 707;":TIMEBASE:WINDOW:SOURCE ALL"
20  END
```

Query

```
:TIMebase:WINDow:SOURce?
```

The query returns the current source for the window.

Returned Format

```
[:TIMebase:WINDow:SOURce] ALL<NL>
```

Example

The following example places the current setting for the window source in the string variable, Current$, then prints the contents of the variable to the controller's screen.

```
10  DIM Current$[50]          !Dimension variable
20  OUTPUT 707;":TIMEBASE:WINDOW:SOURCE?"
30  ENTER 707;Current$
40  PRINT Current$
50  END
```

21

Trigger Commands

Trigger Commands

Trigger Commands

The Trigger subsystem commands define the conditions for triggering. Many of the commands in the Trigger subsystem are used in more than one of the trigger modes. The command set has been defined to represent more closely what is in the front-panel trigger menus, at the expense of some compatibility with the command sets for previous instruments. For this reason, the instrument accepts some commands for compatibility with previous instruments. The alternative command that is accepted by the instrument is noted for each command.

The Trigger subsystem consists of the following commands and queries:

- HYSTeresis
- LEVel
- SLOPe
- SOURce
- SWEep

HYSTeresis

Command

```
:TRIGger:HYSTeresis {NORMal | HSENsitivity}
```

Specifies the trigger hysteresis. NORM (NORMal) is the typical hysteresis selection, HSEN (High SENsitivity) gives minimum hysteresis and highest bandwidth.

Query

```
:TRIGger:HYSTeresis?
```

The query returns the current hysteresis mode.

Returned Format

```
[TRIGger:HYSTeresis] {NORMal|HSENsitivity}<NL>
```

LEVel

Command

```
:TRIGger:LEVel TRIGgerN, <level>
```

TRIGgerN N, an integer 1 to 4, represents the slot where the external trigger resides.

<level> Trigger level limits are plug-in dependent.

Query

```
:TRIGger:LEVel? TRIGgerN
```

The query returns the trigger level for the selected source.

Returned Format

```
[:TRIGger:LEVel] TRIGgerN,<level>
```

SLOPe

Command

```
:TRIGger:SLOPe {POSitive | NEGative}
```

Specifies the slope of the edge on which to trigger in EDGE triggering mode. It sets the slope of the channel previously selected with the TRIGger:SOURce or TRIGger:EDGE:SOURce command. It is equivalent to the TRIGger:EDGE:SLOPe command and is included for compatibility with older instruments.

Query

```
:TRIGger:SLOPe?
```

The query returns the current slope for the currently selected trigger mode.

Returned Format

```
[:TRIGger:SLOPe] {POSitive | NEGative}
```

SOURce

Command

```
:TRIGger:SOURce TRIGgerN
```

Selects the trigger channel that produces the trigger.

TRIGgerN

N, an integer 1 to 4, represents the slot where the external trigger resides.

Query

```
:TRIGger:SOURce?
```

The query returns the current trigger source of the current mode.

Returned Format

```
[:TRIGger:SOURce] TRIGgerN
```

SWEep

Command

```
:TRIGger:SWEep {TRIGgered | FREerun}
```

Selects the sweep mode.

If the TRIGGERED mode is selected, and no trigger is present, the unit will not sweep, and the data acquired on the previous trigger will remain on-screen. If the FREERUN mode is selected, the instrument automatically causes data to be acquired from the internal clock.

Query

```
:TRIGger:SWEep?
```

The query returns the current sweep mode.

Returned Format

```
[:TRIGger:SWEep] {TRIGgered | FREerun}
```

22

TriggerN Commands

TriggerN Commands

The TriggerN subsystem commands and queries are specific to a channel plug-in with external triggering capabilities. The HP 83480A and HP 54750A plug-ins have external triggering capabilities. These commands affect only the TriggerN input. The N at the end of trigger is the slot, 1 through 4, in which the external trigger input resides.

The TriggerN subsystem consists of the following commands and queries:

- BWLimit

BWLimit

Command

```
:TRIGgerN:BWLimit {ON | 1 | OFF | 0}
```

Controls an internal lowpass filter in the HP 83480A and HP 54750A plug-ins. When ON, the bandwidth of the specified trigger channel is limited to approximately 100 MHz.

Example

The following example turns on the bandwidth limit filter for an HP 83480A plug-in in slot 1:

```
10  OUTPUT 707;":TRIGGER1:BWLIMIT ON"
20  END
```

Query

```
:TRIGgerN:BWLimit?
```

The query returns the current setting for the specified trigger input.

Returned Format

```
[:TRIGgerN:BWLimit] {1 | 0}
```

Example

The following example gets the setting of the bandwidth limit filter for an HP 83480A plug-in in slot 1, and prints it on the controller screen.

```
10  OUTPUT 707;":TRIGGER1:BWLIMIT?"
20  ENTER 707;Bwlimit
30  PRINT Bwlimit
40  END
```

23

Waveform Commands

Waveform Commands

The Waveform subsystem is used to transfer waveform data between a controller and the instrument. It contains commands to set up the waveform transfer and to send or receive waveform records to or from the instrument. The waveform record is contained in two portions: the preamble and the waveform data. The preamble contains the scaling and other values used to describe the data. The waveform data contains the actual data in the waveform. The preamble and waveform data must be read or sent with two separate commands: WAVeform:PREamble and WAVeform:DATA.

The Waveform subsystem contains the following commands and queries:

- BANDpass?
- BYTeorder
- COMPlete?
- COUNt?
- COUPling?
- DATA
- FORMat
- POINts?
- PREamble
- SOURce
- TYPE?
- VIEW
- XDISplay?
- XINCrement?
- XORigin?
- XRANge?
- XREFerence?
- XUNits?
- YDISplay?
- YINCrement?
- YORigin?
- YRANge?
- YREFerence?
- YUNits?

Data Acquisition

When the data is acquired using the Digitize command, the data is placed in the channel or function buffer of the specified source. After the Digitize command, the instrument is stopped. If the instrument is restarted over HP-IB or the front panel, the data acquired with the Digitize command is overwritten.

The preamble, elements of the preamble, or waveform data can be queried while the instrument is running, but the data will reflect only the current acquisition and subsequent queries will not reflect consistent data. For example, if the instrument is running and the X origin is queried, then the data is queried in a separate HP-IB command, 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, this mode of operation is not recommended. Instead, the Digitize command should be used to stop the instrument so that all subsequent queries will be consistent.

The data in the channel, function, and memory buffers is non-volatile. Therefore, if the instrument power is cycled, the acquired data may still be queried with the Waveform Query commands. However, it is not recommended.

Function data is volatile and must be read following a Digitize command or the data will be lost when the instrument is turned off.

Waveform Data and Preamble

The waveform record is actually contained in two portions: the waveform data and the preamble. The waveform data is the actual data acquired for each point in the specified source. The preamble contains the information for interpreting the waveform data, which includes 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 set in the preamble are determined when the DIGITIZE command is executed or when the front-panel STORE key is pressed. The preamble values are based on the settings of variables in the ACQUIRE subsystem, or they are based on the front-panel setup when the STORE key is pressed.

Although preamble values can be changed with a controller, the way the data is acquired cannot be changed. Changing the preamble values cannot change the type of data that was actually acquired, the number of points actually acquired, etc. Therefore, you must use extreme caution when changing any waveform preamble values to ensure the data is still useful. For example,

setting points in the preamble to a different value from the actual number of points in the waveform results in inaccurate data.

The waveform data and preamble must be read or sent with two separate commands: WAVeform:DATA and WAVeform:PREamble.

Data Conversion

Data sent from the instrument must be scaled for useful interpretation. The values used to interpret the data are the X and Y origins, the X and Y increments, and the X and Y references. These values can be read from the waveform preamble.

Y-axis Units = (data value – Yreference) × Yincrement + Yorigin

X-axis Units = (data value – Xreference) × Xincrement + Xorigin

Data Format for HP-IB Transfer

There are four types of data formats that can be selected with the WAVEform:FORMat command: ASCII, byte, word, and long. Refer to the FORMat command for more information on data format.

BANDpass?

Query

```
:WAVeform:BANDpass?
```

The query returns an estimation of the maximum and minimum bandwidth limits of the source signal. Bandwidth limits are computed as a function of the coupling and filter mode selected. Cutoff frequencies are derived from the acquisition path and software filtering.

Returned Format

```
[:WAVeform:BANDpass]<lower_cutoff>,<upper_cutoff><NL>
```

<lower_cutoff> Minimum frequency passed by the acquisition.

<upper_cutoff> Maximum frequency passed by the acquisition.

Example

The following example places the estimated maximum and minimum bandwidth limits of the source signal in the string variable, Bandpass$, then prints the contents of the variable to the controller's screen.

```
10  DIM Bandpass$[50]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:BANDPASS?"
30  ENTER 707;Bandpass$
40  PRINT Bandpass$
50  END
```

BYTeorder

Command

```
:WAVeform:BYTeorder {MSBFirst | LSBFirst}
```

Selects the order in which bytes are transferred to and from the instrument, over the HP-IB, for the 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 is MSBFirst.

Example

The following example sets up the instrument to send the most significant byte first during data transmission.

```
10  OUTPUT 707;":WAVEFORM:BYTEORDER MSBFIRST"
20  END
```

Query

```
:WAVeform:BYTeorder?
```

The query returns the current setting for the byte order.

Returned Format

```
[:WAVeform:BYTeorder] {MSBFirst | LSBFirst}<NL>
```

Example

The following example places the current setting for the byte order in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[10]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:BYTEORDER?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

NOTE

MSBFirst is for microprocessors, like Motorola, where the most significant byte resides at the lower address. LSBFirst is for microprocessors, like Intel, where the least significant byte resides at the lower address.

COMPlete?

Query

```
:WAVeform:COMPlete?
```

The query returns the percent of time buckets that are complete for the currently selected waveform.

For RAW waveform type, the percent complete is the percent of the number of time buckets that have data in them compared to the record length. This is usually 100 percent.

For the AVERAGE waveform type, the percent complete is the percent of the number of time buckets that have the number of specified hits compared to the record length. The hits are specified by the WAVEFORM:COUNT command.

For the VERSUS waveform type, the percent complete is the least complete of the x-axis and y-axis waveforms.

Returned Format

```
[:WAVeform:COMPlete] <criteria><NL>
```

<criteria>

0 to 100 percent, rounded down to the closest integer.

Example

The following example places the current completion criteria in the string variable, Criteria$, then prints the contents of the variable to the controller's screen.

```
10  DIM Criteria$[50]         !Dimension variable
20  OUTPUT 707;":WAVEFORM:COMPLETE?"
30  ENTER 707;Criteria$
40  PRINT Criteria$
50  END
```

COUNt?

Query

```
:WAVeform:COUNt?
```

The query 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 NORMAL and VERSUS waveform types, the count value returned is one, unless the data contains holes. If the data contains holes, zero is returned.

Returned Format

```
[:WAVeform:COUNt] <number><NL>
```

<number>

An integer. Values range from 1 to 4096 for NORMal types and from 1 to 32768 for VERSus type.

Example

The following example places the current value in the count field in the string variable, Count$, then prints the contents of the variable to the controller's screen.

```
10  DIM Count$[50]            !Dimension variable
20  OUTPUT 707;":WAVEFORM:COUNT?"
30  ENTER 707;Count$
40  PRINT Count$
50  END
```

COUPling?

Query

```
:WAVeform:COUPling?
```

The query returns the input coupling of the currently selected waveform.

Returned Format

```
[:WAVeform:COUPling?] DCFifty<NL>
```

Example

The following example places the current input coupling of the selected waveform in the string variable, Setting$, then prints the contents of the variable.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:COUPLING?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

See Also

The CHANnel<number>:INPut command sets the coupling for a particular channel.

DATA

Command

```
:WAVeform:DATA <block_data>[,<block_data>]
```

Transfers waveform data to the instrument over HP-IB 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 and the second array contains the data on the y-axis. The two arrays are transferred one at a time over HP-IB 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.

Database data is transferred as a two-dimensional array of unsigned word values from 0 to 63488. The two-dimensional array has dimensions of 256 rows by 451 columns and is row priority.

The full qlevel range of the A/D converter will be returned with the data query. The Y increment, Y origin, and Y reference values should be used to convert the qlevels to voltage values. The Y range and Y display values should be used to plot the voltage values. All of these reference values are available from the waveform preamble.

<block_data> Binary block data in the # format.

Example The following 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(*)
30  END
```

NOTE

is an HP BASIC image specifier that suppresses the automatic output of the EOL sequence following the last output item.

K is an HP BASIC image specifier that outputs a number or string in standard form with no leading or trailing blanks.

W is an HP BASIC image specifier that outputs 16-bit words with the most significant byte first.

Query

```
:WAVeform:DATA?
```

The query outputs waveform data to the controller over HP-IB that is stored in a waveform memory, function, or channel buffer previously specified with the WAVeform:SOURce command. The data that is returned is described by the waveform preamble.

Returned Format

```
[:WAVeform:DATA]<block_data>[,<block_data>]<NL>
```

Example

The following 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 "#,2A,8D";Header$,Length
60  Length = Length/2                        !Length in words
70  ALLOCATE INTEGER Wdata(1:Length)
80  ENTER 707 USING "#,W";Wdata(*)
90  ENTER 707 USING "-K,B";End$
100 END
```

NOTE

is an HP BASIC image specifier that terminates the statement when the last ENTER item is terminated. EOI and line feed are the item terminators.

2A is an HP BASIC image specifier that places the next two characters received in a string variable.

8D is an HP BASIC image specifier that places the next 8 characters in a numeric variable.

W is an HP 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 HP 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 HP 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 instrument 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. The 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 bytes to signed integers. In the 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 larger to a smaller size. For waveform transfer into the instrument, the maximum valid qlevel is 124. The minimum valid qlevel is −128.

WORD

WORD formatted data is transferred as signed 16-bit integers in two bytes. If WAVEFORM:BYTeorder is set to MSBFirst, then the most significant byte of each word is sent first. If the BYTeorder is LSBFirst, then the least significant byte of each word is sent first. In the word format, the value 31232 represents a hole level (a hole in the data acquisition). The value 32256 represents a clipped-high level. The value 31744 represents a clipped-low level. For waveform transfer into the instrument, the maximum valid qlevel is 30720. The minimum qlevel is −32736.

The word format is also used to transfer color grade database data to and from the instrument. Color grade data values will range from 0 through 63488 in an unsigned format.

LONG

LONG formatted data is transferred as signed 32-bit integers in four bytes. If WAVeform:BYTeorder is set to MSBFirst, then the most significant byte of each long word is sent first. If the BYTeorder is LSBFirst, then the least significant byte of each long word is sent first. In the long format, the value 2046820352 represents a hole level (a hole in the data acquisition). The value 2113929216 represents a clipped-high level. The value 2080374784 represents a clipped-low level. For waveform transfer into the instrument, the maximum valid qlevel is 2013265920. The minimum qlevel is 0. The long format is used to transfer histogram data to and from the instument.

Example

The following example selects the WORD format for waveform data transmission.

```
10  OUTPUT 707;":WAVEFORM:FORMAT WORD"
20  END
```

Query

```
:WAVeform:FORMat?
```

The query returns the current output format for transferring waveform data.

Returned Format

```
[:WAVeform:FORMat] {ASCii | BYTE | LONG | WORD}<NL>
```

Example

The following example places the current output format for data transmission in the string variable, Mode$, then prints the contents of the variable to the controller's screen.

```
10  DIM Mode$[50]        !Dimension variable
20  OUTPUT 707;":WAVEFORM:FORMAT?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

POINts?

Query

```
:WAVeform:POINts?
```

The query 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 4096.

The number of points for vertical histograms will not exceed 256.
The number of points for horizontal histograms will not exceed 451.
The number of points for database waveforms will always be 115,456.

Example

The following example places the current acquisition length in the numeric variable, Length, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:POINTS?"
30  ENTER 707;Length
40  PRINT Length
50  END
```

NOTE

When you are receiving numeric data into numeric variables, the headers should be turned off. Otherwise, the headers may cause misinterpretation of returned data.

See Also

The ACQuire:POINts command in the Acquire commands chapter.

PREamble

Command

```
:WAVeform:PREamble <preamble_block>
```

Sends a waveform preamble to the previously selected waveform memory in the instrument. 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 table, at the end of this command, lists the elements in the preamble.

The preamble can be used to translate raw data into time and voltage values.

<preamble_block>

```
<format NR1>, <type NR1>, <points NR1>,
<count NR1>, <X increment NR3>,
<X origin NR3>, <X reference NR1>,
<Y increment NR3>, <Yorigin NR3>,
<Y referenceNR1>, <coupling NR1>,
<X display range NR3>, <X display origin NR3>,
<Y display range NR3>, <Y display origin NR3>,
<date, string>, <time, string>,
<frame model #, string>, <plug-in model #, string>,
<acquisition mode NR1>, <completion NR1>,
<X units NR1>, <Y units NR1>,
<max bandwidth limit NR3>, <min bandwidth limit NR3>
```

<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.

<framemodel #> A string containing the model number and serial number of the frame in the format MODEL#:SERIAL#.

<plug-in model #> A string containing the model number and serial number of the plug-in in the format MODEL#:SERIAL#. These values are zero if no plug-in is used.

<format>
0 for ASCII format
1 for BYTE format
2 for WORD format
3 for LONG format

<type>
1 for RAW type
2 for AVERAGE type
3 for VHISTOGRAM (vertical histogram) type
4 for HHISTOGRAM (horizontal histogram) type
5 for VERSUS type
6 for DATABASE type

<acquisition mode> 2 for SEQUENTIAL mode (this value is always returned)

<coupling>
0 for AC coupling
1 for DC coupling
2 for DCFIFTY coupling
3 for LFREJECT coupling (this value is always returned)

<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 HITS unit
7 for PERCENT units
8 for WATTS units
9 for OHMS units
10 for PERCENT REFLECTION units
11 for GAIN units

Example

The following example demonstrates how to output a preamble block to a waveform memory.

```
10 OUTPUT 707;":WAVEFORM:SOURCE WMEMORY1"
20 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE:PREAMBLE "
30 END
```

NOTE

is an HP BASIC image specifier that suppresses the automatic output of the EOL sequence following the last output item.

K is an HP BASIC image specifier that outputs a number or string in standard form with no leading or trailing blanks.

In line 20 of the program example, a space is inserted between the word "PREAMBLE" and the close quotation mark. This space must be inside the quotation mark because in this format (#,K), the data is packed together. Failure to add the space produces a word that is not a proper command.

Query

```
:WAVeform:PREamble?
```

The query outputs a waveform preamble to the controller from the waveform source, which can be a waveform memory or channel buffer.

Returned Format

```
[:WAVeform:PREamble] <preamble_block><NL>
```

Example

The following 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
```

NOTE

—K is an HP 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.

Waveform Preamble Elements

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 instrument. (See WAVeform:FORMat)
Type	This value describes how the waveform was acquired. (See WAVeform:TYPE)
Points	The number of data points or data pairs contained in the waveform data. (See ACQuire:POINts.) The number of points for vertical histograms will not exceed 256. The number of points for horizontal histograms will not exceed 451. The number of points for database waveforms will always be 115,456.
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 RAW and VERSUS waveform types, this value is 0 or 1. The count value is ignored when it is sent to the instrument 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 instrument, 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 instrument, this value is always zero. (See WAVeform:YREFerence)
Coupling	The input coupling of the waveform. The coupling value is ignored when sent to the instrument in the preamble. (See WAVeform:COUPling)

Waveform Preamble Elements (continued)

Element	Description
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	The y-display origin is the y-axis value at the center of the display. For voltage signals, it is the voltage at the center of the display. (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 instrument in the preamble.
Plug-in Model #	The model number of the plug-in that acquired this waveform. If no plug-in was used, then the value in this field is zero. The plug-in model number is ignored when it is sent to the instrument 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 instrument 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)
Band Pass	The band pass is two values that are an estimation of the maximum and minimum bandwidth limits of the source signal. Bandwidth limits are computed as a function of the coupling and filter mode selected. (See WAVeform:BANDpass)

SOURce

Command

```
:WAVeform:SOURce {WMEMory<number> | FUNCtion<number>
| CHANnel<number> | DATAbase | HISTogram | FFT}
```

Selects a channel, function, waveform memory, database, histogram, or FFT as the waveform source.

<number>

For channels: the number represents an integer, 1 through 4.

For waveform memories: 1, 2, 3, or 4.

For functions: 1 or 2.

Example

The following example selects channel 1 as the waveform source.

```
10  OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"
20  END
```

Query

```
:WAVeform:SOURce?
```

The query returns the currently selected waveform source.

Returned Format

```
[:WAVeform:SOURce] {WMEMory<number> | FUNCtion<number>
| CHANnel<number> | DATAbase | HISTogram | FFT}<NL>
```

Example

The following example places the current selection for the waveform source in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]        !Dimension variable
20  OUTPUT 707;":WAVEFORM:SOURCE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

TYPE?

Query

```
:WAVeform:TYPE?
```

The query 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, AVERAGE, DATABASE, HHISTOGRAM, VHISTOGRAM, or VERSUS.

RAW

Raw data consists of the last data point in each time bucket.

AVERAGE

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.

DATABASE

The database is transferred using unsigned values in the word format. The database is transferred as a block of data representing a two-dimensional array of 256 rows by 451 columns. The database may be generated using the DISPlay:CGRade command.

HHISTOGRAM

The data is a horizontal histogram. Histograms are transferred using the LONG format. They can be generated using the Histogram subsystem commands.

VHISTOGRAM

The data is a vertical histogram. Histograms are transferred using the LONG format. They can be generated using the Histogram subsystem commands.

VERSUS

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 | AVERage | DATAbase |
HHIStogram | VHIStogram | VERSus}<NL>
```

Example

The following example places the current acquisition data type in the string variable, Type$, then prints the contents of the variable to the controller's screen.

```
10  DIM Type$[50]        !Dimension variable
20  OUTPUT 707;":WAVEFORM:TYPE?"
30  ENTER 707;Type$
40  PRINT Type$
50  END
```

See Also

The acquisition data type is set with the ACQuire:AVERAGE command.

VIEW

Command

```
:WAVeform:VIEW {ALL | MAIN | WINDow}
```

Selects which view of the waveform is selected for data and preamble queries. The command can be set to ALL, MAIN, or WINDOW. The view has different meanings depending upon the waveform source selected.

Channels

For channels, ALL, MAIN, or WINDOW views may be selected. If ALL is specified, all of the data in the waveform record is referenced. If MAIN is selected, only the data in the main time base range is referenced. The first value corresponds to the first time bucket in the main time base range and the last value corresponds to the last time bucket in the main time base range. If WINDOW is selected, only the data in the window time base range is referenced. The first value corresponds to the first time bucket in the window time base range and the last value corresponds to the last time bucket in the window time base range.

Memories

For memories, if ALL is specified, all the data in the waveform record is referenced. WINDOW and MAIN refer to the data contained in the memory time base range for the particular memory. The first value corresponds to the first time bucket in the memory time base range and the last value corresponds to the last time bucket in the memory time base range.

Functions

For functions, ALL, MAIN, and WINDOW refer to all of the data in the waveform record.

The default setting for this command is ALL. The following table summarizes the parameters for this command for each source.

Waveform View Parameters

Source/Parameter	ALL	MAIN	WINDOW
CHANNEL	All data	Main time base	Window time base
MEMORY	All data	Memory time base	Memory time base
FUNCTION	All data	All data	All data

Example

The following example sets up the instrument to view all of the data.

```
10  OUTPUT 707;":WAVEFORM:VIEW ALL"
20  END
```

Query

```
:WAVeform:VIEW?
```

The query returns the view of the data currently selected.

Returned Format

```
[:WAVeform:VIEW] {ALL | MAIN | WINDow}<NL>
```

Example

The following example returns the current view setting to the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:VIEW?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

XDISplay?

Query

```
:WAVeform:XDISplay?
```

The query 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.

Returned Format

```
[:WAVeform:XDISplay] <value><NL>
```

<value>

A real number representing the x-axis value at the left edge of the display.

Example

The following example returns the x-axis value at the left edge of the display to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XDISPLAY?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

XINCrement?

Query

```
:WAVeform:XINCrement?
```

The query 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

The following example places the current Xincrement value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XINCREMENT?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Xincrement value can also be obtained through the WAVeform:PREamble? query.

XORigin?

Query

```
:WAVeform:XORigin?
```

The query returns 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. 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>

A real number representing the x-axis value of the first data point in the data record.

Example

The following example places the current Xorigin value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XORIGIN?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Xorigin value can also be obtained through the WAVeform:PREamble query.

XRANge?

Query

```
:WAVeform:XRANge?
```

The query 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.

Returned Format

```
[:WAVeform:XRANge] <value><NL>
```

<value>

A real number representing the x-axis duration of the displayed waveform.

Example

The following example returns the x-axis duration of the displayed waveform to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XRANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

XREFerence?

Query

```
:WAVeform:XREFerence?
```

The query returns the data point or level associated with the Xorigin data value. It is at this data point or level that the X origin is defined. In this instrument, the value is always zero.

Returned Format

```
[:WAVeform:XREFerence] 0<NL>
```

Example

The following example places the current X Reference value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XREFERENCE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Xreference value can also be obtained through the WAVeform:PREamble? query.

XUNits?

Query

```
:WAVeform:XUNits?
```

The query returns the x-axis units of the currently selected waveform source. The currently selected source may be a channel, function, database, histogram, or waveform memory.

Returned Format

```
[:WAVeform:XUNits] {UNKNown | VOLT | SECond | OHM | REFLect |
GAIN | CONStant | AMP | DECibels | HITS}<NL>
```

Example

The following example returns the x-axis units of the currently selected waveform source to the string variable, Unit$, then prints the contents of the variable to the controller's screen.

```
10  DIM Unit$[50]        !Dimension variable
20  OUTPUT 707;":WAVEFORM:XUNITS?"
30  ENTER 707;Unit$
40  PRINT Unit$
50  END
```

YDISplay?

Query

```
:WAVeform:YDISplay?
```

The query returns the y-axis value at the center of the display. For voltage signals, it is the voltage at the center of the display.

Returned Format

```
[:WAVeform:YDISplay] <value><NL>
```

<value>

A real number representing the y-axis value at the center of the display.

Example

The following example returns the current Y Display value to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;"":WAVEFORM:YDISPLAY?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

YINCrement?

Query

```
:WAVeform:YINCrement?
```

The query returns the duration between the y-axis levels.

For BYTE and WORD data, and for voltage waveforms, it is the voltage corresponding to one level.

For ASCII data format, the YINCrement is the full voltage range covered by the A/D converter.

Returned Format

```
[:WAVeform:YINCrement] <real_value><NL>
```

<real_value>

A real number in exponential (NR3) format.

Example

The following example places the current Yincrement value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:YINCREMENT?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Yincrement value can also be obtained through the WAVeform:PREamble? query.

YORigin?

Query

```
:WAVeform:YORigin?
```

The query 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

The following example places the current Y origin value in the numeric variable, Center, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:YORIGIN?"
30  ENTER 707;Center
40  PRINT Center
50  END
```

See Also

The YORIGIN value can also be obtained through the WAVeform:PREamble? query.

YRANge?

Query

```
:WAVeform:YRANge?
```

The query returns the y-axis duration of the displayed waveform. For voltage signals, it is the amount of voltage across the display.

Returned Format

```
[:WAVeform:YRANge] <value><NL>
```

<value>

A real number representing the y-axis duration of the displayed waveform.

Example

The following example returns the current Y Range value to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:YRANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

YREFerence?

Query

```
:WAVeform:YREFerence?
```

The query returns the level associated with the Y origin. It is at this level that the Y origin is defined. In this instrument, the value is always zero.

Returned Format

```
[:WAVeform:YREFerence]<integer_value><NL>
```

<integer_value>

Always 0.

Example

The following example places the current Y Reference value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:YREFERENCE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Yreference value can also be obtained through the WAVeform:PREamble? query.

YUNits?

Query

```
:WAVeform:YUNits?
```

The query returns the y-axis units of the currently selected waveform source. The currently selected source may be a channel, function, database, histogram, or waveform memory.

Returned Format

```
[:WAVeform:YUNits] {UNKNown | VOLT | SECond | OHM | REFLect
| GAIN | CONStant | AMP | DECibels | HITS | WATT}<NL>
```

Example

The following example returns the y-axis units of the currently selected waveform source to the string variable, Unit$, then prints the contents of the variable to the controller's screen.

```
10  DIM Unit$[50]        !Dimension variable
20  OUTPUT 707;":WAVEFORM:YUNITS?"
30  ENTER 707;Unit$
40  PRINT Unit$
50  END
```

24

Waveform Memory Commands

Waveform Memory Commands

The Waveform Memory subsystem commands control the waveform save functions. They allow saving and displaying of waveforms, memories, and functions.

The Waveform Memory subsystem consists of the following commands and queries:

- DISPlay
- SAVE
- XOFFset
- XRANge
- YOFFset
- YRANge

DISPlay

Command

```
:WMEMoryN:DISPlay {ON | 1  | OFF | 0}
```

Enables or disables the viewing of the selected memory.

Query

```
:WMEMoryN:DISPlay?
```

Returns the state of the selected memory.

Returned Format

```
[:WMEMoryN:DISPlay] {1 | 0} <NL>
```

SAVE

Command

```
:WMEMoryN:SAVE {CHANnelN | WMEMoryN | FUNCtionN | HISTogram | FFT}
```

Stores the specific channel, waveform, memory, function, histogram, or FFT waveform to the waveform memory.

XOFFset

Command

```
:WMEMoryN:XOFFset <offset>
```

Sets the x-axis (horizontal) offset for the selected waveform memory's display scale. Offset is referenced to center screen.

<offset> Number that sets the offset value.

Query

```
:WMEMoryN:XOFFset?
```

Returns the current x-axis (horizontal) offset for the selected waveform memory.

Returned Format

```
[:WMEMoryN:XOFFset] <offset><NL>
```

XRANge

Command

```
:WMEMoryN:XRANge <range>
```

Sets the x-axis (horizontal) range for the selected waveform memory's display scale.

<range> Value that sets the range.

Query

```
WMEMoryN:XRANge?
```

Returns the current x-axis (horizontal) range (scale) for the selected waveform memory.

Returned Format

```
[:WMEMoryN:XRANge] <range><NL>
```

YOFFset

Command

```
:WMEMoryN:YOFFset <offset>
```

Sets the x-axis (horizontal) offset for the selected waveform memory's display scale. Offset is referenced to center screen.

<offset> Number that sets the offset value.

Query

```
:WMEMoryN:YOFFset?
```

Returns the current x-axis (horizontal) offset for the selected waveform memory.

Returned Format

```
[:WMEMoryN:YOFFset]<offset><NL>
```

YRANge

Command

```
:WMEMoryN:YRANge <range>
```

Sets the x-axis (horizontal) range for the selected waveform memory's display scale.

<range> Value that sets the range.

Query

```
:WMEMoryN:YRANge?
```

Returns the current x-axis (horizontal) range (scale) for the selected waveform memory.

Returned Format

```
[:WMEMoryN:YRANge]<range><NL>
```

Index

Index