Programmer's Guide

Publication number 54600-97032
April 2001

This guide contains programming information for the following Agilent oscilloscope models:

54600
54601
54602
54603
54610
54615
54616

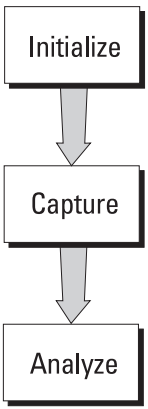
For Safety information, Warranties, and Regulatory information, see the pages behind the Index.

Agilent 54600-Series Oscilloscopes

Programming the Oscilloscope

When you attach an interface module to the rear of the Agilent 54600-Series Oscilloscopes, the oscilloscope becomes programmable. That is, you can hook a controller (such as a PC or workstation) to the oscilloscope, and write programs on that controller to automate oscilloscope setup and data capture. Both GPIB (also known as IEEE-488) and RS-232-C interfaces are available.

The following figure shows the basic structure of every program you will write for the oscilloscope.



Initialize

To ensure consistent, repeatable performance, you need to start the program, controller, and oscilloscope in a known state. Without correct initialization, your program may run correctly in one instance and not in another. This might be due to changes made in configuration by previous program runs or from the front panel of the oscilloscope.

- Program initialization defines and initializes variables, allocates memory, or tests system configuration.
- Controller initialization ensures that the interface to the oscilloscope (either GPIB or RS-232) is properly setup and ready for data transfer.
- Oscilloscope initialization sets the channel, trigger, timebase, and acquisition subsystems for the desired measurement.

Capture

Once you initialize the oscilloscope, you can begin capturing data for measurement. Remember that while the oscilloscope is responding to commands from the controller, it is not performing acquisitions. Also, when you change the oscilloscope configuration, any data already captured is most likely invalid.

To collect data, you use the DIGITIZE command. This command clears the waveform buffers and starts the acquisition process. Acquisition continues until the criteria, such as number of averages, completion criteria, and number of points is satisfied. Once the criteria is satisfied, the acquisition process is stopped. The acquired data is displayed by the oscilloscope, and the captured data can be measured, stored in memory in the oscilloscope, or transferred to the controller for further analysis. Any additional commands sent while DIGITIZE is working are buffered until DIGITIZE is complete.

You could also start the oscilloscope running, then use a wait loop in your program to ensure that the oscilloscope has completed at least one acquisition before you make a measurement. This is not recommended, because the needed length of the wait loop may vary, causing your program to fail. DIGITIZE, on the other hand, ensures that data capture is complete. Also, DIGITIZE, when complete, stops the acquisition process, so that all measurements are on displayed data, not a constantly changing data set.

Analyze

After the oscilloscope has completed an acquisition, you can find out more about the data, either by using the oscilloscope measurements or by transferring the data to the controller for manipulation by your program. Built-in measurements include IEEE standard parametric measurements (such as Vpp, frequency, pulse width) or the positioning and reading of voltage and time markers.

Using the WAVEFORM commands, you can transfer the data to your controller for special analysis, if desired.

The *Agilent 54600-Series Oscilloscopes Programmer's Guide* is your introduction to programming the Agilent 54600-Series Oscilloscopes using an instrument controller. This book, with the online *Agilent 54600-Series Oscilloscopes Programmer's Reference*, provides a comprehensive description of the oscilloscope's programmatic interface. The *Programmer's Reference* is supplied as a Microsoft Windows Help file on a 3.5" diskette.

To program the Agilent 54600-Series Oscilloscope, you need an interface module, such as the Agilent 54650A or 54651A. You also need an instrument controller that supports either the IEEE-488 or RS-232-C interface standards, and a programming language capable of communicating with these interfaces. You can also use the Agilent 54655A/56A Test Automation Module and the Agilent 54658A/59B Measurement/Storage Module.

Chapter 1 gives a general overview of oscilloscope programming.

Chapter 2 shows a simple program, explains its operation, and discusses considerations for data types.

Chapter 3 discusses the general considerations for programming the instrument over an GPIB interface.

Chapter 4 discusses the general considerations for programming the instrument over an RS-232-C interface.

Chapter 5 describes conventions used in representing syntax of commands throughout this book and in the online *Agilent 54600-Series Oscilloscopes Programmer's Reference*, and gives an overview of the command set.

Chapter 6 discusses the oscilloscope status registers and how to use them in your programs.

Chapter 7 tells how to install the *Agilent 54600-Series Oscilloscopes Programmer's Reference* online help file in Microsoft Windows, and explains help file navigation.

Chapter 8 lists all the commands and queries available for programming the oscilloscope.

For information on oscilloscope operation, see the *Agilent 54600-Series Oscilloscopes User and Service Guide*. For information on interface configuration, see the documentation for the oscilloscope and the interface card used in your controller (for example, the 82341C interface for IBM PC-compatible computers).

Introduction to Programming

Introduction to Programming

Chapters 1 and 2 introduce the basics for remote programming of an oscilloscope. The programming instructions in this manual conform to the IEEE 488.2 Standard Digital Interface for Programmable Instrumentation. The programming instructions provide the means of remote control.

To program the Agilent 54600-series oscilloscope you must add either an GPIB (for example, Agilent 54650A) or RS-232-C (for example, Agilent 54651A) interface to the rear panel.

You can perform the following basic operations with a controller and an oscilloscope:

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

Other tasks are accomplished by combining these basic functions.

Languages for Program Examples

The programming examples for individual commands in this manual are written in Agilent BASIC, C, or SICL C.

Talking to the Instrument

Computers acting as controllers communicate with the instrument by sending and receiving messages over a remote interface. Instructions for programming normally appear as ASCII character strings embedded inside the output statements of a “host” language available on your controller. The input statements of the host language are used to read in responses from the oscilloscope.

For example, Agilent BASIC uses the OUTPUT statement for sending commands and queries. After a query is sent, the response is usually read in using the ENTER statement.

Messages are placed on the bus using an output command and passing the device address, program message, and terminator. Passing the device address ensures that the program message is sent to the correct interface and instrument.

The following Agilent BASIC statement sends a command which sets the bandwidth limit of channel 1 on:

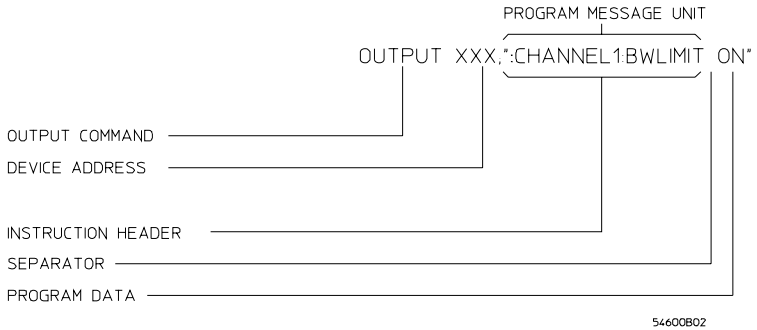
```
OUTPUT < device address > ;":CHANNEL1:BWLIMIT ON"<terminator>
```

The < device address > represents the address of the device being programmed. Each of the other parts of the above statement are explained in the following pages.

Program Message Syntax

To program the instrument remotely, you must understand the command format and structure expected by the instrument. The IEEE 488.2 syntax rules govern how individual elements such as headers, separators, program data, and terminators may be grouped together to form complete instructions. Syntax definitions are also given to show how query responses are formatted. The figure below shows the main syntactical parts of a typical program statement.

Figure 1-1



Program Message Syntax

Output Command

The output command is entirely dependent on the programming language. Throughout this manual, Agilent BASIC is used in most examples of individual commands. If you are using other languages, you will need to find the equivalents of Agilent BASIC commands like OUTPUT, ENTER, and CLEAR in order to convert the examples. The instructions listed in this manual are always shown between quotation marks in the example programs.

Device Address

The location where the device address must be specified is also dependent on the programming language you are using. In some languages, this may be specified outside the output command. In Agilent BASIC, this is always specified after the keyword OUTPUT. The examples in this manual assume the oscilloscope is at device address 707. When writing programs, the address varies according to how the bus is configured.

Instructions

Instructions (both commands and queries) normally appear as a string embedded in a statement of your host language, such as BASIC, Pascal, or C. The only time a parameter is not meant to be expressed as a string is when the instruction's syntax definition specifies <block data>, such as learnstring. There are only a few instructions which use block data.

Instructions are composed of two main parts:

- The header, which specifies the command or query to be sent.
- The program data, which provide additional information needed to clarify the meaning of the instruction.

Instruction Header

The instruction header is one or more mnemonics separated by colons (:) that represent the operation to be performed by the instrument. The command tree in chapter 5 illustrates how all the mnemonics can be joined together to form a complete header (see chapter 5, "Programming and Documentation Conventions").

The example in figure 1 is a command. Queries are indicated by adding a question mark (?) to the end of the header. Many instructions can be used as either commands or queries, depending on whether or not you have included the question mark. The command and query forms of an instruction usually have different program data. Many queries do not use any program data.

White Space (Separator)

White space is used to separate the instruction header from the program data. If the instruction does not require any program data parameters, you do not need to include any white space. In this manual, white space is defined as one or more spaces. ASCII defines a space to be character 32 (in decimal).

Program Data

Program data are used to clarify the meaning of the command or query. They provide necessary information, such as whether a function should be on or off, or which waveform is to be displayed. Each instruction's syntax definition shows the program data, as well as the values they accept. The section "Program Data Syntax Rules" in this chapter has all of the general rules about acceptable values.

When there is more than one data parameter, they are separated by commas (,). Spaces can be added around the commas to improve readability.

Header Types

There are three types of headers:

- Simple Command headers.
- Compound Command headers.
- Common Command headers.

Simple Command Header Simple command headers contain a single mnemonic. AUTOSCALE and DIGITIZE are examples of simple command headers typically used in this instrument. The syntax is:

```
<program mnemonic><terminator>
```

Simple command headers must occur at the beginning of a program message; if not, they must be preceded by a colon.

When program data must be included with the simple command header (for example, :DIGITIZE CHAN1), white space is added to separate the data from the header. The syntax is:

```
<program mnemonic><separator><program data><terminator>
```

Compound Command Header Compound command headers are a combination of two program mnemonics. The first mnemonic selects the subsystem, and the second mnemonic selects the function within that subsystem. The mnemonics within the compound message are separated by colons. For example:

To execute a single function within a subsystem:

```
:<subsystem>:<function><separator><program data><terminator>
```

(For example :CHANNEL1:BWLIMIT ON)

Common Command Header Common command headers control IEEE 488.2 functions within the instrument (such as clear status). Their syntax is:

```
*<command header><terminator>
```

No space or separator is allowed between the asterisk (*) and the command header. *CLS is an example of a common command header.

Combining Commands from the Same Subsystem

To execute more than one function within the same subsystem a semi-colon (;) is used to separate the functions:

```
:<subsystem>:<function><separator><data>;
     <function><separator><data><terminator>
```

(For example :CHANNEL1:COUPLING DC;BWLIMIT ON)

Duplicate Mnemonics

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

```
:CHANNEL1:RANGE   .4
```

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

```
:TIMEBASE:RANGE 1
```

sets the horizontal time base to 1 second full scale.

CHANNEL1 and TIMEBASE are subsystem selectors and determine which range is being modified.

Query Command

Command headers immediately followed by a question mark (?) are queries. After receiving a query, the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the answer is transmitted across the bus to the designated listener (typically a controller). For example, the query :TIMEBASE:RANGE? places the current time base setting in the output queue. In Agilent BASIC, the controller input statement:

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

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

Query commands are used to find out how the instrument is currently configured. They are also used to get results of measurements made by the instrument. For example, the command :MEASURE:RISETIME? instructs the instrument to measure the rise time of your waveform and places the result in the output queue.

The output queue must be read before the next program message is sent. For example, when you send the query :MEASURE:RISETIME? you must follow that query with an input statement. In Agilent BASIC, this is usually done with an ENTER statement immediately followed by a variable name. This statement reads the result of the query and places the result in a specified variable.

Read the Query Result First

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

Program Header Options

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

Program command and query headers may be sent in either long form (complete spelling), short form (abbreviated spelling), or any combination of long form and short form.

```
TIMEBASE:DELAY 1US  - long form
TIM:DEL 1US  - short form
```

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.

Command Syntax Programming Rules

The rules for the short form syntax are shown in chapter 5, "Programming and Documentation Conventions."

Program Data Syntax Rules

Program data is used to convey a variety of types of parameter information related to the command header. At least one space must separate the command header or query header from the program data.

```
<program mnemonic><separator><data><terminator>
```

When a program mnemonic or query has multiple program data a comma separates sequential program data.

```
<program mnemonic><separator><data>,<data><terminator>
```

For example, :MEASURE:TVOLT 1.0V,2 has two program data: 1.0V and 2.

There are two main types of program data which are used in commands: character and numeric program data.

Character Program Data

Character program data is used to convey parameter information as alpha or alphanumeric strings. For example, the :TIMEBASE:MODE command can be set to normal, delayed, XY, or ROLL. The character program data in this case may be NORMAL, DELAYED, XY, or ROLL. The command :TIMEBASE:MODE DELAYED sets the time base mode to delayed.

The available mnemonics for character program data are always included with the instruction's syntax definition. When sending commands, either the long form or short form (if one exists) may be used. Upper-case and lower-case letters may be mixed freely. When receiving query responses, upper-case letters are used exclusively.

Numeric Program Data

Some command headers require program data to be expressed numerically. For example, :TIMEBASE:RANGE requires the desired full scale range to be expressed numerically.

For numeric program data, you have the option of using exponential notation or using suffix multipliers to indicate the numeric value. The following numbers are all equal:

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

When a syntax definition specifies that a number is an integer, that means that the number should be whole. Any fractional part would be ignored, truncating the number. Numeric data parameters which accept fractional values are called real 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.

Embedded Strings

Embedded strings contain groups of alphanumeric characters which are treated as a unit of data by the oscilloscope. 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.

Program Message Terminator

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

New Line Terminator Functions

The NL (New Line) terminator has the same function as an EOS (End Of String) and EOT (End Of Text) terminator.

Selecting Multiple Subsystems

You can send multiple program commands and program queries for different subsystems on the same line by separating each command with a semicolon. The colon following the semicolon enables you to enter a new subsystem. For example:

```
<program mnemonic><data>;:<program mnemonic><data><terminator>

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

Combining Compound and Simple Commands

Multiple commands may be any combination of compound and simple commands.

Programming Getting Started

Programming Getting Started

This chapter explains how to set up the instrument, how to retrieve setup information and measurement results, how to digitize a waveform, and how to pass data to the controller.

Languages for Programming Examples

The programming examples in this guide are written in Agilent BASIC, C, or SICL C.

Initialization

To make sure the bus and all appropriate interfaces are in a known state, begin every program with an initialization statement. Agilent BASIC provides a CLEAR command which clears the interface buffer:

```
CLEAR 707  ! initializes the interface of the instrument
```

When you are using GPIB, CLEAR also resets the oscilloscope's parser. The parser is the program which reads in the instructions which you send it.

After clearing the interface, initialize the instrument to a preset state:

```
OUTPUT 707;"*RST"  ! initializes the instrument to a preset
state.
```

Information for Initializing the Instrument

The actual commands and syntax for initializing the instrument are discussed in the common commands section of the online *Agilent 54600-Series Oscilloscopes Programmer's Reference*.

Refer to your controller manual and programming language reference manual for information on initializing the interface.

Autoscale

The AUTOSCALE feature performs a very useful function on unknown waveforms by setting up the vertical channel, time base, and trigger level of the instrument.

The syntax for the autoscale function is:

```
:AUTOSCALE<terminator>
```

Setting Up the Instrument

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

```
:CHANNEL1:PROBE X10;RANGE 16;OFFSET 1.00<terminator>
:TIMEBASE:MODE NORMAL;RANGE 1E-3;DELAY 100E-6<terminator>
```

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

Example Program

This program demonstrates the basic command structure used to program the oscilloscope.

```
10     CLEAR 707                                ! Initialize instrument interface
20     OUTPUT 707;"*RST"                        ! Initialize inst to preset state
30     OUTPUT 707;":TIMEBASE:RANGE 5E-4"        ! Time base to 50 us/div
40     OUTPUT 707;":TIMEBASE:DELAY 0"           ! Delay to zero
50     OUTPUT 707;":TIMEBASE:REFERENCE CENTER" ! Display reference at center
60     OUTPUT 707;":CHANNEL1:PROBE X10"         ! Probe attenuation to 10:1
70     OUTPUT 707;":CHANNEL1:RANGE 1.6"         ! Vertical range to 1.6 V full scale
80     OUTPUT 707;":CHANNEL1:OFFSET -.4"        ! Offset to -0.4
90     OUTPUT 707;":CHANNEL1:COUPLING DC"       ! Coupling to DC
100    OUTPUT 707;":TRIGGER:MODE NORMAL"        ! Normal triggering
110    OUTPUT 707;":TRIGGER:LEVEL -.4"          ! Trigger level to -0.4
120    OUTPUT 707;":TRIGGER:SLOPE POSITIVE"     ! Trigger on positive slope
130    OUTPUT 707;":ACQUIRE:TYPE NORMAL"        ! Normal acquisition
140    OUTPUT 707;":DISPLAY:GRID OFF"           ! Grid off
150    END
```

- Line 10 initializes the instrument interface to a known state.
- Line 20 initializes the instrument to a preset state.
- Lines 30 through 50 set the time base mode to normal with the horizontal time at 50 µs/div with 0 s of delay referenced at the center of the graticule.
- Lines 60 through 90 set the vertical range to 1.6 volts full scale with center screen at -0.4 volts with 10:1 probe attenuation and DC coupling.
- Lines 100 through 120 configure the instrument to trigger at -0.4 volts with normal triggering.
- Line 130 configures the instrument for normal acquisition.
- Line 140 turns the grid off.

Using the DIGitize Command

The DIGitize command is a macro that captures data satisfying the specifications set up by the ACQuire subsystem. When the digitize process is complete, the acquisition is stopped. The captured data can then be measured by the instrument or transferred to the controller for further analysis. The captured data consists of two parts: the waveform data record and the preamble.

Ensure New Data is Collected

After changing the oscilloscope configuration, the waveform buffers are cleared. Before doing a measurement, the DIGitize command should be sent to the oscilloscope to ensure new data has been collected.

When you send the DIGitize command to the oscilloscope, 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.

Set :TIMebase:MODE to NORMal when Using :DIGitize

:TIMebase:MODE must be set to NORMal to perform a :DIGitize or to perform any WAVeform subsystem query. A "Settings conflict" error message will be returned if these commands are executed when MODE is set to ROLL, XY, or DELayed. Sending the *RST (reset) command will also set the time base mode to normal.

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"<terminator>
OUTPUT 707;":ACQUIRE:COMPLETE 100"<terminator>
OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"<terminator>
OUTPUT 707;":WAVEFORM:FORMAT BYTE"<terminator>
OUTPUT 707;":ACQUIRE:COUNT 8"<terminator>
OUTPUT 707;":WAVEFORM:POINTS 500"<terminator>
OUTPUT 707;":DIGITIZE CHANNEL1"<terminator>
OUTPUT 707;":WAVEFORM:DATA?"<terminator>
```

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.

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

Digitized waveforms are passed from the instrument to the controller 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 transferring a digitized waveform depends on data structures, formatting available 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, see the waveform subsystem commands and corresponding program code examples in the online *Agilent 54600-Series Oscilloscopes Programmer's Reference*.

Aborting a Digitize Operation Over GPIB

When using GPIB, a digitize operation may be aborted by sending a Device Clear over the bus (CLEAR 707).

Receiving Information from the Instrument

After receiving a query (command header followed by a question mark), the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the answer is transmitted across the interface to the designated listener (typically a controller). The input statement for receiving a response message from an instrument's output queue typically has two parameters; the device address, and a format specification for handling the response message. For example, to read the result of the query command :CHANNEL1:COUPLING? you would execute the Agilent BASIC statement:

```
ENTER <device address> ;Setting$
```

where <device address> represents the address of your device. This would enter the current setting for the channel one coupling in the string variable Setting$.

All results for queries sent in a program message must be read before another program message is sent. For example, when you send the query :MEASURE:RISETIME?, you must follow that query with an input statement. In Agilent BASIC, this is usually done with an ENTER statement.

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

Executing an input statement before sending a query causes the controller to wait indefinitely.

The format specification for handling response messages is dependent on both the controller and the programming language.

String Variables

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.

Express String Variables Using Exact Syntax

In Agilent BASIC, string variables are case sensitive and must be expressed exactly the same each time they are used.

Address Varies According to Configuration

For the example programs in the help file, assume that the device being programmed is at device address 707. The actual address varies according to how you have configured the bus for your own application.

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
```

Numeric Variables

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
```

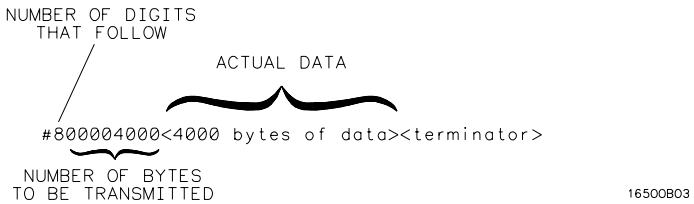
After running this program, the controller displays:

```
.8
```

Definite-Length Block Response Data

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

For example, for transmitting 4000 bytes of data, the syntax would be:



The “8” states the number of digits that follow, and “00004000” states the number of bytes to be transmitted.

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 the string variable Results$ with the command:

```
ENTER 707;Results$
```

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

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

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

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

Instrument Status

Status registers track the current status of the instrument. By checking the instrument status, you can find out whether an operation has been completed, whether the instrument is receiving triggers, and more. Chapter 6, “Status Reporting” explains how to check the status of the instrument.

Programming over GPIB

Programming over GPIB

This section describes the GPIB interface functions and some general concepts. In general, these functions are defined by IEEE 488.1. They deal with general interface management issues, as well as messages which can be sent over the interface as interface commands.

For more information on connecting the controller to the oscilloscope, see the documentation for the GPIB interface card you are using.

Interface Capabilities

The interface capabilities of the oscilloscope, as defined by IEEE 488.1, are SH1, AH1, T5, L4, SR1, RL1, PP0, DC1, DT1, C0, and E2.

Command and data concepts

The interface has two modes of operation:

- command mode
- data mode

The bus is in the command mode when the ATN line is true. The command mode is used to send talk and listen addresses and various bus commands, such as a group execute trigger (GET).

The bus is in the data mode when the ATN line is false. The data mode is used to convey device-dependent messages across the bus. The device-dependent messages include all of the instrument commands and responses.

Addressing

Set the instrument address by using the front panel controls on the oscilloscope after the GPIB interface has been installed on the rear panel of the oscilloscope.

1 Press `Print/Utility` , then press the **I/O Menu** softkey.

2 Press the **Inst Addr** softkey to select the instrument address. Increment the address by successively pressing the **Inst Addr** softkey. The address can also be incremented or decremented by turning the knob closest to the `Cursors` key.

- Each device on the GPIB resides at a particular address, ranging from 0 to 30.
- The active controller specifies which devices talk and which listen.
- An instrument may be talk addressed, listen addressed, or unaddressed by the controller.

If the controller addresses the instrument to talk, the instrument remains configured to talk until it receives an interface clear message (IFC), another instrument's talk address (OTA), its own listen address (MLA), or a universal untalk command (UNT).

If the controller addresses the instrument to listen, the instrument remains configured to listen until it receives an interface clear message (IFC), its own talk address (MTA), or a universal unlisten command (UNL).

Communicating over the bus

Since GPIB can address multiple devices through the same interface card, the device address passed with the program message must include not only the correct interface select code, but also the correct instrument address.

Interface Select Code (Selects Interface)

Each interface card has a unique interface select code. This code is used by the controller to direct commands and communications to the proper interface. The default is typically "7" for GPIB controllers.

Instrument Address (Selects Instrument)

Each instrument on an GPIB must have a unique instrument address between decimal 0 and 30. The device address passed with the program message must include not only the correct instrument address, but also the correct interface select code.

DEVICE ADDRESS = (Interface Select Code * 100) + (Instrument Address)

For example, if the instrument address for the oscilloscope is 4 and the interface select code is 7, when the program message is passed, the routine performs its function on the instrument at device address 704.

For the oscilloscope, the instrument address is typically set to

Oscilloscope Device Address

The examples in this manual and in the online Agilent 54600-Series Oscilloscopes Programmer's Reference assume the oscilloscope is at device address 707.

See the documentation for your GPIB interface card for more information on select codes and addresses.

Lockout

You can use the SYSTem:LOCK ON command to disable front-panel control while a program is running. By default, the instrument accepts and executes bus commands, and the front panel is entirely active.

Restore Front-Panel Control

Cycling power also restores front panel control.

With GPIB, the instrument is placed in the lockout mode by sending the local lockout command (LLO). The instrument can be returned to local by sending the go-to-local command (GTL) to the instrument.

Bus Commands

The following commands are IEEE 488.1 bus commands (ATN true). IEEE 488.2 defines many of the actions which are taken when these commands are received by the instrument.

Device Clear

The device clear (DCL) or selected device clear (SDC) commands clear the input and output buffers, reset the parser, and clear any pending commands. If either of these commands is sent during a digitize operation, the digitize operation is aborted.

Interface Clear (IFC)

The interface clear (IFC) command halts all bus activity. This includes unaddressing all listeners and the talker, disabling serial poll on all devices, and returning control to the system controller.

Programming over RS-232-C

Programming over RS-232-C

This section describes the interface functions and some general concepts of the RS-232-C. The RS-232-C interface on this instrument is Hewlett-Packard's implementation of EIA Recommended Standard RS-232-C, "Interface Between Data Terminal Equipment and Data Communications Equipment Employing Serial Binary Data Interchange." With this interface, data is sent one bit at a time and characters are not synchronized with preceding or subsequent data characters. Each character is sent as a complete entity without relationship to other events.

IEEE 488.2 Operates with IEEE 488.1 or RS-232-C

IEEE 488.2 is designed to work with IEEE 488.1 as the physical interface. When RS-232-C is used as the physical interface, as much of IEEE 488.2 is retained as the hardware differences will allow. No IEEE 488.1 messages such as DCL, GET, and END are available.

Interface Operation

The oscilloscope can be programmed with a controller over RS-232-C using either a minimum three-wire or extended hardwire interface. The operation and exact connections for these interfaces are described in more detail in the following sections. When you are programming the oscilloscope over RS-232-C with a controller, you are normally operating directly between two DTE (Data Terminal Equipment) devices as compared to operating between a DTE device and a DCE (Data Communications Equipment) device.

When operating directly between two RS-232-C devices, certain considerations must be taken into account. For three-wire operation, an XON/XOFF software handshake must be used to handle handshaking between the devices. For extended hardwire operation, handshaking may be handled either with XON/XOFF or by manipulating the CTS and RTS lines of the oscilloscope. For both three-wire and extended hardwire operation, the DCD and DSR inputs to the oscilloscope must remain high for proper operation.

With extended hardwire operation, a high on the CTS input allows the oscilloscope to send data and a low on this line disables the oscilloscope data transmission. Likewise, a high on the RTS line allows the controller to send data and a low on this line signals a request for the controller to disable data transmission. Since three-wire operation has no control over the CTS input, internal pull-up resistors in the oscilloscope ensure that this line remains high for proper three-wire operation.

Cables

Selecting a cable for the RS-232-C interface is dependent on your specific application. The following paragraphs describe which lines of the oscilloscope are used to control the operation of the RS-232-C bus relative to the oscilloscope. To locate the proper cable for your application, refer to the reference manual for your controller. This manual should address the exact method your controller uses to operate over the RS-232-C bus.

Minimum three-wire interface with software protocol

With a three-wire interface, the *software* (as compared to interface hardware) controls the data flow between the oscilloscope and the controller. This provides a much simpler connection between devices since you can ignore hardware handshake requirements. The oscilloscope uses the following connections on its RS-232-C interface for three-wire communication:

- Pin 7 SGND (Signal Ground)
- Pin 2 TD (Transmit Data from oscilloscope)
- Pin 3 RD (Receive Data into oscilloscope)

The TD (Transmit Data) line from the oscilloscope must connect to the RD (Receive Data) line on the controller. Likewise, the RD line from the oscilloscope must connect to the TD line on the controller. Internal pull-up resistors in the oscilloscope ensure the DCD, DSR, and CTS lines remain high when you are using a three-wire interface.

No Hardware Means to Control Data Flow

The three-wire interface provides no hardware means to control data flow between the controller and the oscilloscope. XON/OFF protocol is the only means to control this data flow.

Extended interface with hardware handshake

With the extended interface, both the software and the hardware can control the data flow between the oscilloscope and the controller. This allows you to have more control of data flow between devices. The oscilloscope uses the following connections on its RS-232-C interface for extended interface communication (on a 25-pin connector):

- Pin 7 SGND (Signal Ground)
- Pin 2 TD (Transmit Data from oscilloscope)
- Pin 3 RD (Receive Data into oscilloscope)

The additional lines you use depends on your controller's implementation of the extended hardwire interface.

- Pin 4 RTS (Request To Send) is an output from the oscilloscope which can be used to control incoming data flow.
- Pin 5 CTS (Clear To Send) is an input to the oscilloscope which controls data flow from the oscilloscope.
- Pin 6 DSR (Data Set Ready) is an input to the oscilloscope which controls data flow from the oscilloscope within two bytes.
- Pin 8 DCD (Data Carrier Detect) is an input to the oscilloscope which controls data flow from the oscilloscope within two bytes.
- Pin 20 DTR (Data Terminal Ready) is an output from the oscilloscope which is enabled as long as the oscilloscope is turned on.

The TD (Transmit Data) line from the oscilloscope must connect to the RD (Receive Data) line on the controller. Likewise, the RD line from the oscilloscope must connect to the TD line on the controller.

The RTS (Request To Send) line is an output from the oscilloscope which can be used to control incoming data flow. A high on the RTS line allows the controller to send data, and a low on this line signals a request for the controller to disable data transmission.

The CTS (Clear To Send), DSR (Data Set Ready), and DCD (Data Carrier Detect) lines are inputs to the oscilloscope which control data flow from the oscilloscope (Pin 2). Internal pull-up resistors in the oscilloscope assure the DCD and DSR lines remain high when they are not connected.

If DCD or DSR are connected to the controller, the controller must keep these lines and the CTS line high to enable the oscilloscope to send data to the controller. A low on any one of these lines will disable the oscilloscope data transmission. Dropping the CTS line low during data transmission will stop oscilloscope data transmission immediately. Dropping either the DSR or DCD line low during data transmission will stop oscilloscope data transmission, but as many as two additional bytes may be transmitted from the oscilloscope.

Configuring the Interface

Set the baud rate and handshake protocol by using the front panel controls on the oscilloscope after the RS-232-C interface has been installed on the rear panel of the oscilloscope.

1. Press `Print/Utility` , then press the **I/O Menu** softkey.
2. To change the baud rate, press the **Baud Rate** softkey until the desired baud rate is displayed.
3. To change the handshake protocol, toggle the **Handshake** softkey until the desired protocol is displayed.

Interface Capabilities

The baud rate, stop bits, parity, handshake protocol, and data bits must be configured exactly the same for both the controller and the oscilloscope to properly communicate over the RS-232-C bus. The oscilloscope's RS-232-C interface capabilities are as follows:

- Baud Rate: 1200, 2400, 9600, or 19,200
- Stop Bits: 1
- Parity: None
- Protocol: DTR or XON/XOFF
- Data Bits: 8

Protocol

DTR (Data Terminal Ready) With a three-wire interface, selecting DTR for the handshake protocol does not allow the sending or receiving device to control data flow. No control over the data flow increases the possibility of missing data or transferring incomplete data.

With an extended hardwire interface, selecting DTR allows a hardware handshake to occur. With hardware handshake, hardware signals control data flow.

XON/XOFF XON/XOFF stands for Transmit On/Transmit Off. With this mode the receiver (controller or oscilloscope) controls data flow and can request that the sender (oscilloscope or controller) stop data flow. By sending XOFF (ASCII 17) over its transmit data line, the receiver requests that the sender disables data transmission. A subsequent XON (ASCII 19) allows the sending device to resume data transmission.

A controller sending data to the oscilloscope should send no more than 32 bytes of data after an XOFF.

The oscilloscope will not send any data after an XOFF is received until an XON is received.

Data Bits

Data bits are the number of bits sent and received per character that represent the binary code of that character.

Information is stored in bytes (8 bits at a time) in the oscilloscope. Data can be sent and received just as it is stored, without the need to convert the data.

Communicating over the RS-232-C bus

Each RS-232-C interface card has its own interface select code. This code is used by the controller to direct commands and communications to the proper interface. Unlike GPIB, which allows multiple devices to be connected through a single interface card, RS-232-C is only connected between two devices at a time through the same interface card. Because of this, only the interface code is required for the device address.

Generally, the interface select code can be any decimal value between 0 and 31, except for those interface codes which are reserved by the controller for internal peripherals and other internal interfaces. This value can be selected through switches on the interface card. For more information, refer to the reference manual for your interface card or controller.

For example, if your RS-232-C interface select code is 20, the device address required to communicate over the RS-232-C bus is 20.

Lockout Command

To lockout the front panel controls use the system command LOCK. When this function is on, all controls (except the power switch) are entirely locked out. Local control can only be restored by sending the command :SYSTEM:LOCK OFF.

Restoring Local Control

Cycling the power will also restore local control, but this will also reset certain RS-232-C states.

Programming and Documentation Conventions

Programming and Documentation Conventions

This chapter covers conventions which are used in programming the instrument, as well as conventions used in the online *Agilent 54600-Series Oscilloscopes Programmer's Reference* and the remainder of this manual. This chapter also contains a detailed description of the command tree and command tree traversal.

Command Set Organization

The command set is divided into common commands, root level commands and sets of subsystem commands. Each of the groups of commands is described in the *Agilent 54600-Series Oscilloscopes Programmer's Reference*, which is supplied as an online help file for Microsoft Windows. See chapter 8 for information on installing and using the help file.

The commands are shown using upper and lowercase letters. As an example, AUToscale indicates that the entire command name is AUTOSCALE. To speed up the transfer, the short form AUT is also accepted by the oscilloscope. Each command listing contains a description of the command and its arguments and the command syntax. Some commands have a programming example.

The subsystems are listed below:

SYSTem controls some basic functions of the oscilloscope.

ACQuire sets the parameters for acquiring and storing data.

CHANnel controls all oscilloscope functions associated with individual channels or groups of channels.

DISPlay controls how waveforms, graticule, and text are displayed and written on the screen.

FUNCtion controls functions in the Measurement/Storage Module.

MASK controls the waveform monitoring test function available when using the Measurement/Storage Module.

MEASure selects the automatic measurements to be made and controls time markers.

SEQuence controls the automated testing functions when using the Test Automation Module.

TIMebase controls all horizontal sweep functions.

TRACe controls features used with trace memories.

TRIGger controls the trigger modes and parameters for each trigger type.

EXTernal Trigger defines the conditions for an external trigger. These commands only appear on the Agilent 54610, 54615, and 54616 Oscilloscopes.

WAVeform provides access to waveform data.

Table 5-1

Alphabetic Command Cross-Reference

Command	Subsystem Where Used	Command	Subsystem Where Used	Command	Subsystem Where Used
ALL	MEASure	*ESR	Common	NEXT	SEQuence
ASTore	Root level			NREJect	TRIGger
AUToscale	Root level	FAILmode	MASK	NUMBer	MASK
		FALLtime	MEASure	NWIDth	MEASure
BLANk	Root level	FIELd	TRIGger		
BWLimit	CHANnel	FORMat	WAVeform	OFFSet	CHANnel
BYTeorder	WAVeform	FREQuency	MEASure	OFFSet	EXTernal trigger [2]
				OFFSet	FUNCtion
CENTer	FUNCtion	GRID	DISPlay	*OPC	Common
CLEAR	TRACe			OPERation	FUNCtion
*CLS	Common	HOLDoff	TRIGger	*OPT	Common
COLumn	DISPlay			OPTMode	TRIGger
COMPlete	ACQuire	*IDN	Common	OVERshoot	MEASure
CONNect	DISPlay	INCRement	MASK		
COUNt	ACQuire	INPut	CHANnel [1]	PALette	DISPlay [3]
COUPling	CHANnel	INPut	EXTernal trigger [1]	PEAKs	FUNCtion
COUPling	EXTernal trigger [1]	INVerse	DISPlay	PERiod	MEASure
COUPling	TRIGger	INVert	CHANnel	PHASe	MEASure
CREATE	MASK			PIXel	DISPlay
		KEY	SYSTem	PMODe	CHANnel [1]
DATA	DISPlay			PMODe	EXTernal trigger [1]
DATA	MASK	LEVel	TRIGger	POINts	ACQuire
DATA	TRACe	LINE	DISPlay	POINts	WAVeform
DATA	WAVeform	LINE	TRIGger	POLarity	TRIGger
DEFine	MEASure	LOCK	SYSTem	POSTfailure	MASK
DELay	MEASure	LOWer	MEASure	PREamble	WAVeform
DELay	TIMebase	*LRN	Common	PREShoot	MEASure
DESTination	MASK			PREVious	SEQuence
DIGitize	Root level	MATH	CHANnel	PRINt	Root level
DITHer	Root level	MENU	Root level	PROBe	CHANnel
DSP	SYSTem	MERGe	Root level	PROBe	EXTernal trigger [1]
DUTycycle	MEASure	MODE	TIMebase	PROTect	CHANnel [1]
		MODE	TRACe	PROTect	EXTernal trigger [1]
ERASe	Root level	MODE	TRIGger	PROTect	SEQuence
ERRor	SYSTem	MOVE	FUNCtion	PSTArt	MEASure
*ESE	Common			PSTOp	MEASure
				PWIDth	MEASure

[1] These commands are used by the Agilent 54610, 54615, and 54616 only.
[2] These commands are used by the Agilent 54610 only.
[3] This command is used only by the Agilent 54616C.

Command	Subsystem Where Used	Command	Subsystem Where Used	Command	Subsystem Where Used
RANGe	CHANnel	*STB	Common	VSTOp	MEASure
RANGe	FUNCtion	STEP	SEQuence	VTIMe	MEASure
RANGe	TIMebase	STOP	Root level	VTOP	MEASure
*RCL	Common				
REFerence	FUNCtion	TDELta	MEASure	*WAI	Common
REFerence	TIMebase	TER	Root level	WINDow	FUNCtion
REJect	TRIGger	TEST	MASK		
RESet	SEQuence	TEST?	SEQuence	XINCrement	WAVeform
RISetime	MEASure	TEXT	DISPlay	XORigin	WAVeform
ROW	DISPlay	THResholds	MEASure	XREFerence	WAVeform
*RST	Common	TOLerance	MASK		
RUN	Root level	*TRG	Common	YINCrement	WAVeform
		*TST	Common	YORigin	WAVeform
*SAV	Common	TSTArt	MEASure	YREFerence	WAVeform
SAVE	MASK	TSTOp	MEASure		
SAVE	TRACe	TVHFrej	TRIGger		
SCRatch	MEASure	TVMode	TRIGger		
SET100	MEASure	TVOLt	MEASure		
SET360	MEASure	TYPE	ACQuire		
SETup	ACQuire	TYPE	WAVeform		
SETup	CHANnel				
SETup	DISPlay	UPPer	MEASure		
SETup	EXTernal trigger [1]				
SETup	SEQuence	VAMPlitude	MEASure		
SETup	SYSTem	VAUToscale	Root level		
SETup	TIMebase	VAVerage	MEASure		
SETup	TRIGger	VBASe	MEASure		
SHOW	MEASure	VDELta	MEASure		
SKEW	CHANnel [1]	VERNier	CHANnel		
SKEW	EXTernal trigger [2]	VERNier	TIMebase [4]		
SLOPe	TRIGger	VIEW	FUNCtion		
SOURce	DISPlay	VIEW	Root level		
SOURce	FUNCtion	VIR	TRIGger		
SOURce	MEASure	VMAX	MEASure		
SOURce	TRIGger	VMIN	MEASure		
SOURce	WAVeform	VPP	MEASure		
SPAN	FUNCtion	VPSTart	MEASure		
*SRE	Common	VPSTop	MEASure		
STANdard	TRIGger	VRMS	MEASure		
STATistics	MASK	VSTArt	MEASure		
STATus	Root level				

[1] These commands are used by the Agilent 54610, 54615, and 54616 only.
[2] These commands are used by the Agilent 54610 only.
[4] These commands are used by all oscilloscope models except Agilent 54615 and 54616.

The Command Tree

The command tree shows all of the commands and the relationship of the commands to each other. The IEEE 488.2 common commands are not listed as part of the command tree since they do not affect the position of the parser within the tree. When a program message terminator (<NL>, linefeed - ASCII decimal 10) or a leading colon (:) is sent to the instrument, the parser is set to the "root" of the command tree.

Command Types

The commands for this instrument can be placed into three types:

- Common commands
- Root level commands
- Subsystem commands

Common Commands The common commands are the commands defined by IEEE 488.2. These commands control some functions that are common to all IEEE 488.2 instruments.

Common commands are independent of the tree, and do not affect the position of the parser within the tree. These commands differ from root level commands in that root level commands place the parser back at the root of the command tree.

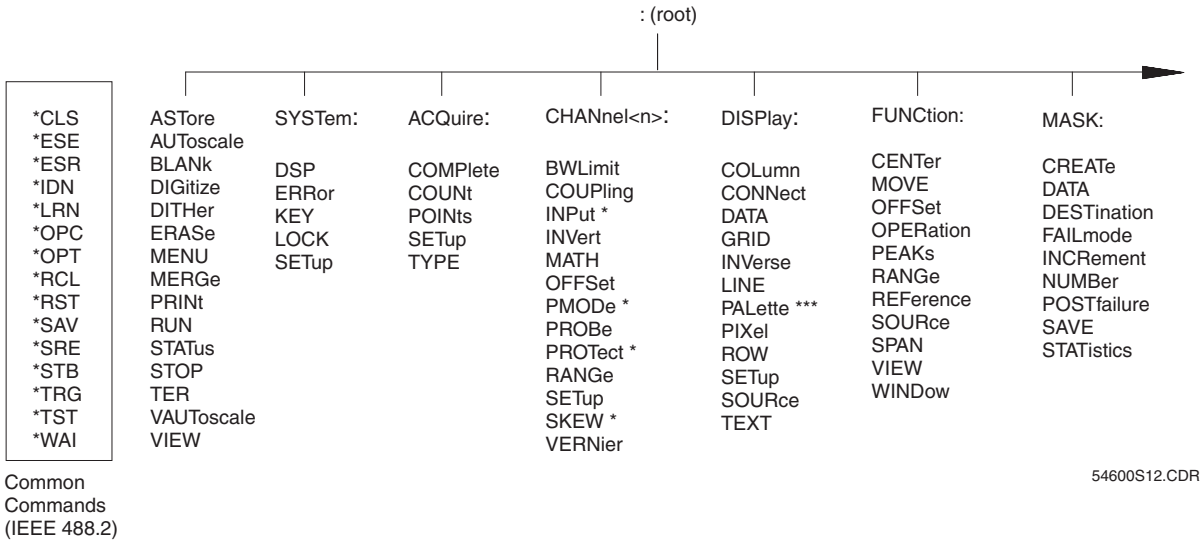
Example:

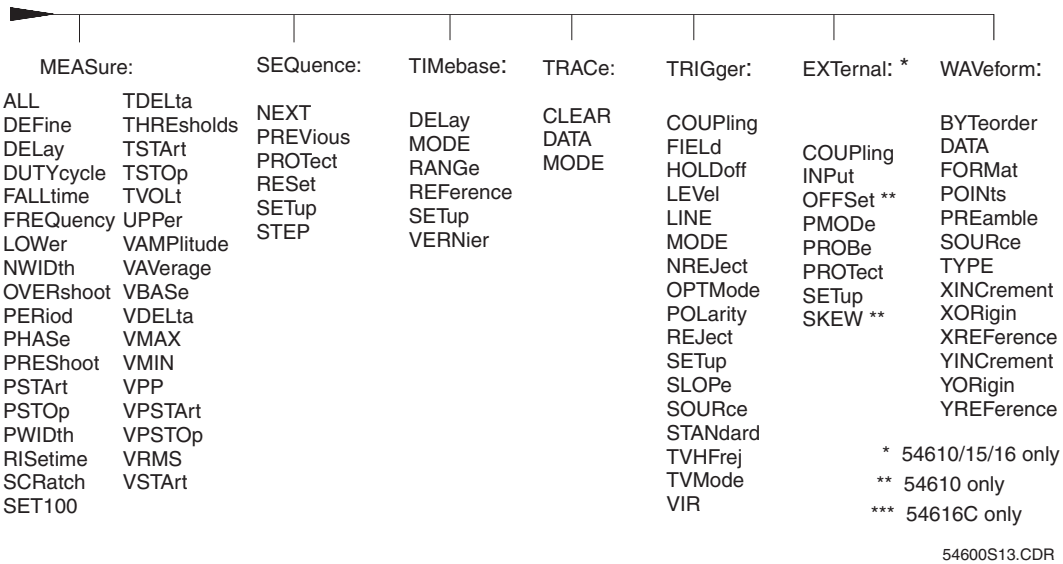
```
*RST
```

Root Level Commands The root level commands control many of the basic functions of the instrument. These commands reside at the root of the command tree. Root level commands are always parsable if they occur at the beginning of a program message, or are preceded by a colon.

Example:

```
:AUTOSCALE
```





Subsystem Commands

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

Tree Traversal Rules

Command headers are created by traversing down the command tree. A legal command header from the command tree would be :CHANNEL1:RANGE. This is called a compound header. A compound header is a header made of two or more mnemonics separated by colons. The mnemonic created contains no spaces. The following rules apply to traversing the tree:

- A leading colon or a <program message terminator> (either an <NL> or EOI true on the last byte) places the parser at the root of the command tree. A leading colon is a colon that is the first character of a program header.
- Executing a subsystem command places you in that subsystem until a leading colon or a <program message terminator> is found. In the Command Tree, use the last mnemonic in the compound header as a reference point (for example, RANGE). Then find the last colon above that mnemonic (CHANNEL<n>). That is the point where the parser resides. Any command below that point can be sent within the current program message without sending the mnemonics that appear above them (for example, OFFSET).

Examples

The OUTPUT statements in the examples are written using Agilent BASIC 5.0. The quoted string is placed on the bus, followed by a carriage return and linefeed (CRLF).

Example 1

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

The colon between CHANNEL1 and RANGE is necessary because CHANNEL1:RANGE is a compound command. The semicolon between the RANGE command and the OFFSET command is the required program message unit separator. The OFFSET command does not need CHANNEL1 preceding it, since the CHANNEL1:RANGE command sets the parser to the CHANNEL1 node in the tree.

Example 2

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

or

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

In the first line of example 2, the “subsystem selector” is implied for the DELAY command in the compound command. The DELAY command must be in the same program message as the REFERENCE command, since the program message terminator places the parser back at the root of the command tree.

A second way to send these commands is by placing TIMEBASE: before the DELAY command as shown in the second part of example 2.

Example 3:

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

The leading colon before CHANNEL1 tells the parser to go back to the root of the command tree. The parser can then see the CHANNEL1:OFFSET command.

Truncation Rules

The truncation rule for the mnemonics used in headers and alpha arguments is:

The mnemonic is the first four characters of the keyword unless:

The fourth character is a vowel, then the mnemonic is the first three characters of the keyword.

This rule is not used if the length of the keyword is exactly four characters.

Some examples of how the truncation rule is applied to various commands are shown in the following table.

Mnemonic Truncation

Long Form	Short Form
RANGE	RANG
PATTERN	PATT
TIMEBASE	TIM
DELAY	DEL
TYPE	TYPE

Infinity Representation

The representation of infinity is 9.9E+37. This is also the value returned when a measurement cannot be made.

Sequential and Overlapped Commands

IEEE 488.2 distinguishes between sequential and overlapped commands. Sequential commands finish their task before the execution of the next command starts. Overlapped commands run concurrently. Commands following an overlapped command may be started before the overlapped command is completed. All of the commands are sequential.

Response Generation

As defined by IEEE 488.2, query responses may be buffered for the following conditions:

- When the query is parsed by the instrument.
- When the controller addresses the instrument to talk so that it may read the response.

The responses to a query are buffered when the query is parsed.

Notation Conventions and Definitions

The following conventions and definitions are used in this manual and the online *Agilent 54600-Series Oscilloscopes Programmer's Reference* in descriptions of remote operation:

Conventions

< > Angle brackets enclose words or characters that symbolize a program code parameter or an interface command.

::= "is defined as." For example, <A> ::= <B> indicates that <A> can be replaced by <B> in any statement containing <A>.

| "or." Indicates a choice of one element from a list. For example, <A> | <B> indicates <A> or <B>, but not both.

... An ellipsis (trailing dots) indicates that the preceding element may be repeated one or more times.

[ ] Square brackets indicate that the enclosed items are optional.

{ } When several items are enclosed by braces, one, and only one of these elements must be selected.

Definitions

d ::= A single ASCII numeric character, 0-9.

n ::= A single ASCII non-zero, numeric character, 1-9.

<NL> ::= Newline or Linefeed (ASCII decimal 10).

<sp> ::= <white space>

<white space> ::= 0 through 32 (decimal) except linefeed (decimal 10). The nominal value is 32 (the space character).

Program Examples

The program examples given for commands in the online *Agilent 54600-Series Oscilloscopes Programmer's Reference* were written using the Agilent BASIC for Windows, C, and SICL C programming languages. The programs always assume the oscillscope is at address 7 and the interface is at address 7 for a program address of 707. If a printer is used, it is always assumed to be at address 701.

In these examples, give special attention to the ways in which the command or query can be sent. The way the instrument is set up to respond to a command or query has no bearing on how you send the command or query. That is, the command or query can be sent using the long form or short form, if a short form exists for that command. You can send the command or query using upper case (capital) letters or lower case (small) letters. Also, the data can be sent using almost any form you wish. If you are sending a timebase range value of 100 ms, that value could be sent using a decimal (.1), or an exponential (1e-1 or 1.0E-1), or a suffix (100 ms or 100MS).

As an example, set the sweep speed to 100 ms by sending one of the following:

- Commands in long form using the decimal format.

```
OUTPUT 707;":CHANNEL1:RANGE .1"
```

- Commands in short form using an exponential format.

```
OUTPUT 707;":CHAN1:RANG 1E-1"
```

- Commands using lower case letters, short forms, and a suffix.

```
OUTPUT 707;":chan1:rang 100 mV"
```

Including the Colon Is Optional

In these examples, the colon as the first character of the command is optional. The space between RANGE and the argument is required.

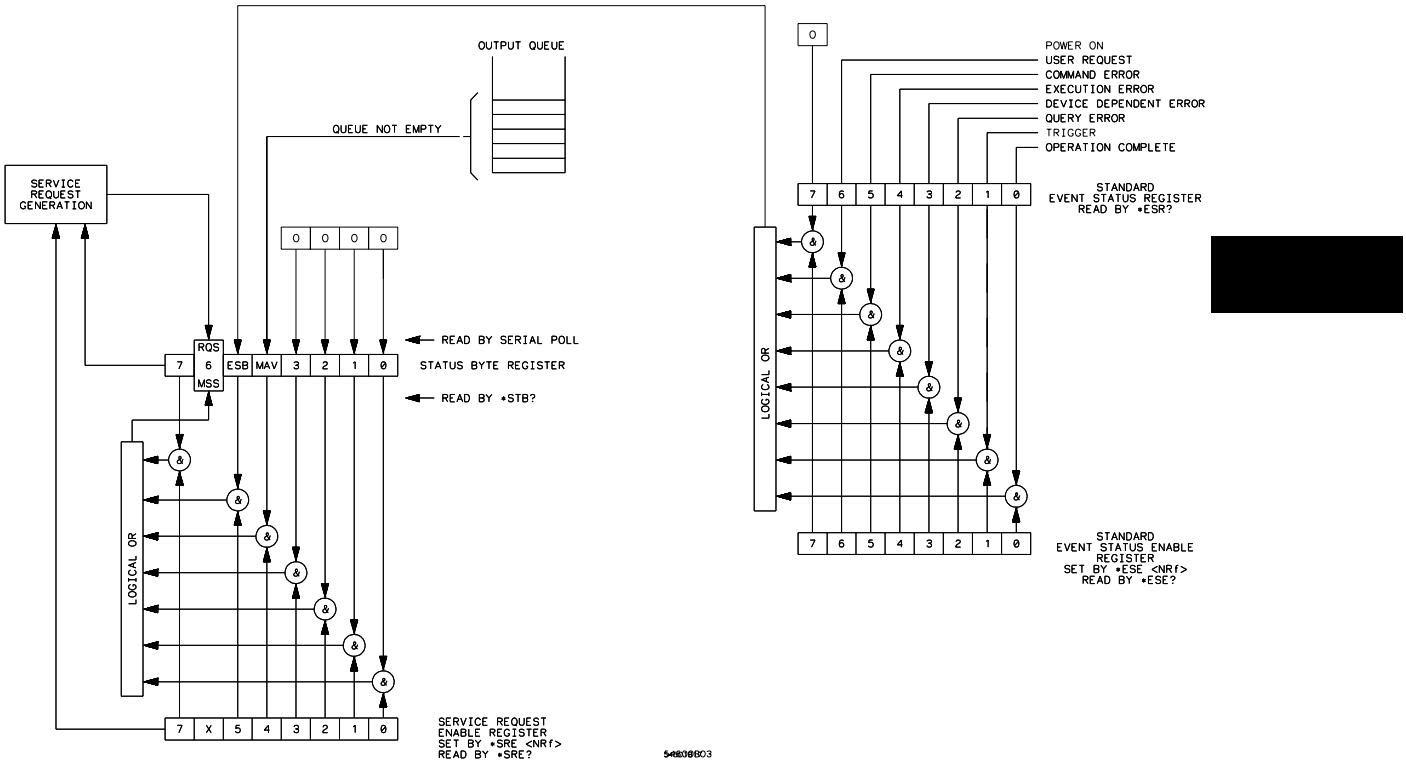
Status Reporting

Status Reporting

IEEE 488.2 defines data structures, commands, and common bit definitions for status reporting on the interface. There are also instrument-defined structures and bits.

The bits in the status byte act as summary bits for the data structures residing behind them. In the case of queues, the summary bit is set if the queue is not empty. For registers, the summary bit is set if any enabled bit in the event register is set. The events are enabled with the corresponding event enable register. Events captured by an event register remain set until the register is read or cleared. Registers are read with their associated commands. The *CLS 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.

Figure 6-1



Status Reporting Data Structures

Bit Definitions

MAV - message available. Indicates whether there is a response in the output queue.

ESB - event status bit. Indicates if any of the conditions in the Standard Event Status Register are set and enabled.

MSS - master summary status. Indicates whether the device has a reason for requesting service. This bit is returned for the *STB? query.

RQS - request service. Indicates if the device is requesting service. This bit is returned during a serial poll. RQS is set to 0 after it is read via a serial poll (MSS is not reset by *STB?).

URQ - user request. Indicates whether a front-panel key has been pressed.

CME - command error. Indicates whether the parser detected an error.

EXE - execution error. Indicates whether a parameter was out of range, or inconsistent with the current settings.

DDE - device specific error. Indicates whether the device was unable to complete an operation for device dependent reasons.

QYE - query error. Indicates whether the protocol for queries has been violated.

RQC - request control. Indicates whether the device is requesting control. The logic analyzer never requests control.

OPC - operation complete. Indicates whether the device has completed all pending operations.

TRG - trigger. Indicates whether a trigger has been received.

Operation Complete (*OPC)

The IEEE 488.2 structure provides one technique which can be used to find out if any operation is finished. The *OPC command, when sent to the instrument after the operation of interest, sets the OPC bit in the Standard Event Status Register when all pending device operations have finished. If the OPC bit and the RQS bit have been enabled, a service request is generated.

```
OUTPUT 707;"*SRE 32 ; *ESE 1"       !enables OPC service request
OUTPUT 707;":DIG CHAN1 ; *OPC"      !initiates data acquisition,
                                    !and
                                    !generates a SRQ when the
                                    !acquisition is complete
```

Trigger Bit (TER)

The Trigger (TER) bit indicates if the device has received a trigger. The TRG event register will stay set after receiving a trigger until it is cleared by reading it or using the *CLS command. If your application needs to detect multiple triggers, the TER 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.

```
OUTPUT 707;"*SRE 32"   ! enables event status register.
                       ! the next trigger will generate an SRQ.
OUTPUT 707;"*ESE 2"    ! enables event status register
OUTPUT 707;":TER?"     ! queries the TRG event register, thus
ENTER 707;A$           ! clearing it.
                       ! the next trigger can now generate an
                       ! SRQ
```

Status Byte

If the device is requesting service (RQS set), and the controller serial polls the device, the RQS bit is cleared. The MSS bit (read with *STB?) is not cleared by reading it. The status byte is not cleared when read, except for the RQS bit.

Serial Poll

This oscilloscope supports the IEEE 488.1 serial poll feature. When a serial poll of the instrument is requested, the RQS bit is returned on bit 6 of the status byte.

Using Serial Poll

The service request can used by conducting a serial poll of all instruments on the bus. For this procedure, assume that there are two instruments on the bus: an oscilloscope at address 7 and a printer at address 1. It is assumed that you are operating on Interface Select Code 7.

The program command for serial poll using Agilent BASIC is Stat=SPOLL(707). The address 707 is the address of the oscilloscope in this example. The command for checking the printer is Stat=SPOLL(701) because the address of that instrument is 01 on bus address 7. This command reads the contents of the GPIB Status Register into the variable called Stat. At that time bit 6 of the variable Stat can be tested to see if it is set (bit 6=1).

The serial poll operation can be conducted in the following manner:

1. Enable interrupts on the bus. This allows the controller to “see” the SRQ line.
2. If the SRQ line is high (some instrument is requesting service) then check the instrument at address 1 to see if bit 6 of its status register is high.
3. Disable interrupts on the bus.
4. To check whether bit 6 of an instrument’s status register is high, use the following command line.

   ```
   IF BIT (Stat, 6) then
   ```

5. If bit 6 of the instrument at address 1 is not high, then check the instrument at address 7 to see if bit 6 of its status register is high.
6. As soon as the instrument with status bit 6 high is found, check the rest of the status bits to determine what is required.

The SPOLL(707) command causes much more to happen on the bus than simply reading the register. This command clears the bus, automatically addresses the talker and listener, sends SPE (serial poll enable) and SPD (serial poll disable) bus commands, and reads the data. For more information about serial poll, refer to your controller manual and programming language reference manuals.

After the serial poll is completed, the RQS bit in the oscilloscope Status Byte Register is reset if it was set. Once a bit in the Status Byte Register is set, it remains set until the status is cleared with a *CLS command, or the instrument is reset. If these bits do not get reset, they cannot generate another SRQ.

Installing and Using the Programmer's Reference

Installing and Using the Programmer's Reference

The *Programmer's Reference* is supplied as an online help file readable with the Microsoft Windows help viewer. Sample programs for the oscilloscopes are included in the Examples subdirectory.

This chapter explains how to install the help file on your system, discusses the text and program files, and explains how you can get the programs and help file via the Internet.

To install the help file under Microsoft Windows

The help file requires Microsoft Windows 95/98/NT running on an IBM-compatible PC. The file uses the Microsoft Windows help viewer, WINHELP.EXE.

1. Insert the 3.5" floppy disk labeled "54600/01/02/03/10/15/16 Programmer's Reference with Example Programs" into the floppy disk drive of your PC.
2. Select **`Start`** | **`Run`** from the Program Manager, then type in the following:

   ```
   <drive>:\setup.exe
   ```

   where **`<drive>`** is your floppy disk drive letter.
3. Follow the instructions on-screen to complete the installation.

   The installer copies the help file to a directory named c:\Program Files\Agilent 54600-Series Programmer's Reference.

   You can choose a different directory if desired. It also creates a Program Manager group and icon that you can use to open the help file with the Microsoft Windows help viewer.

To get updated help and program files via the Internet

The latest versions of the help and example program files are available via the internet using your web browser or using ftp software.

- Log on to your Internet service.

Using your web browser

1 To connect using your web browser, type the following on the address line in your internet browser:

```
ftp://ftp.cos.agilent.com/dist/hp54600/hp54600
```

2 Check the README.TXT file for more information on the files in this directory.

Using an ftp application

1 To connect using an ftp application, a sample set of commands might be as follows:

```
$ ftp ftp.cos.agilent.com
Name: anonymous
Password: <your email address>
```

2 Change to the directory containing 54600 help files.

```
ftp> cd dist/hp54600/hp54600
```

3 Get sample programs or updated help files from the directory as desired.

For example, if you want the latest version of the Agilent 54600-Series Oscilloscopes Programmer's Reference online help file, you set the transfer mode to binary and get the file:

```
ftp> binary
ftp> get 54600.hlp
```

4 Check the README.TXT file for more information on the files in this directory.

If you have trouble making the connection, or need more information on ftp, see your network administrator.

To start the help file

- To open the help file under Microsoft Windows, double-click the "Programmer's Reference" icon in the "Agilent 54600-Series Oscilloscopes Programmer's Reference" program group in the Program Manager.

  The help file requires the program WINHELP.EXE for Microsoft Windows95/98/NT. The properties for the Program Manager icon are set to expect this file in the Windows directory.

To navigate through the help file

- Navigate through the help file by clicking on highlighted text and buttons.

  See your Microsoft Windows documentation for more information, or select `Help | How to Use Help` in the Help window.

Introduction

The Programmer's Quick Reference provides the commands and queries with their corresponding arguments and returned formats for the Agilent 54600-Series Oscilloscopes. The arguments for each command list the minimum argument required. The part of the command or query listed in uppercase letters refers to the short form of that command or query. The long form is the combination of the uppercase and lowercase letters. Any optional parameters are listed at the end of each parameter listing.

This quick reference lists commands for the following Agilent oscilloscope models:

54600
54601
54602
54603
54610
54615
54616

Conventions

The following conventions used in this guide include:

< >	Indicates that words or characters enclosed in angular brackets symbolize a program code parameter or an GPIB command.
::= "is defined as."	<A>::= <B> indicates that <A> can be replaced by <B> in any statement containing <A>.
| "or"	Indicates a choice of one element from a list. For example, <A> | <B> indicates <A> or <B> but not both.
...	Indicates that the element preceding the ellipses may be repeated one or more times.
[ ]	Indicates that the bracketed items are optional.
{ }	Indicates that when items are enclosed by braces, one, and only one of the elements may be selected.
{N,..,P}	Indicates selection of one integer between N and P inclusive.

Suffix Multipliers

The following suffix multipliers are available for arguments.

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

For more information regarding specific commands or queries, please refer to the online *Agilent 54600-Series Oscilloscopes Programmer's Reference*.

Commands and Queries

The following tables facilitate easy access to each command and query for the Agilent 54600-Series Oscilloscopes. The commands and queries are divided into separate categories with each entry alphabetized.

The arguments for each command list the minimum argument required. The part of the command or query listed in uppercase letters refers to the short form of that command or query. The long form is the combination of the uppercase and lowercase letters.

These commands also show specific information about how the command operates on a particular oscilloscope model. For additional information, refer to the online Agilent 54600-Series Oscilloscopes Programmer's Reference.

Command	Query	Options and Query Returns
:ACQuire:COMPlete <complete_argument>	:ACQuire:COMPlete?	<complete_argument> ::= 0 to 100; an integer in NR1 format
:ACQuire:COUNt <count_argument>	:ACQuire:COUNT?	<count_argument> ::= 8, 64, or 256; an integer in NR1 format
n/a	**:ACQuire:POINts?**	For all models except 54615/16: 1 to 4000; an integer in NR1 format. For the 54615/16: 1 to 5000; an integer in NR1 format.
n/a	**:ACQuire:SETup?**	ACQuire:TYPE{NORM | AVER | PEAK}; COUNt<count_argument>; (8, 64, or 256; an integer in NR1 format); POINts<points_argument>; For all models except 54615/16: 1 to 4000; an integer in NR1 format. For the 54615/16: 1 to 5000; an integer in NR1 format. COMPlete<complete_argument> 0 to 100; an integer in NR1 format
:ACQuire:TYPE <acq_type>	:ACQuire:TYPE?	<acq_type> ::= {NORMal | AVERage | PEAK}
:ASTore	n/a	n/a
:AUToscale	n/a	n/a
:BLANk <display>	n/a	<display> ::= {CHAN <n> | PMEM{1 | 2}} for the 54600/01/02/03/15/16 {CHAN <n> | PMEM{1 | 2} | EXTernal} for the 54610 <n> ::= 1 or 2; an integer in NR1 format for the 54600/03/10/15/16 1, 2, 3, or 4; an integer in NR1 format for the 54601/02
:CHANnel<n>:BWLimit {ON | OFF}	:CHANnel<n>:BWLimit?	{ON | OFF} <n> ::= 1 or 2; an integer in NR1 format
:CHANnel<n>:COUPling {AC | DC | GND}	:CHANnel<n>:COUPling?	{AC | DC | GND} <n> ::= 1 or 2; an integer in NR1 format for 54600/03/10/15/16 1, 2, 3 or 4; an integer in NR1 format for the 54601/02
:CHANnel<n>:INPut {FIFTy | ONEMeg}	:CHANnel<n>:INPut?	{FIFTy | ONEMeg} <n> ::= 1 or 2; an integer in NR1 format
:CHANnel<n>:INVert {ON | OFF}	:CHANnel<n>:INVert?	{ON | OFF} <n> ::= 1 or 2; an integer in NR1 format
:CHANnel:MATH {OFF | PLUS | SUBTract}	:CHANnel:MATH?	{OFF | PLUS | SUBTract}
:CHANnel<n>:OFFSet <offset_argument>	:CHANnel<n>:OFFSet?	<offset_argument> ::= offset value in volts in <NR3> format. <n> ::= 1 or 2; an integer in NR1 format for 54600/03/10/15/16 1, 2, 3 or 4; an integer in NR1 format for the 54601/02
:CHANnel<n>:PMODe {AUTo | MANual}	:CHANnel<n>:PMODe?	{AUT | MAN} <n> ::= 1 or 2; an integer in NR1 format

Command	Query	Options and Query Returns
:CHANnel<n>:PROBe <attenuation>	:CHANnel<n>:PROBe?	<attenuation> ::= X1, X10, X100 for 51600/01/02/03 X1, X10, X20, X100 for the 54610/15/16 <n> ::= 1 or 2; an integer in NR1 format for 54600/03/10/15/16 1, 2, 3 or 4; an integer in NR1 format for the 54601/02
:CHANnel<n>:PROTect {OFF | ON}	:CHANnel<n>:PROTect?	{OFF | ON} <n> ::= 1 or 2; an integer in NR1 format
:CHANnel<n>:RANGe <range_argument>	:CHANnel<n>:RANGe?	<range_argument> ::= Full-scale range value for channels 1 or 2 in NR3 format, and {LOW | HIGH} for channels 3 or 4.
n/a	**:CHANnel<n>:SETup?**	For 54600/01/02/03 channels 1 and 2: CHANnel<n>:RANGe <range>; OFFSet <offset>; COUPling {AC | DC | GND}; BWLimit {ON | OFF}; INVert {ON | OFF}; VERNier {ON | OFF}; PROBe {X1 | X10 | X100} For 54610/15/16 channel 1: CHANnel1:RANGe <range>; OFFSet <offset>; COUPling {AC | DC | GND}; BWLimit {ON | OFF}; INVert {ON | OFF}; VERNier {ON | OFF}; PROBe {X1 | X10 | X20 | X100}; PMODe {AUT | MAN}; INPut {FIFTy | ONEMeg}; PROTect {OFF | ON} For 54610/15/16 channel 2: CHANnel2:RANGe <range>; OFFSet <offset>; COUPling {AC | DC | GND}; BWLimit {ON | OFF}; INVert {ON | OFF}; VERNier {ON | OFF}; PROBe {X1 | X10 | X20 | X100}; PMODe {AUT | MAN}; INPut {FIFTy | ONEMeg}; PROTect {OFF | ON}; SKEW <skew_value> For 54601/02 channels 3 or 4: CHANnel<n>:RANGe {HIGH | LOW}; OFFSet <offset>; COUPling {DC | GND}; PROBe {X1 | X10 | X100}
:CHANnel2:SKEW <skew_argument>	:CHANnel2:SKEW?	<skew_argument> ::= the skew value in seconds in <NR3> format
:CHANnel<n>:VERNier {ON | OFF}	:CHANnel<n>:VERNier?	{ON | OFF}
***CLS**	n/a	n/a
:DIGitize CHANnel<n>, [CHANnel<n>]	n/a	n/a
:DITHer {ON | OFF}	:DITHer?	{ON | OFF}
:DISPLAY:COLumn <number>	:DISPLAY:COLumn?	<number> ::= 0 through 63; an integer in NR1 format
:DISPlay:CONNect {ON | OFF}	:DISPlay:CONNect?	{ON | OFF}
:DISPlay:DATA <binary block_data>	:DISPlay:DATA?	<binary block_data> ::= 16256 bytes of data in IEEE 488.2 # format

Command	Query	Options and Query Returns
:DISPlay:GRID {ON | OFF | SIMPle | TV}	:DISPlay:GRID?	{ON | OFF | SIMPle | TV}
:DISPlay:INVerse {ON | OFF}	:DISPlay:INVerse?	{ON | OFF}
:DISPlay:LINE <string>	n/a	<string> ::= any series of ASCII characters enclosed in quotation marks
:DISPlay:PALette <palette_number>	:DISPlay:PALette?	<palette_number> ::= 0 through 6; an integer in NR1 format
:DISPlay:PIXel <x>, <y>, <intensity>	:DISPlay:PIXel? <x>,<y>	For 54616C: <x> ::= x coordinate of the pixel to be set; an integer (0 to 500) in NR1 format <y> ::= y coordinate of the pixel to be set; an integer (0 to 275) in NR1 format <intensity> (comand) ::= an integer in NR1 format: 0 to clear pixel 1 to light pixel in autostore plane 2 to light pixel in graticule plane <intensity> (query) ::= an integer in NR1 format: 0 for pixel off 1 for autostore and any text on 2 for any waveforms on 3 for autostore and any text on, and any waveform on For all other models: <x> ::= x coordinate of the pixel to be set; an integer (0 to 511) in NR1 format <y> ::= y coordinate of the pixel to be set; an integer (0 to 303) in NR1 format <intensity> (command) ::= an integer in NR1 format: 0 to clear pixel 1 for half-bright 2 for full-bright other value to clear pixel <intensity> (query) ::= an integer in NR1 format: 0 for pixel off 1 for pixel with half-bright on 2 for pixel with full-bright on 3 for pixel with both half-bright and full-bright on
:DISPlay:ROW <row number>	:DISPlay:ROW?	<row number> ::= 1...20; an integer in NR1 format

Command	Query	Options and Query Returns
n/a	**:DISPlay:SETup?**	:DISPlay:ROW <row_number>; <row_number> ::= 1...20; an integer in NR1 format COLumn <column_number>; <column_number> ::= 0...63; an integer in NR1 format INVerse <inverse>; <inverse> ::= {ON | OFF} GRID <grid>; <grid> ::= {ON | OFF} SOURce <source>; <source> ::= {PMEMory1 | PMEMory2} CONNect <connect_status> <connect_status> ::= {ON | OFF} PALette <palette_number> (54616C only) <palette_number> ::= 0...6; an integer in NR1 format
:DISPlay:SOURce <value>	:DISPlay:SOURce?	<value> ::= {PMEMory1 | PMEMory2}
:DISPlay:TEXT BLANk	n/a	n/a
:ERASe <value>	n/a	<value> ::= {PMEMory1 | PMEMory2}
***ESE** <mask_argument>	*ESE?	<mask_argument> ::= 0...255; an integer in NR1 format Bit Weight Enables 7 128 NOT USED 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 TRG - Trigger Query 0 1 OPC - Operation Complete
n/a	***ESR?**	<status> ::= 0...255; an integer in NR1 format
:EXTernal:COUPling {DC | AC | GND}	:EXTernal:COUPling?	{DC | AC | GND}
:EXTernal:INPut {FIFTy | ONEMEG}	:EXTernal:INPut?	{FIFTy | ONEMEG}
:EXTernal:OFFSet <offset_argument>	:EXTernal:OFFSet?	<offset_argument> ::= offset value in volts in NR3 format
:EXTernal:PMODe {AUTo | MANual}	:EXTernal:PMODe?	{AUTo | MANual}
:EXTernal:PROBe <attenuation>	:EXTernal:PROBe?	<attenuation> ::= {X1 | X10 | X20 | X100} for the 54610/15/16
:EXTernal:PROTect {OFF | ON}	:EXTernal:PROTect?	{OFF | ON}

Command	Query	Options and Query Returns
n/a	**:EXTernal:SETup?**	For the 54610: EXTernal:OFFSet <offset_value>; COUPling {DC | AC | GND}; PROBe {X1 | X10 | X20 | X100}; PMODe {AUTo | MANual}; INPut {FIFTy | ONEMeg}; PROTect {OFF | ON}; SKEW <skew_value> For the 54615/16: EXTernal:COUPling {DC | AC | GND}; PROBe {X1 | X10 | X20 | X100}; PMODe {AUTo | MANual}; INPut {FIFTy | ONEMeg}; PROTect {OFF | ON}
:EXTernal:SKEW <skew_value>	:EXTernal:SKEW?	<skew_value> ::= external trigger skew value in seconds in NR3 format
:FUNCtion2:CENTer <frequency>	:FUNCtion2:CENTer?	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from 0 Hz to 10.00 GHz.
:FUNCtion2:MOVE {LEFT)	n/a	n/a
:FUNCtion<N>:OFFSet <offset>	:FUNCtion<N>:OFFSet?	<offset> ::= the value at center screen in NR3 format. The range of legal values is +-10 times the current sensitivity of the selected function. <N> ::= 1 or 2
:FUNCtion<N>:OPERation <operation>	:FUNCtion<N>:OPERation?	<operation> ::= {ADD | SUBTract | MULTiply} for :FUNCtion1:OPERation {INTegrate | DIFFerentiate | FFT} for :FUNCtion2:OPERation <N> ::= 1 or 2
n/a	**:FUNCtion2:PEAKs?** {FREQ1 | DB1 | FREQ2 | DB2}	<measurement> ::= {FREQ1 | DB1 | FREQ2 | DB2}. The measurement is the value of the peak specified in NR3 format.
:FUNCtion<N>:RANGe <range>	:FUNCtion<N>:RANGe?	<range> ::= the full-scale vertical axis value in NR3 format. The range for FUNCtion1 is 8E-6 to 8E+6. The range for the INTegrate function is 8E-9 to 400E+3. The range for the DIFFerentiate function is 8E-6 to 1.6E11. The range for the FFT function is 8 to 400 dB/div. <N> ::= 1 or 2
:FUNCtion2:REFerence <level>	:FUNCtion2:REFerence?	<level> ::= the current reference level in NR3 format. The range of legal values is from -160.0 dBV to +240.0 dBV in increments of 2.5 dBV.
:FUNCtion2:SOURce {CHANnel1 | CHANnel2 | FUNCtion1}	:FUNCtion2:SOURce?	{CHANnel1 | CHANnel2 | FUNCtion1}. The current reference level value is in NR3 format. The range of legal values is from -160.0 dBV to +240.0 dBV in increments of 2.5 dBV.
:FUNCtion2:SPAN 	:FUNCtion2:SPAN?	::= the current frequency span in NR3 format. Legal values are 1.221 Hz to 9.766 Ghz
:FUNCtion<N>:VIEW {ON | OFF}	:FUNCtion<N>:VIEW?	{ON | OFF} <N> ::= 1 or 2
:FUNCtion2:WINDow {RECTangular | HANNing | FLATtop | EXPonent}	n/a	{RECTangular | HANNing | FLATtop | EXPonent}

Command	Query	Options and Query Returns
n/a	***IDN?**	HEWLETT-PACKARD,<model>, 0, X.X <model> ::= the model number of the instrument <X.X> ::= the software revision of the instrument
n/a	***LRN?**	<learn_string> ::= a maximum of 218 bytes of data in IEEE 488.2 # format
:MASK:CREATe	n/a	n/a
:MASK:DATA	:MASK:DATA?	<header> ::= block header that contains the ASCII characters #8000998 and is sent prior to the data. <mask_data> ::= 998 bytes of data that represent the currently selected mask template.
:MASK:DESTination {TRACe | PRINter}	:MASK:DESTination?	{TRACe | PRINter}
:MASK:FAILmode {IN | OUT}	:MASK:FAILmode?	{IN | OUT}
:MASK:INCRement {ON | OFF}	:MASK:INCRement?	{ON | OFF}
:MASK:NUMBer <number>	:MASK:NUMBer?	<number> ::= memory (1 or 2)
:MASK:POSTfailure {RUN | STOP}	:MASK:POSTfailure?	{RUN | STOP}
:MASK:SAVE {ON | OFF}	:MASK:SAVE?	{ON | OFF}
n/a	**:MASK:STATistics?**	<compares, failures, failure %> ::= current number of mask tests performed number of failures detected percentage of failures
:MASK:TEST {ON | OFF}	:MASK:TEST?	{ON | OFF}
:MASK:TOLerance <value>	:MASK:TOLerance?	<value> ::= the tolerance used when creating a mask template. The entered value can be from 0.00 to 20.0 percent.
n/a	**:MEASure:ALL?**	<value list> ::= <FREQ result>, <PERIOD result>, <+ WID result>, <- WID result>, <RISE result>, <FALL result>, <VPP result>, <DUTY CYCLE result>, <VRMS result>, <VMAX result>, <VMIN result>, <VTOP result>, <VBASE result>, <VAVG result>, <VAMP result>, <Vovershoot result>, <Vpreshoot result> <result> ::= individual measurement results in NR3 format
:MEASure:DEFine DELay <edge1>,<edge2>	:MEASure:DEFine? DELay	<edgeN> ::= the edge selection for channels 1 and 2. N is the selected edge number (1 to 5).
:MEASure:DELay	:MEASure:DELay?	<return_value> ::= floating point number delay time in seconds in NR3 format
:MEASure:DUTYcycle	:MEASure: DUTYcycle?	<return_value> ::= ratio of positive pulse width to period in NR3 format

Command	Query	Options and Query Returns
:MEASure:FALLtime	:MEASure:FALLtime?	<return_value> ::= time in seconds between the 10% and 90% voltage levels in NR3 format
:MEASure:FREQuency	:MEASure:FREQuency?	<return_value> ::= frequency in Hertz in NR3 format
:MEASure:LOWer <voltage>	:MEASure:LOWer?	<voltage> ::= the user-defined lower threshold in volts in NR3 format
:MEASure:NWIDth	:MEASure:NWIDth?	<return_value> ::= negative pulse width in seconds in NR3 format
:MEASure:OVERshoot	:MEASure:OVERshoot?	<voltage> ::= the percent of the overshoot of the selected waveform in NR3 format
:MEASure:PERiod	:MEASure:PERiod?	<return_value> ::= waveform period in seconds in NR3 format
:MEASure:PHASe	:MEASure:PHASe?	<return_value> ::= the phase angle value in degrees in NR3 format
:MEASure:PREShoot	:MEASure:PREShoot?	<return_value> ::= the percent of preshoot of the selected waveform in NR3 format
:MEASure:PSTArt <value>	:MEASure:PSTArt?	<value> ::= the relative position of time marker 1 in degrees in NR3 format
:MEASure:PSTOp <value>	:MEASure:PSTOp?	<value> ::= the relative position of time marker 2 in degrees in NR3 format
:MEASure:PWIDth	:MEASure:PWIDth?	<return_value> ::= width of positive pulse in seconds in NR3 format
:MEASure:RISEtime	:MEASure: RISEtime?	<return_value> ::= rise time in seconds in NR3 format
:MEASure:SCRatch	n/a	n/a
:MEASure:SET100	n/a	n/a
:MEASure:SET360	n/a	n/a
:MEASure:SHOW {ON | OFF}	:MEASure:SHOW?	{ON | OFF}
:MEASure:SOURce CHANnel <n>	:MEASure:SOURce?	<n> ::= 1 or 2; an integer in NR1 format for 54600/03/10/15/16 1, 2, 3 or 4; an integer in NR1 format for the 54601/02
n/a	**:MEASure:TDELta?**	<return_value> ::= time difference in seconds between start and stop markers in NR3 format
:MEASure:THResholds {T1090 | T2080 | VOLTage}	:MEASure:THResholds?	{T1090 | T2080 | VOLTage}
:MEASure:TSTArt <value>	:MEASure:TSTArt?	<value> ::= time at the start marker in seconds in NR3 format
:MEASure:TSTOp <value>	:MEASure:TSTOp?	<value> ::= time at the stop marker in seconds in NR3 format
n/a	**:MEASure:TVOLt** <tvolt_argument>, <slope><occurrence>	<tvolt_argument> ::= positive or negative voltage level that the waveform must cross. <slope> ::= direction of the waveform when <tvolt_argument> is crossed. <occurrence> ::= number of crossings to be reported. <return_value> ::= time in seconds of specified voltage crossing in NR3 format

Command	Query	Options and Query Returns
:MEASure:UPPer <voltage>	:MEASure:UPPer?	<voltage> ::= the user-defined upper threshold in volts in NR3 format
:MEASure:VAMPlitude	:MEASure:VAMPlitude?	<return_value> ::= the amplitude of the selected waveform in volts in NR3 format
:MEASure:VAVerage	:MEASure:VAVerage?	<return_value> ::= calculated average voltage in NR3 format
:MEASure:VBASe	:MEASure:VBASe?	<base_voltage> ::= voltage at the base of the selected waveform in NR3 format
n/a	**:MEASure:VDELta?**	<return_value> ::= delta V value in volts in NR3 format
:MEASure:VMAX	:MEASure:VMAX?	<return_value> ::= maximum voltage of the selected waveform in NR3 format
:MEASure:VMIN	:MEASure:VMIN?	<return_value> ::= minimum voltage of the selected waveform in NR3 format
:MEASure:VPP	:MEASure:VPP?	<return_value> ::= voltage peak to peak in NR3 format
:MEASure:VPSTArt <value>	:MEASure:VPSTArt?	<value> ::= the relative position of voltage marker 1 in percent in NR3 format
:MEASure:VPSTOp <value>	:MEASure:VPSTOp?	<value> ::= the relative position of voltage marker 2 in percent in NR3 format
:MEASure:VRMS	:MEASure:VRMS?	<return_value> ::= calculated dc RMS voltage in NR3 format
:MEASURE:VSTArt <vstart_argument>	:MEASure:VSTArt?	<vstart_argument> ::= voltage value for VMarker 1 in NR3 format <return_value> ::= voltage at VMarker 1 in NR3 format
:MEASure:VSTOp <vstop_argument>	:MEASure:VSTOp?	<vstop_argument> ::= voltage value for VMarker 2 in NR3 format <return_value> ::= voltage at VMarker 2 in NR3 format
n/a	**:MEASure:VTIMe** <vtime_argument>	<vtime_argument> ::= displayed time from trigger in seconds in NR3 format <return_value> ::= voltage at the specified time in NR3 format
:MEASure:VTOP	:MEASure:VTOP?	<return_value> ::= voltage at the top of the waveform in NR3 format

Command	Query	Options and Query Returns
:MENU <integer>	:MENU?	<integer> ::= the following: Menu — Number No menu selected — 0 Channel 1 — 1 Channel 2 — 2 Channel 3 (54601/02) — 3 External Trigger (54610/15/16) — 3 Channel 4 (54601/02) — 4 Math — 5 Trigger source — 6 Trigger mode — 7 Trigger slope — 8 Main/delayed (horizontal) — 9 Time measurements — 10 Voltage measurements — 11 Cursors — 12 Trace — 13 Setup — 14 Display — 15 Utility/Print — 16
:MERGe <pixel memory>	n/a	<pixel memory> ::= {PMEMory1 | PMEMory2}
***OPC**	*OPC?	ASCII "1" is placed in the output queue when all pending device operations have completed.
n/a	***OPT?**	n identifies the module and option pairing. X.X identifies the module software revision. Module: — No Opt. 005 — With Opt. 005 Basic Interface — 0,X.X — 50,X.X Test Automation — 1,X.X — 51,X.X Measurement/Storage — 2,X.X — 52,X.X
n/a	**:PRINt?** [enhancement]	[enhancement] ::= [HIRes [,PCLColor]] HIRes ::= contains both half-bright and full-bright display information PCLColor ::= color DeskJet selection only on 54616C
***RCL** <value>	n/a	<value> ::= {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 }
***RST**	n/a	See reset values in the online Programmer's Reference.
:RUN	n/a	n/a
***SAV** <value>	n/a	<value> ::= {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 }
:SEQuence:NEXT	n/a	n/a
:SEQuence:PREVious	n/a	n/a
:SEQuence:PROTect {ON | OFF}	:SEQuence:PROTect?	<protect> ::= {ON | OFF}, which reports the status of the protection.
:SEQuence:RESet	n/a	n/a

Command	Query	Options and Query Returns
:SEQuence:SETup {MASK | STEP}, <number>, <header> <setup_string>	:SEQuence:SETup? {MASK | STEP}, <number>	MASK ::= an individual mask sent to the setup string. STEP ::= an individual step sent to the setup string. <number> ::= the mask number or step number sent to the setup string. <header> ::= the type of setup to be sent or returned: For individual masks, ::= #800001000. For individual steps, ::= #800000244. For whole sequences, ::= #800064122. <setup_string> ::= the setup string to be sent: For individual masks, := 1000-byte string. For individual steps, ::= 244-byte string. For whole sequences, ::= 64122-byte string.
:SEQuence:STEP <number>	:SEQuence:STEP?	<number> ::= an integer from 1 to 100 in NR1 format
n/a	**:SEQuence:TEST?**	<result> ::= an integer from 0 to 3 in NR1 format: 0 = pass 1 = fail minimum limit line 2 = fail maximum limit line 3 = fail both minimum and maximum limit lines
***SRE** <mask>	*SRE?	<mask> ::= sum of all bits that are set, 0,...,255; an integer in NR1 format. <mask> ::= following values: Bit 7 Weight 128 Enables: Not Used Bit 6 Weight 64 Enables: RQS - Request Service Bit 5 Weight 32 Enables: ESB - Event Status Bit Bit 4 Weight 16 Enables: MAV - Message Available Bit 3 Weight 8 Enables: Not used Bit 2 Weight 4 Enables: Not used Bit 1 Weight 2 Enables: Not used Bit 0 Weight 1 Enables: Not used
n/a	**:STATus?** <display>	{ON | OFF} <display> ::= {CHANnel <n> | PMEMory{1 | 2}} for the 54600/01/02/03/15/16 {CHANnel <n> | PMEMory{1 | 2} | EXTernal} for the 54610 <n> ::= 1 or 2; an integer in NR1 format for the 54600/03/10/15/16 1, 2, 3, or 4; an integer in NR1 format for the 54601/02
n/a	***STB?**	<value> ::= 0,...,255; an integer in NR1 format, as shown in the following: Bit 7 Weight 128 Name ---- Condition: NOT USED Bit 6 Weight 64 Name RQS/MS Condition: 0 = instrument has no reason for service; 1= instrument is requesting service Bit 5 Weight 32 Name ESB Condition: 0 = no event status conditions occurred; 1 = enabled event status condition occurred Bit 4 Weight 16 Name MAV Condition: 0 = no output messages are ready; 1 = an output message is ready Bit 3 Weight 8 Name ---- Condition: 0 = not used Bit 2 Weight 4 Name ---- Condition: 0 = not used Bit 1 Weight 2 Name ---- Condition: 0 = not used Bit 0 Weight 1 Name ---- Condition: 0 = not used

Command	Query	Options and Query Returns
:STOP	n/a	n/a
:SYSTem:DSP <string>	n/a	<string> ::= quoted ASCII string
n/a	**:SYSTem:ERRor?**	<error> ::= an integer error code See error values in the online Programmer's Reference.
:SYSTem:KEY <key_code>	:SYSTem:KEY?	<key_code> ::= -1 to 16, or 19 to 50; an integer See key code values in the online Programmer's Reference.
:SYSTem:LOCK <value>	:SYSTem:LOCK?	<value> ::= {ON | OFF}
:SYSTem:SETup <setup_data>	:SYSTem:SETup?	<setup_data> ::= a maximum of 218 bytes of data in IEEE 488.2 # format.
n/a	**:TER?**	<return_value> ::= 0 or 1
:TIMebase:DELay <delay_value>	:TIMebase:DELay?	<delay_value> ::= time from trigger to display reference in seconds. The display reference is left or center in NR3 format.
:TIMebase:MODE <value>	:TIMebase:MODE?	<value> ::= {NORMal | DELayed | XY | ROLL}
:TIMebase:RANGe <range_value>	:TIMebase:RANGe?	<range_value> ::= the following values in NR3 format: 50 ns through 50 s for 54603 20 ns through 50 s for 54600/01/02 10 ns through 50 s for 54610/15/16
:TIMebase:REFerence {LEFT | CENTer}	:TIMebase:REFerence?	<return_value> ::= {LEFT | CENTer} for Normal or Delayed modes. <return_value> ::= {CENTer | RIGHt} for ROLL mode.
n/a	**:TIMebase:SETup?**	For all models except the 54615/16: TIMebase:MODE {NORM | DEL | XY};RANGe <range>; DELay <delay>;REF {LEFT | CENT};VERN {ON | OFF} For the 54615/16: TIMebase:MODE {NORM | DEL | XY};RANGe <range>; DELay <delay>;REF {LEFT | CENT} <range> ::= the following values in NR3 format: 50 ns through 50 s for 54603 20 ns through 50 s for 54600/01/02 10 ns through 50 s for 54610/15/16 <delay> ::= time from trigger to delay reference in seconds in NR3 format
:TIMebase:VERNier {ON | OFF}	:TIMebase:VERNier?	{ON | OFF}
:TRACe:CLEAR <N>	n/a	<N> ::= the trace memory number (1 to 100)
:TRACe:DATA <N>,<trace_data>	:TRACe:DATA? <N>	<N> ::= the trace memory number (1 to 100). <header> ::= the 10-byte block header that contains the ASCII characters #8000nnnnn and is sent prior to the data. (nnnnn is the number of bytes in the data string.) <trace_data> ::= a maximum of 16,342 bytes of data, setup, and label information that represents the current trace.

Command	Query	Options and Query Returns
:TRACe:MODE <N> {ON | OFF}	:TRACe:MODE? <N>	<N> ::= 1 to 100 <return_state> ::= {ON | OFF}
:TRACe:SAVE <N>	n/a	<N> ::= the trace memory number (1 to 100).
***TRG**	n/a	n/a
:TRIGger:COUPling {AC | DC}	:TRIGger:COUPling?	{AC | DC}
:TRIGger:FIELd {ALTernate | ONE | TWO | VERTical}	:TRIGger:FIELd?	{ALTernate | ONE | TWO | VERTical}
:TRIGger:HOLDoff <holdoff_time>	:TRIGger:HOLDoff?	<holdoff_time> ::= the holdoff time value in seconds in NR3 format.
:TRIGger:LEVel <level_argument>	:TRIGger:LEVel?	<return_value> ::= the trigger level in volts in NR3 format.
:TRIGger:LINE <line_number>	:TRIGger:LINE?	<line_number> ::= integer in NR1 format.
:TRIGger:MODE {AUTLevel | AUTO | NORMal | SINGle | TV}	:TRIGger:MODE?	{AUTLevel | AUTO | NORMal | SINGle | TV}
:TRIGger:NREJect {OFF | ON}	:TRIGger:NREJect?	{OFF | ON}
:TRIGger:OPTMode {LINE | FIELD1 | FIELD2 | VERTical | ALLLINES | ALLFLDS}	:TRIGger:OPTMode?	{LINE | FIELD1 | FIELD2 | VERTical | ALLLINES | ALLFLDS}
:TRIGger:POLarity {POSitive | NEGative}	:TRIGger:POLarity?	{POS | NEG}
:TRIGger:REJect {OFF | LF | HF}	:TRIGger:REJect?	{OFF | LF | HF}
n/a	:TRIGger:SETup?	TRIG:MODE {AUTL | AUTO | NORM | SING | TV}; SOURCE <source>; LEVEL <level>; HOLD <time>; SLOPE {POS | NEG}; COUP {AC | DC}; REJ {OFF | LF | HF}; NREJ {ON | OFF}; POL {POS | NEG}; TVMODE <tvmode>; TVHF {ON | OFF} <level> ::= trigger level in volts in NR3 format <time> ::= holdoff time value in seconds in NR3 format <source> ::= {CHAN{1 | 2} | EXT | LINE} for 54600/03/10/15/16 {CHAN{1 | 2 | 3 | 4} | LINE} for 54601/02 <tvmode>::= {FIELD1 | FIELD2 | LINE} for 54600/01/03 {FIELD1 | FIELD2 | LINE | VERT} for 54602/10/15/16
:TRIGger:SLOPe {NEGative | POSitive}	:TRIGger:SLOPe?	{NEG | POS}

Command	Query	Options and Query Returns
:TRIGger:SOURce <source>	:TRIGger:SOURce?	<source> ::= {CHANnel1 | CHANnel2 | EXTernal | LINE} for 54600/03/10/15/16 {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4} for 54601/02
:TRIGger:STANdard {GENeric | NTSC | PAL | PALM | SECam}	:TRIGger:STANdard?	{GENeric | NTSC | PAL | SECam}
:TRIGger:TVHFrej {OFF | ON}	:TRIGger:TVHFrej?	{OFF | ON}
:TRIGger:TVMode <mode>	:TRIGger:TVMode?	<mode> ::= {LINE | FIELD1 | FIELD2} for 54600/01/03/15/16 {LINE | FIELD1 | FIELD2 | VERTical} for 54602/10
:TRIGger:VIR {ON | OFF}	:TRIGger:VIR?	{ON | OFF}
n/a	***TST?**	<result> ::= 0 or non-zero value; an integer in NR1 format 0 indicates the test passed. Non-zero indicates the test failed.
:VAUToscale	n/a	n/a
:VIEW <display>	n/a	<display> ::= {CHANnel <n> | PMEMory{1 | 2}} for the 54600/01/02/03/15/16 {CHANnel <n> | PMEMory{1 | 2} | EXTernal} for the 54610 <n> ::= 1 or 2; an integer in NR1 format for the 54600/03/10/15/16 1, 2, 3, or 4; an integer in NR1 format for the 54601/02
***WAI**	n/a	n/a
:WAVeform:BYTeorder <value>	:WAVeform:BYTeorder?	<value> ::= {LSBFirst | MSBFirst}
:WAVeform:DATA <binary block data in # format>	:WAVeform:DATA?	<binary block length bytes>, <binary data> For example, to transmit 4000 bytes of data, the syntax would be: #800004000<4000 bytes of data><NL> 8 is the number of digits that follow 00004000 is the number of bytes to be transmitted <4000 bytes of data> is the actual data
:WAVeform:FORMat <value>	:WAVeform:FORMat?	<value> ::= {ASC | WORD | BYTE}
:WAVeform:POINts <value>	:WAVeform:POINts?	<value> ::= integer {100 | 200 | 250 | 400 | 500 | 800 | 1000 | 2000 | 4000 | 5000} in NR1 format

Command	Query	Options and Query Returns
n/a	**:WAVeform:PREamble?**	<preamble_block> ::= <format NR1>, <type NR1>, <points NR1>, <count NR1>, <xincrement NR3>, <xorigin NR3>, <xreference NR1>, <yincrement NR3>, <yorigin NR3>, <yreference NR1> <format> ::= an integer in NR1 format: 0 for ASCii format 1 for BYTE format 2 for WORD format <type> ::= an integer in NR1 format: 0 for AVERage type 1 for NORMal type 2 for PEAK detect type <count> ::= an integer in NR1 format: 1, always 1 and is present for compatibility
:WAVeform:SOURce CHANnel <n>	:WAVeform:SOURce?	<n> ::= {1 | 2} for the 54600/03/10/15/16 {1 | 2 | 3 | 4} for the 54601/02
n/a	**:WAVeform:TYPE?**	<return_mode> ::= {NORMal | PEAK | AVERage}
n/a	**:WAVeform:XINCrement?**	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	**:WAVeform:XORigin?**	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	**:WAVeform:XREFerence?**	<return_value> ::= x-reference value in the current preamble in NR1 format
n/a	**:WAVeform:YINCrement?**	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	**:WAVeform:YORigin?**	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	**:WAVeform:YREFerence?**	<return_value>::= y-reference value in the current preamble in NR1 format

Microsoft is a registered trademark of Microsoft Corporation.

Restricted Rights Legend.

Use, duplication or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS

252.227-7013 for DOD agencies, and subparagraphs (c) (1) and (c) (2) of the Commercial Computer Software Restricted Rights clause at FAR 52.227-19 for other agencies.

Agilent Technologies

3000 Hanover Street Palo Alto, California 94304 U.S.A.

Document Warranty

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

Agilent Technologies makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability or fitness for a particular purpose.

Agilent Technologies shall not be liable for errors contained herein or for damages in connection with the furnishing, performance, or use of this material.

Safety

This apparatus has been designed and tested in accordance with IEC Publication 1010, Safety Requirements for Measuring Apparatus, and has been supplied in a safe condition. This is a Safety Class I instrument (provided with terminal for protective earthing). Before applying power, verify that the correct safety precautions are taken (see the following warnings). In addition, note the external markings on the instrument that are described under "Safety Symbols."

Warning

- Before turning on the instrument, you must connect the protective earth terminal of the instrument to the protective conductor of the (mains) power cord. The mains plug shall only be inserted in a socket outlet provided with a protective earth contact. You must not negate the protective action by using an extension cord (power cable) without a protective conductor (grounding). Grounding one conductor of a two-conductor outlet is not sufficient protection.
- Only fuses with the required rated current, voltage, and specified type (normal blow, time delay, etc.) should be used. Do not use repaired fuses or short-circuited fuseholders. To do so could cause a shock or fire hazard.
- Service instructions are for trained service personnel. To avoid dangerous electric shock, do not perform any service unless qualified to do so. Do not attempt internal service or adjustment unless another person, capable of rendering first aid and resuscitation, is present.
- If you energize this instrument by an auto transformer (for voltage reduction), make sure the common terminal is connected to the earth terminal of the power source.
- Whenever it is likely that the ground protection is impaired, you must make the instrument inoperative and secure it against any unintended operation.
- Do not operate the instrument in the presence of flammable gasses or fumes. Operation of any electrical instrument in such an environment constitutes a definite safety hazard.
- Do not install substitute parts or perform any unauthorized modification to the instrument.
- Capacitors inside the instrument may retain a charge even if the instrument is disconnected from its source of supply.
- Use caution when exposing or handling the CRT. Handling or replacing the CRT shall be done only by qualified maintenance personnel.

Safety Symbols

Instruction manual symbol: the product is marked with this symbol when it is necessary for you to refer to the instruction manual in order to protect against damage to the product.

Hazardous voltage symbol.

Earth terminal symbol: Used to indicate a circuit common connected to grounded chassis.

WARNING

The Warning sign denotes a hazard. It calls attention to a procedure, practice, or the like, which, if not correctly performed or adhered to, could result in personal injury. Do not proceed beyond a Warning sign until the indicated conditions are fully understood and met.

CAUTION

The Caution sign denotes a hazard. It calls attention to an operating procedure, practice, or the like, which, if not correctly performed or adhered to, could result in damage to or destruction of part or all of the product. Do not proceed beyond a Caution symbol until the indicated conditions are fully understood or met.

Agilent Technologies
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901

Product Warranty

This Agilent Technologies product has a warranty against defects in material and workmanship for a period of three years from date of shipment. During the warranty period, Agilent Technologies will, at its option, either repair or replace products that prove to be defective.

For warranty service or repair, this product must be returned to a service facility designated by Agilent Technologies.

For products returned to Agilent Technologies for warranty service, the Buyer shall prepay shipping charges to Agilent Technologies and Agilent Technologies shall pay shipping charges to return the product to the Buyer. However, the Buyer shall pay all shipping charges, duties, and taxes for products returned to Agilent Technologies from another country.

Agilent Technologies warrants that its software and firmware designated by Agilent Technologies for use with an instrument will execute its programming instructions when properly installed on that instrument. Agilent Technologies does not warrant that the operation of the instrument software, or firmware will be uninterrupted or error free.

Limitation of Warranty

The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.

No other warranty is expressed or implied. Agilent Technologies specifically disclaims the implied warranties of merchantability or fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are the buyer's sole and exclusive remedies. Agilent Technologies shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Agilent Technologies products.

For any assistance, contact your nearest Agilent Technologies Sales Office.

Certification

Agilent Technologies certifies that this product met its published specifications at the time of shipment from the factory. Agilent Technologies further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.

About this edition

This is the first edition of the *Agilent 54600-Series Oscilloscopes Programmer's Guide.*

Publication number
54600-97032, April 2001

Printed in USA.

Print history is as follows:

54600-97032, April 2001

54600-97017, July 1996

54600-97015, October 1996

54600-97014, April 1995

New editions are complete revisions of the manual. Many product updates do not require manual changes; and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.