Programmer's Guide

Publication Number 54622-97038
September 2002

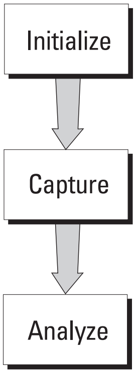
For Safety information see the pages behind the Index.

Agilent 54621A/22A/24A/41A/42A Oscilloscopes and Agilent 54621D/22D/41D/42D Mixed-Signal Oscilloscopes

Programming the Oscilloscope

When you attach an interface module to the rear of the oscilloscope, it becomes programmable. That is, you can hook a controller (such as a PC or workstation) to it, and write programs on that controller to automate oscilloscope setup and data capture.

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 set up and ready for data transfer.
- Oscilloscope initialization sets the channel configuration and labels, threshold voltages, trigger specification and mode, timebase, and acquisition type.

Capture

Once you initialize the oscilloscope, you can begin capturing data for analysis. 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 will most likely be rendered.

To collect data, you use the :DIGitize command. This command clears the waveform buffers and starts the acquisition process. Acquisition continues until acquisition memory is full, then stops. The acquired data is displayed by the oscilloscope, and the captured data can be measured, stored in trace 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 put the oscilloscope into run mode, then use a wait loop in your program to ensure that the oscilloscope has completed at least one acquisition before you make a measurement. Agilent does not recommend this 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 on 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 frequency, duty cycle, period, and positive and negative pulse width.

Using the :WAVeform commands, you can transfer the data to your controller. You may want to display the data, compare it to a known good measurement, or simply check logic patterns at various time intervals in the acquisition.

In This Book

This *Programmer's Guide* is your introduction to programming the oscilloscope using an instrument controller. This book, with the *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.

The oscilloscope has a built-in RS-232-C port for programming. To program the oscilloscope over GPIB, you need the N2757A GPIB Interface Module. 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.

This book contains the following information:

Chapter 1 Introduction to Programming, gives a general overview of oscilloscope programming.

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

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

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

Chapter 5 Programming and Documentation Conventions, describes the conventions used in representing the syntax of commands throughout this book and the Programmer's Reference, and gives an overview of the oscilloscope command set.

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

Chapter 7 Installing and Using the Programmer's Reference, tells how to install the Programmer's Reference online help file in Microsoft Windows, and explains help file navigation.

Chapter 8 Programmer's Quick Reference, lists all the commands and queries available for programming the oscilloscope.

For information on oscilloscope operation, see the *User's Guide*. For information on interface configuration, see the documentation for the oscilloscope interface module and the interface card used in your controller (for example, the Agilent 82350A interface for IBM PC-compatible computers).

Contents

1 Introduction to Programming

2 Programming Getting Started

3 Programming over GPIB

4 Programming over RS-232-C

1

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 IEEE488.2 Standard Digital Interface for Programmable Instrumentation. The programming instructions provide the means of remote control.

To program the oscilloscope you must add either a GPIB (N2757A) interface, or program over the built-in RS-232-C interface on the rear panel.

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

- Set up the instrument.
- Make measurements.
- Acquire data (waveform, measurements, configuration) from the oscilloscope.
- Send information (pixel images, 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 HPBASIC 6.3 or 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, HPBASIC 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 HP BASIC statement sends a command which turns on label display.

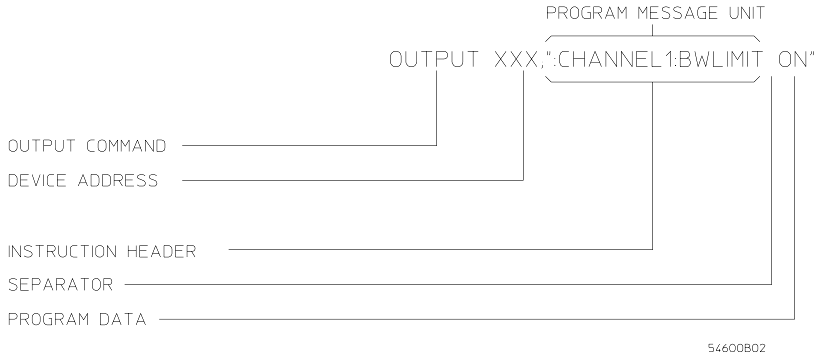
```
OUTPUT < device address > ;":CHANNEL1:BWLIMIT ON"<terminator>
```

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 following figure 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, HPBASIC is used in most examples of individual commands. If you are using other languages, you will need to find the equivalents of HP BASIC commands like OUTPUT, ENTER, and CLEAR 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 HP 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 <learn string>. There are only a few instructions that 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 1-1 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 space characters. 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 CHANNEL1), 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, separate the functions with a semicolon (;):

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

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

Duplicate Mnemonics

Identical function mnemonics can be used in 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 HP 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 HP 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 clears the output buffer and the current response. It also generates a query interrupted error in the error queue.

Program Header Options

You can send program headers 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 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 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, :CHANNEL:THRESHOLD POD1,TTL has two program data: POD1 and TTL.

Two main types of program data are used in commands: character and numeric.

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. See the online Programmer's Reference for more information. When sending commands, you may either the long form or short form (if one exists). Uppercase and lowercase letters may be mixed freely. When receiving query responses, uppercase 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 be ignored, truncating the number. Numeric data parameters accept fractional values are called real numbers.

All numbers must 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 handled 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.

2

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 manual are written in HPBASIC 6.3 or C.

Initialization

To make sure the bus and all appropriate interfaces are in a known state, begin every program with an initialization statement. HP 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 *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 for 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. An example of the commands that might be sent to the oscilloscope are:

```
:CHANNEL1:PROBE 10;RANGE 16;OFFSET 1.00<terminator>
:TIMEBASE:MODE NORMAL;RANGE 1E-3;DELAY 100E-6<terminator>
```

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

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 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 10"          ! 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:SWEEP 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  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 ms/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.

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

When you change the oscilloscope configuration, the waveform buffers are cleared. Before doing a measurement, send the DIGitize command 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 command 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 *Programmer's Reference*.

Aborting a Digitize Operation Over GPIB

When using GPIB, you can abort a digitize operation 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 HP 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 HP BASIC, this is usually done with an ENTER statement.

Sending another command before reading the result of the query clears the output buffer and the current response. 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 HP BASIC 6.3, 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 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:

```
+40.0E-00
```

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:

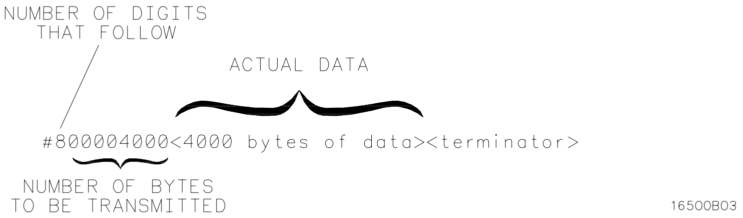
```
40
```

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:

Figure 2-1


NUMBER OF DIGITS THAT FOLLOW
ACTUAL DATA
#800004000<4000 bytes of data><terminator>
NUMBER OF BYTES TO BE TRANSMITTED
16500B03

Definite-length block response data

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 and then display them:

```
ENTER 707;Result1,Result2
PRINT 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.

3

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.

The optional Agilent N2757A GPIB Interface Module must be connected to the oscilloscope to allow programming over GPIB.

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

To set up the GPIB interface (optional Agilent N2757A GPIB Interface Module must be connected to the oscilloscope), refer to the “To set up the I/O port to use a controller” topic in the Utilities chapter of the User’s Guide.

- 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

Because 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 707.

Oscilloscope Device Address

The examples in this manual and in the online 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

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 (GTL) command 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 you send either of these commands 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.

4

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 interface. 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 subsequent sections of this chapter. 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. Because 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 because 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

Use the controller mode when you operate the instrument with a controller over RS-232-C. To set up the RS-232-C interface on the oscilloscope, refer to the “To set up the I/O port to use a controller” topic in the Utilities chapter of the User’s Guide.

Make sure that the RS-232-C settings on your oscilloscope match the settings of your COM1 or COM2 port on your pc.

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: 9600, 19,200, 38,400, or 57,600
- Stop Bits: preset to 1
- Parity: preset to None
- Protocol: DTR or XON/XOFF
- Data Bits: preset to 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.

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.

5

Programming and Documentation Conventions

Programming and Documentation Conventions

This chapter covers conventions used in programming the instrument, as well as conventions used in the online *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 *Programmer's Reference*, which is supplied as an online help file for Microsoft Windows. See chapter 7, "Installing and Using the Programmer's Reference" for information on installing and using the help file.

The commands shown use 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 command syntax. Some commands have a programming example.

The subsystems are listed below:

Subsystem	Description
ACQuire	sets the parameters for acquiring and storing data
CALibrate	provides utility commands for determining the state of the calibration factor protection switch
CHANnel<n>	controls all oscilloscope functions associated with individual analog channels or groups of channels
Common	commands defined by IEEE 488.2 standard common to all instruments
DIGital	controls all oscilloscope functions associated with individual digital channels
DISPlay	controls how waveforms, graticule, and text are displayed and written on the screen
EXTernal	controls the input characteristics of the external trigger input
FUNCtion	controls functions in the Measurement/Storage Module
HARDcopy	provides commands to set and query the selection of hardcopy device and formatting options
MARKer	provides commands to set and query the settings of X-axis markers (X1 and X2 cursors) and the Y-axis markers (Y1 and Y2 cursors).
MEASure	selects automatic measurements to be made and controls time markers
POD	controls all oscilloscope functions associated with groups of digital channels.
Root	controls many of the basic functions of the oscilloscope and reside at the root of the command tree
SYStem	controls some basic functions of the oscilloscope
TIMebase	controls all horizontal sweep functions
TRIGger	controls the trigger modes and parameters for each trigger type
WAVeform	provides access to waveform data

Table 5-1

Alphabetic Command Reference

Command	Subsystem Where used
ACKNowledge	TRIGger:CAN
ACTivity	CHANnel<n>
ACTivity	Root level
ADDRess	TRIGger:IIC:PATTern
AER	Root level
AUToscale	Root level
BAUDrate	TRIGger:CAN:SIGNal
BAUDrate	TRIGger:LIN:SIGNal
BLANk	Root level
BWLimit	CHANnel<n>
BWLimit	EXTernal
BYTeorder	WAVeform
CDISplay	Root
CENTer	FUNCtion
CLEAr	DISPlay
CLEAr	MEASure
CLOCk	TRIGger:IIC:SOURce
CLOCk	TRIGger:SPI
CLOCk	TRIGger:SPI:SOURce
*CLS	Common
COMPlete	ACQuire
CONNect	DISPlay
COUNt	TRIGger:SEQuence
COUNt	WAVeform
COUNter	MEASure
COUPling	CHANnel
COUPling	TRIGger:EDGE
DATA	DISPlay
DATA	TRIGger:CAN:PATTern
DATA	TRIGger:IIC:PATTern
DATA	TRIGger:IIC:SOURce
DATA	TRIGger:SPI:PATTERN
DATA	TRIGger:SPI:SOURce
DATA	WAVeform
DATE	CALibrate
DATE	SYSTem
DEFine	MEASure
DEFinition	TRIGger:CAN:SIGNal
DEFinition	TRIGger:LIN:SIGNal
DELay	MEASure
DELay	TIMebase
DESTinatin	HARDcopy
DEVice	HARDcopy
DIGitize	Root level
DISPlay	CHANnel<n>
DISPlay	DIGital
DISPlay	FUNCtion
DISPlay	POD
*DMC	Common
DMINus	TRIGger:USB:SOURce
DPLus	TRIGger:USB:SOURce
DSP	SYSTem
DUTycycle	MEASure
DURation	TRIGger:GLITch
EDGE	TRIGger
EDGE	TRIGger:SEQuence
*EMC	Common
ERASe	Root level
ERRor	SYSTem
*ESE	Common
*ESR	Common
FACTors	HARDcopy
FALLtime	MEASure
FFEed	HARDcopy
FIND	TRIGger:SEQuence
FORMat	HARDcopy
FORMat	WAVeform
FRAMe	TRIGger:SPI:SOURce
FRAMing	TRIGger:SPI
FREQuency	MEASure
*GMC	Common
GRAYscale	HARDcopy
GREaterthan	TRIGger:DURation
GREaterthan	TRIGger:GLITch
HFReject	TRIGger
HOLDoff	TRIGger
ID	TRIGger:CAN:PATTern
ID	TRIGger:LIN:SIGNal
*IDN	Common
IMPedance	CHANnel<n>
IMPedance	EXTernal
INPut	CHANnel<n>
INVert	CHANnel<n>
LABel	CALibrate
LABel	CHANnel<n>
LABel	DIGital
LABel	DISPLay
LABList	DISPlay
LENGth	TRIGger:CAN:PATTern:DATA
LESSthan	TRIGger:DURation
LESSthan	TRIGger:GLITch
LEVel	TRIGger:EDGE
LEVel	TRIGger:GLITch
LINE	TRIGger:TV
*LMC	Common
LOCK	SYSTem
*LRN	Common
MERGe	Root level
MODE	ACQuire
MODE	TRIGger:CAN:PATTern:ID
MODE	MARKer
MODE	TIMebase
MODE	TRIGger
MODE	TRIGger:TV
NREJect	TRIGger
NWIDth	MEASure
OFFSet	CHANnel<n>
OFFSet	FUNCtion
*OPC	Common
OPEE	Root level
OPER	Root level
*OPT	Common
ORDer	DISPlay
OVERshoot	MEASure
OVLenable	Root level
OVLRegister	Root level
PATTern	TRIGger
PATTern	TRIGger:CAN
PATTern	TRIGger:DURation
PATTern	TRIGger:SEQuence
PATTern	TRIGger:SPI
PERiod	MEASure
PERSistence	DISPlay
PHASe	MEASure

Command	Subsystem Where used
*PMC	Common
PMODe	CHANnel<n>
POINts	ACQuire
POINts	WAVeform
POLarity	TRIGger:TV
POLarity	TRIGger:GLITch
POSition	DIGital
POSition	TIMebase
POSition	TIMebase:WINDow
PREamble	WAVeform
PREShoot	MEASure
PRINt	Root level
PROBe	CHANnel<n>
PROBe	EXTernal
PROTection	CHANnel<n>
PROTection	EXTernal
PWIDth	MEASure
QUALifier	TRIGger:DURation
QUALifier	TRIGger:GLITch
QUALifier	TRIGger:IIC:TRIGer
RANGe	CHANnel<n>
RANGe	EXTernal
RANGe	FUNCtion
RANGe	TIMebase
RANGe	TIMebase:WINDow
RANGe	TRIGger:DURAtion
RANGe	TRIGger:GLITch
*RCL	Common
REFerence	FUNCtion
REFerence	TIMebase
REJect	TRIGger:EDGE
RESet	TRIGger:SEQuence
RISetime	MEASure
*RST	Common
RUN	Root level
SAMPlepoint	TRIGger:CAN
*SAV	Common
SCALe	CHANnel<n>
SCALe	FUNCtion
SCALe	TIMebase
SCALe	TIMebase:WINDow
SCRatch	MEASure
SERial	Root level
SETup	SYSTem
SHOW	MEASure
SIGNal	TRIGger:CAN
SIGnal	TRIGger:LIN
SINGle	Root level
SKEW	CHANnel<n>:PROBe
SLOPe	TRIGger:EDGE
SLOPe	TRIGger:SPI:CLOCk
SOUrce	DISPlay
SOURce	FUNCtion
SOURce	MEASure
SOURce	TRIGger:CAN
SOURce	TRIGger:GLITch
SOURce	TRIGger:IIC
SOURce	TRIGger:LIN
SOURce	TRIGger:SPI
SOURce	TRIGger:TV
SOURce	TRIGger:USB
SOURce	WAVeform
SPAN	FUNCtion
SPEed	TRIGger:USB
SRATe	ACQuire
*SRE	Common
STANdard	TRIGger:TV
STATus	Root level
*STB	Common
STOP	Root level
SWEep	TRIGger
SWITch	CALibrate
TEDGe	MEASure
TER	Root level
THReshold	CHANnel
THReshold	DIGital
THReshold	POD
THReshold	TRIGger
TIMer	TRIGger:SEQuence
TIMeout	TRIGger:SPI:CLOCk
*TRG	Common
TRIGger	TRIGger:CAN
TRIGger	TRIGger:IIC
TRIGger	TRIGger:LIN
TRIGger	TRIGger:SEQuence
TRIGger	TRIGger:USB
*TST	Common
TVALue	MEASure
TVMode	TRIGger:TV
TVOLt	MEASure
TYPE	ACQuire
TYPE	WAVeform
TYPE	TRIGger:IIC:TRIGger
UNITs	CHANnel<n>
UNITs	EXTernal
UNSigned	WAVeform
VAMPlitude	MEASure
VAVerage	MEASure
VBASe	MEASure
VECTors	DISPlay
VIEW	FUNCtion
VIEW	Root level
VIEW	WAVeform
VMAX	MEASure
VMIN	MEASure
VPP	MEASure
VRMS	MEASure
VTIMe	MEASure
VTOP	MEASure
*WAI	Common
WIDth	TRIGger:SPI:PATTERN
WINDow	FUNCtion
X1Position	MARKer
X1Y1source	MARKer
X2Position	MARKer
X2Y2source	MARKer
XDELta	MARKer
XINCrement	WAVeform
XMAX	MEASure
XMIN	MEASure
XORigin	WAVeform
XREFerence	WAVeform
Y1Position	MARKer
X1Y1source	MARKer
Y2Position	MARKer
YDELta	MARKer
YINCrement	WAVeform
YORigin	WAVeform
YREFerence	WAVeform

The Command Tree

The command tree shows all of the commands and the relationships of the commands to each other. The IEEE 488.2 common commands are not listed as part of the command tree because 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 are in three categories:

- 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.

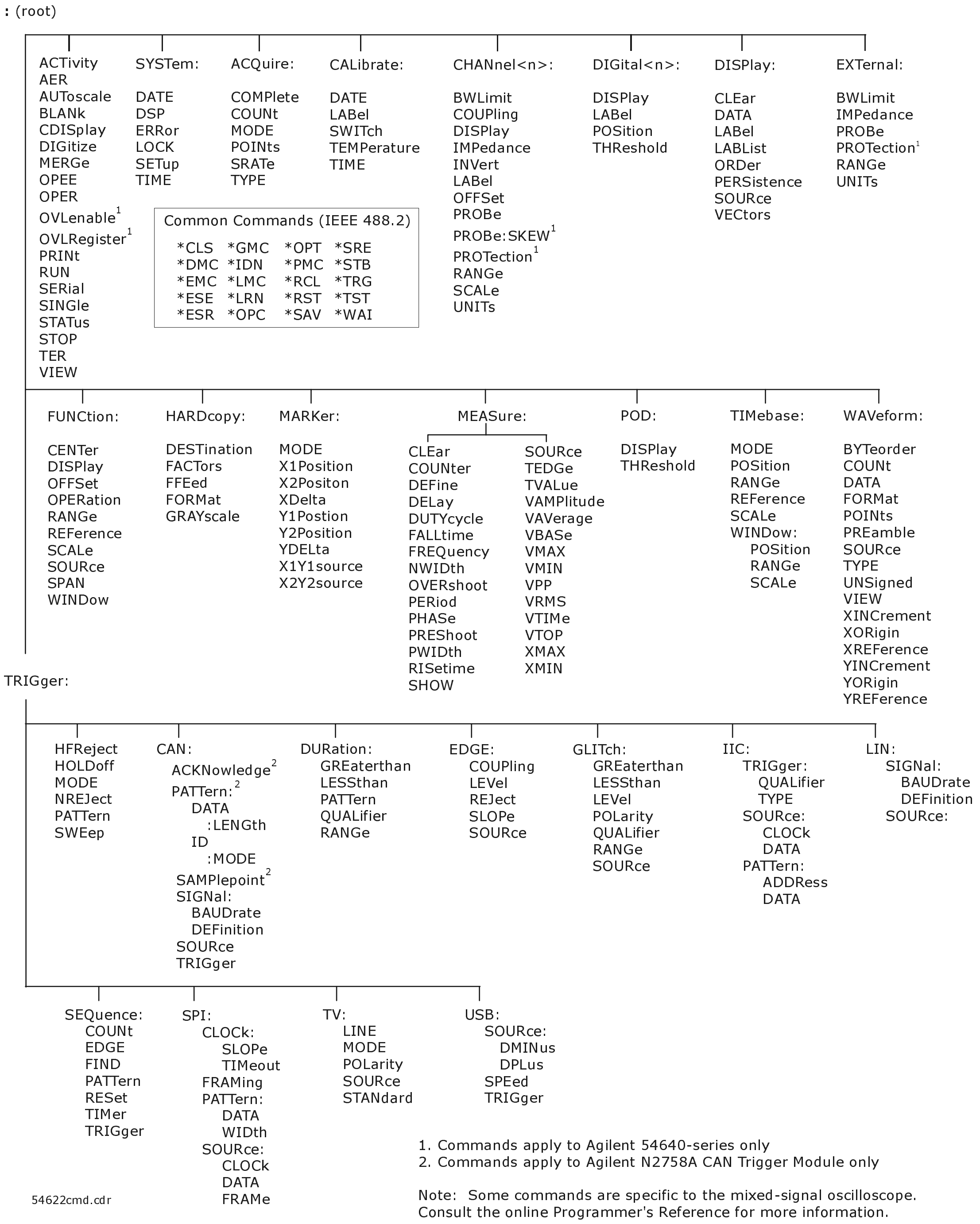
Example:

```
*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
```



: (root)

- ACTivity
- AER
- AUToscale
- BLANk
- CDISplay
- DIGitize
- MERGe
- OPEE
- OPER
- OVLenable[1]
- OVLRegister[1]
- PRINt
- RUN
- SERial
- SINGle
- STATus
- STOP
- TER
- VIEW

SYSTem:
- DATE
- DSP
- ERRor
- LOCK
- SETup
- TIME

ACQuire:
- COMPlete
- COUNt
- MODE
- POINts
- SRATe
- TYPE

CALibrate:
- DATE
- LABel
- SWITch
- TEMPerature
- TIME

CHANnel<n>:
- BWLimit
- COUPling
- DISPlay
- IMPedance
- INVert
- LABel
- OFFSet
- PROBe
- PROBe:SKEW[1]
- PROTection[1]
- RANGe
- SCALe
- UNITs

DIGital<n>:
- DISPlay
- LABel
- POSition
- THReshold

DISPlay:
- CLEar
- DATA
- LABel
- LABList
- ORDer
- PERSistence
- SOURce
- VECtors

EXTernal:
- BWLimit
- IMPedance
- PROBe
- PROTection[1]
- RANGe
- UNITs

Common Commands (IEEE 488.2)

```
*CLS  *GMC  *OPT  *SRE
*DMC  *IDN  *PMC  *STB
*EMC  *LMC  *RCL  *TRG
*ESE  *LRN  *RST  *TST
*ESR  *OPC  *SAV  *WAI
```

FUNCtion:
- CENTer
- DISPlay
- OFFSet
- OPERation
- RANGe
- REFerence
- SCALe
- SOURce
- SPAN
- WINDow

HARDcopy:
- DESTination
- FACTors
- FFEed
- FORMat
- GRAYscale

MARKer:
- MODE
- X1Position
- X2Positon
- XDelta
- Y1Postion
- Y2Position
- YDELta
- X1Y1source
- X2Y2source

MEASure:
- CLEar
- COUNter
- DEFine
- DELay
- DUTYcycle
- FALLtime
- FREQuency
- NWIDth
- OVERshoot
- PERiod
- PHASe
- PREShoot
- PWIDth
- RISetime
- SHOW
- SOURce
- TEDGe
- TVALue
- VAMPlitude
- VAVerage
- VBASe
- VMAX
- VMIN
- VPP
- VRMS
- VTIMe
- VTOP
- XMAX
- XMIN

POD:
- DISPlay
- THReshold

TIMebase:
- MODE
- POSition
- RANGe
- REFerence
- SCALe
- WINDow:
  - POSition
  - RANGe
  - SCALe

WAVeform:
- BYTeorder
- COUNt
- DATA
- FORMat
- POINts
- PREamble
- SOURce
- TYPE
- UNSigned
- VIEW
- XINCrement
- XORigin
- XREFerence
- YINCrement
- YORigin
- YREFerence

TRIGger:
- HFReject
- HOLDoff
- MODE
- NREJect
- PATTern
- SWEep

CAN:
- ACKNowledge[2]
- PATTern:[2]
  - DATA
    - :LENGth
  - ID
    - :MODE
- SAMPlepoint[2]
- SIGNal:
  - BAUDrate
  - DEFinition
- SOURce
- TRIGger

DURation:
- GREaterthan
- LESSthan
- PATTern
- QUALifier
- RANGe

EDGE:
- COUPling
- LEVel
- REJect
- SLOPe
- SOURce

GLITch:
- GREaterthan
- LESSthan
- LEVel
- POLarity
- QUALifier
- RANGe
- SOURce

IIC:
- TRIGger:
  - QUALifier
  - TYPE
- SOURce:
  - CLOCk
  - DATA
- PATTern:
  - ADDRess
  - DATA

LIN:
- SIGNal:
  - BAUDrate
  - DEFinition
- SOURce:

SEQuence:
- COUNt
- EDGE
- FIND
- PATTern
- RESet
- TIMer
- TRIGger

SPI:
- CLOCk:
  - SLOPe
  - TIMeout
- FRAMing
- PATTern:
  - DATA
  - WIDth
- SOURce:
  - CLOCk
  - DATA
  - FRAMe

TV:
- LINE
- MODE
- POLarity
- SOURce
- STANdard

USB:
- SOURce:
  - DMINus
  - DPLus
- SPEed
- TRIGger

54622cmd.cdr

1. Commands apply to Agilent 54640-series only
2. Commands apply to Agilent N2758A CAN Trigger Module only

Note: Some commands are specific to the mixed-signal oscilloscope. Consult the online Programmer's Reference for more information.

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 HPBASIC 6.3. 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"
```

or

```
OUTPUT 707;":TIMEBASE:REFERENCE CENTER; :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.

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.

Obsolete and Discontinued Commands

Core Commands

Core commands are a common set of commands that provide basic oscilloscope functionality on this oscilloscope and future Agilent 54600-series oscilloscopes. Core commands are unlikely to modified in the future. If you restrict your programs to core commands, the programs should work across product offerings in the future, assuming appropriate programming methods are employed.

Non-Core Commands

Non-core commands are commands that provide specific features, but are not universal across all oscilloscope models. Non-core commands may be modified or deleted in the future. With a command structure as complex as the 54620/40-series, some evolution over time is inevitable. Agilent's intent is to continue to expand command subsystems, such as the rich and evolving trigger feature set.

Obsolete Commands

Obsolete commands are older forms of commands that are provided to reduce customer rework for existing systems and programs. Generally, these commands are mapped onto some of the Core and Non-core commands, but may not strictly have the same behavior as the new command. None of the obsolete commands are guaranteed to functional in future products. New systems and programs should use the Core (and Non-core) commands.

Obsolete Commands

Obsolete Command	Current Command Equivalent	Behavior Differences
ANALog<n>:BWLimit	CHANnel<n>:BWLimit	
ANALog<n>:COUPling	CHANnel<n>:COUPling	
ANALog<n>:INVert	CHANnel<n>:INVert	
ANALog<n>:LABel	CHANnel<n>:LABel	
ANALog<n>:OFFSet	CHANnel<n>:OFFSet	
ANALog<n>:PROBe	CHANnel<n>:PROBe	
ANALog<n>:PMODe	none	
ANALog<n>:RANGe	CHANnel<n>:RANGe	
CHANnel:ACTivity	ACTivity	
CHANnel:LABel	CHANnel<n>:LABel or DIGital<n>:LABel	use CHANnel<n>:LABel for analog channels and use DIGital<n>:LABel for digital channels
CHANnel:THReshold	POD:THReshold or DIGital<n>:THReshold	
CHANnel2:SKEW	CHANnel<n>:PROBe:SKEW	
CHANnel<n>:INPut	CHANnel<n>:IMPedance	
CHANnel<n>:PMODe	none	
DISPlay:CONNect	DISPlay:VECTors	
DISPlay:ORDer	none	
ERASe	CDISplay	
EXTernal:INPut	EXTernal:IMPedance	
EXTernal:PMODe	none	
FUNCtion1, FUNCtion2	FUNCtion subsystem	ADD not included

Obsolete Command	Current Command Equivalent	Behavior Differences
FUNCtion:VIEW	FUNCtion:DISPlay	
HARDcopy:DEVice	HARDcopy:FORMat	PLOTter, THINkjet not supported; TIF, BMP, CSV, SEIko added
MEASure:LOWer	MEASure:DEFine:THResholds	MEASure:DEFine:THResholds can define absolute values or percentage from 1% to 99% for lower, middle, and upper thresholds on analog channels.
MEASure:SCRatch	MEASure:CLEar	
MEASure:TDELta	MARKer:TDELta	
MEASure:THResholds	MEASure:DEFine:THResholds	MEASure:DEFine:THResholds can define absolute values or percentage from 1% to 99% for lower, middle, and upper thresholds on analog channels.
MEASure:TMAX	MEASure:XMAX	
MEASure:TMIN	MEASure:XMIN	
MEASure:TSTArt	MARKer:X1Position	
MEASure:TSTOp	MARKer:X2Position	
MEASure:TVOLt	MEASure:TVALue	TVALue measures additional values such as db, Vs, etc.
MEASure:UPPer	MEASure:DEFine:THResholds	MEASure:DEFine:THResholds can define absolute values or percentage from 1% to 99% for lower, middle, and upper thresholds on analog channels.
MEASure:VDELta	MARKer:VDELta	
MEASure:VSTArt	MARKer:Y1Position	
MEASure:VSTOp	MARKer:Y2Position	
TIMebase:DELay	TIMebase:POSition or TIMebase:WINDow:POSition	TIMebase:POSition is position value of main time base; TIMebase:WINDow:POSition is position value of delayed time base window.
TRIGger:THReshold	POD:THREshold or DIGital<n>:THREshold	
TRIGger:TV:TVMode	TRIGger:TV:MODE	

Discontinued Commands

Discontinued commands are commands that were used by previous oscilloscopes, but are not supported by the 54620/40-series oscilloscopes. Listed below are the Discontinued commands and the nearest equivalent command available (if any).

Discontinued Commands

Discontinued Command	Current Command Equivalent	Comments
ASTore	DISPlay:PERSistence INFinite	
CHANnel:MATH	FUNCtion:OPERation	ADD not included
DISPlay:INVerse	none	
DISPlay:COLumn	none	
DISPlay:GRID	none	
DISPLay:LINE	none	
DISPlay:PIXel	none	
DISPlay:POSition	none	
DISPlay:ROW	none	
DISPlay:TEXT	none	
FUNCtion:MOVE	none	
FUNCtion:PEAKs	none	
HARDcopy:ADDRess	none	Only parallel printer port is supported. GPIB printing not supported
MASK	none	All commands discontinued, feature not available
SYSTem:KEY	none	
TEST:ALL	*TST	
TRACE subsystem	none	All commands discontinued, feature not available
TRIGger:ADVanced subsystem		Use new GLITch, PATTern or TV trigger modes
TRIGger:TV:FIELd	TRIGger:TV:MODE	
TRIGger:TV:TVHFrej	none	
TRIGger:TV:VIR	none	
VAUToscale	none	

Discontinued Parameters

Some previous oscilloscope queries returned control setting values of OFF and ON. The 54620/40-series oscilloscopes only return the enumerated values 0 (for off) and 1 (for on).

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.

Table 5-2 **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 *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 BASIC program examples given for commands in the online *Programmer's Reference* were written using the HPBASIC 6.3 programming language. The programs always assume the oscilloscope 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, placing the colon as the first character of the command is optional. The space between RANGE and the argument is required.

6

Status Reporting

Status Reporting

Figure 6-1 is an overview of the oscilloscope's status reporting structure. The status reporting structure allows monitoring specified events in the oscilloscope. The ability to monitor and report these events allows determination of such things as the status of an operation, the availability and reliability of the measured data, and more.

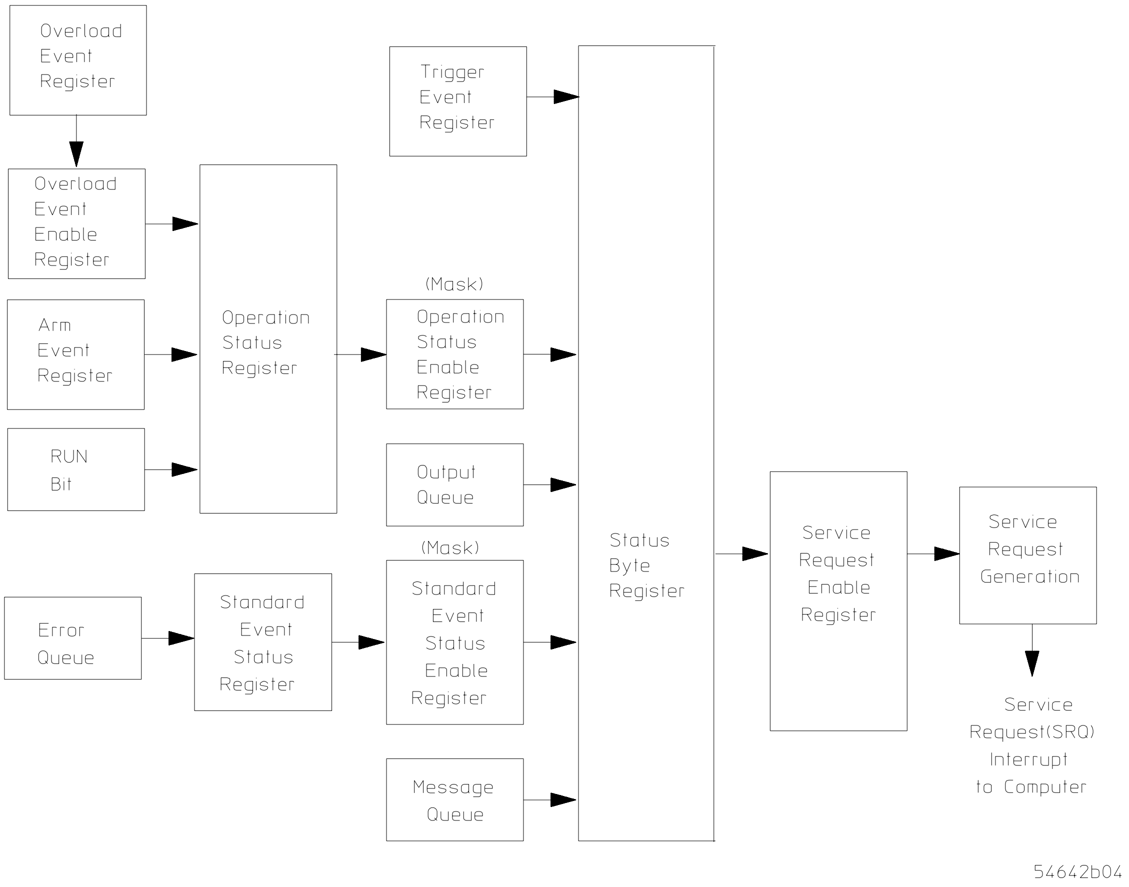
- To monitor an event, first clear the event, then enable the event. All of the events are cleared when you initialize the instrument.
- To generate a service request (SRQ) interrupt to an external controller, enable at least one bit in the Status Byte Register.

The Status Byte Register, the Standard Event Status Register group, and the Output Queue are defined as the Standard Status Data Structure Model in IEEE 488.2-1987.

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

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 you send *CLS is sent immediately after a program message terminator, the output queue is also cleared.

Figure 6-1


Overload Event Register
Overload Event Enable Register
Arm Event Register
RUN Bit
Operation Status Register
Trigger Event Register
(Mask)
Operation Status Enable Register
Output Queue
(Mask)
Error Queue
Standard Event Status Register
Standard Event Status Enable Register
Message Queue
Status Byte Register
Service Request Enable Register
Service Request Generation
Service Request(SRQ) Interrupt to Computer
54642b04

Status Reporting Overview Block Diagram

The status reporting structure consists of the registers in figure 6-1.

Table 6-1 is a list of the bit definitions for the bit in the status reporting data structure.

Table 6-1 **Status Reporting Bit Definition**

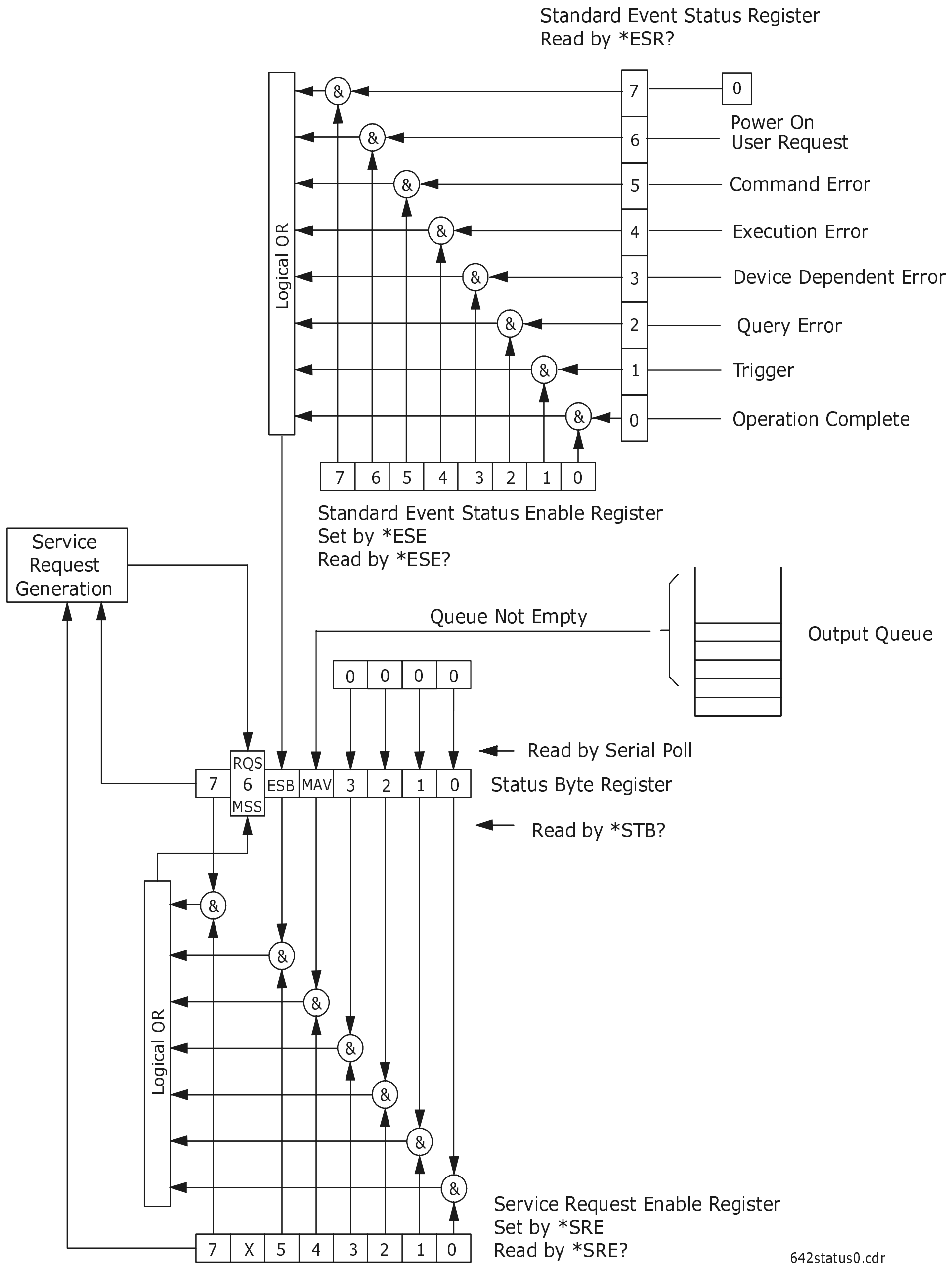
Bit	Description	Indicates
PON	Power On	Power is turned on.
URQ	User Request	Whether a front-panel key has been pressed.
CME	Command Error	Whether the parser detected an error.
EXE	Execution Error	Whether a parameter was out of range, or inconsistent with the current settings.
DDE	Device Dependent Error	Whether the device was unable to complete an operation for device dependent reasons.
QYE	Query Error	If the protocol for queries has been violated.
RQL	Request Control	Whether the device is requesting control.
OPC	Operation Complete	Whether the device has completed all pending operations.
OPER	Operation Status Register	If any of the enabled conditions in the Operation Status Register have occurred.
RQS	Request Service	That the device is requesting service.
MSS	Master Summary Status	Whether a device has a reason for requesting service.
ESB	Event Status Bit	If any of the enabled conditions in the Standard Event Status Register have occurred.
MAV	Message Available	If there is a response in the output queue.
MSG	Message	An advisory has been displayed.
TRG	Trigger	Whether a trigger has been received.
WAIT TRIG	Wait for Trigger	Instrument is armed and ready for trigger.
OVLR	Overload Event Register	If an overload is sensed on a 50Ω input.
RUN	Instrument running	That the instrument is not stopped

Status Reporting Data Structures

Figure 6-2 brings together the different status reporting data structures mentioned in this chapter and shows how they work together. To make it possible for any of the Standard Event Status Register bits to generate a summary bit, the bits must be enabled. These bits are enabled by using the *ESE common command to set the corresponding bit in the Standard Event Status Enable Register.

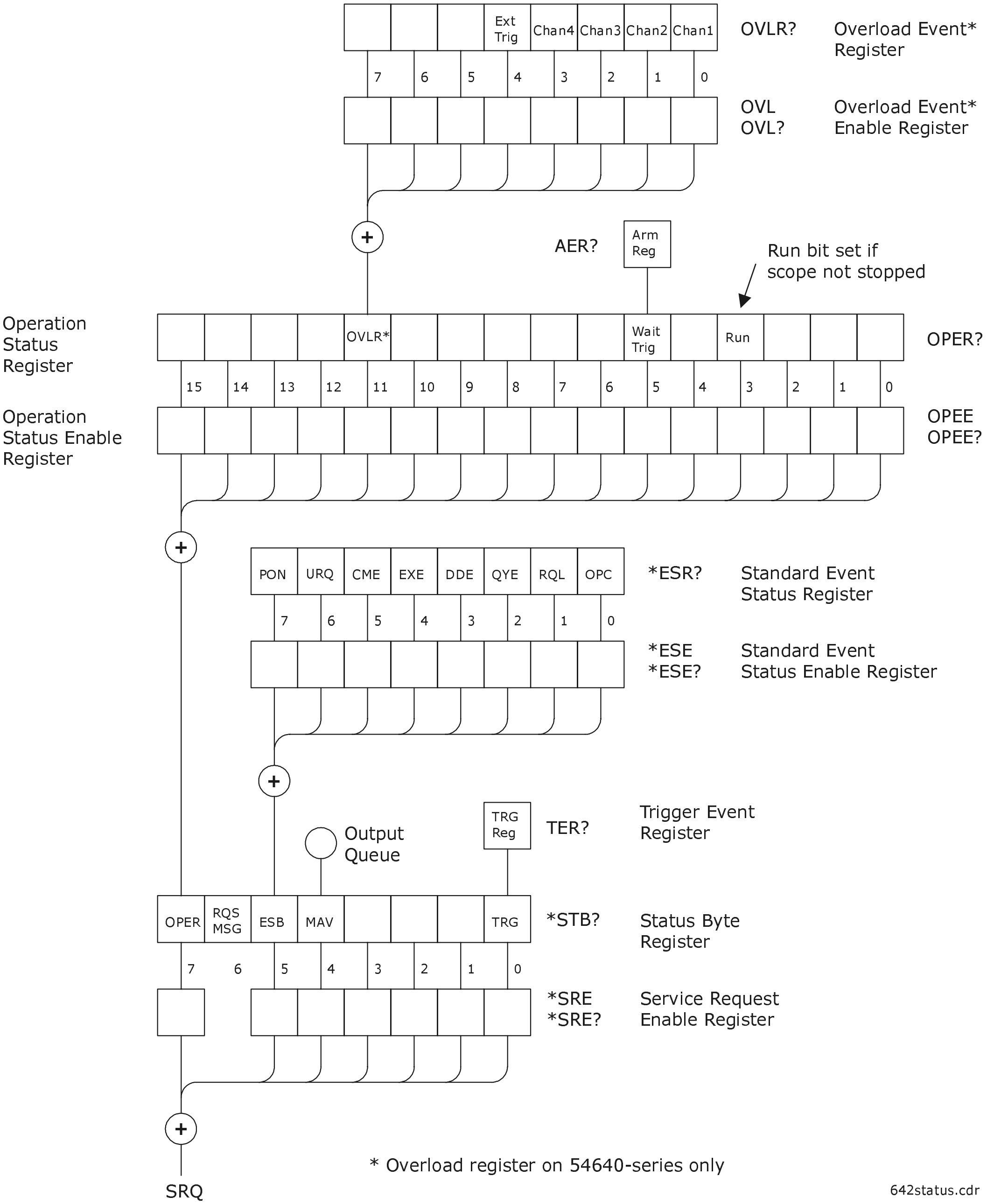
To generate a service request (SRQ) interrupt to an external controller, at least one bit in the Status Byte Register must be enabled. These bits are enabled by using the *SRE common command to set the corresponding bit in the Service Request Enable Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.

Figure 6-2



Status Reporting Data Structures

Figure 6-2 (continued)



Status Reporting Data Structures

Status Byte Register (SBR)

The Status Byte Register is the summary-level register in the status reporting structure. It contains summary bits that monitor activity in the other status registers and queues. The Status Byte Register is a live register. That is, its summary bits are set and cleared by the presence and absence of a summary bit from other event registers or queues.

If the Status Byte Register is to be used with the Service Request Enable Register to set bit 6 (RQS/MSS) and to generate an SRQ, at least one of the summary bits must be enabled, then set. Also, event bits in all other status registers must be specifically enabled to generate the summary bit that sets the associated summary bit in the Status Byte Register.

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

The use of bit 6 can be confusing. This bit was defined to cover all possible computer interfaces, including a computer that could not do a serial poll. The important point to remember is that, if you are using an SRQ interrupt to an external computer, the serial poll command clears bit 6. Clearing bit 6 allows the oscilloscope to generate another SRQ interrupt when another enabled event occurs.

No other bits in the Status Byte Register are cleared by either the *STB? query or the serial poll, except the Message Available bit (bit 4). If there are no other messages in the Output Queue, bit 4 (MAV) can be cleared as a result of reading the response to the *STB? command.

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

Example

The following example uses the *STB? query to read the contents of the oscilloscopes Status Byte Register.

```
10  OUTPUT 707;"*STB?"   !Query the Status Byte Register
20  ENTER 707;Result     !Place result in a numeric variable
30  PRINT Result         !Print the result
40  End
```

The next program prints 112 and clears bit 6 (RQS) of the Status Byte Register. The difference in the decimal value between this example and the previous one is the value of bit 6 (weight = 64). Bit 6 is set when the first enabled summary bit is set and is cleared when the Status Byte Register is read by the serial poll command.

Example

The following example uses the HP BASIC serial poll (SPOLL) command (GPIB only) to read the contents of the oscilloscopes Status Byte Register.

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

Use Serial Polling to Read Status Byte Register

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

Service Request Enable Register (SRER)

Setting the Service Request Enable Register bits enable corresponding bits in the Status Byte Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.

Bits are set in the Service Request Enable Register using the *SRE command and the bits that are set are read with the *SRE? query.

Refer to figure 6-2.

Example

The following example sets bit 4 (MAV) and bit 5 (ESB) in the Service Request Enable Register.

```
OUTPUT 707;"*SRE 48"
```

This example uses the parameter 48 to enable the oscilloscope to generate an SRQ interrupt under the following conditions:

- When one or more bytes in the Output Queue set bit 4 (MAV).
- When an enabled event in the Standard Event Status Register generates a summary bit that sets bit 5 (ESB).

Trigger Event Register (TRG)

This register sets the TRG bit in the status byte when a trigger event occurs.

The TRG event register stays set until it is cleared by reading the register or using the *CLS command. If your application needs to detect multiple triggers, the TRG event register must be cleared after each one.

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

Standard Event Status Register (SESR)

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

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

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

You can read the contents of the Standard Event Status Register and clear the register by sending the *ESR? query. The value returned is the total bit weights of all of the bits that are set at the present time.

Example

The following example uses the *ESR query to read the contents of the Standard Event Status Register.

```
10  OUTPUT 707;"*ESR?"
20  ENTER 707;Result      !Place result in a numeric variable
30  PRINT Result          !Print the result
40  End
```

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

Standard Event Status Enable Register (SESER)

To allow any of the Standard Event Status Register (SESR) bits to generate a summary bit, you must first enable that bit. Enable the bit by using the *ESE (Event Status Enable) common command to set the corresponding bit in the Standard Event Status Enable Register (SESER).

Set bits are read with the *ESE? query.

Example

Suppose your application requires an interrupt whenever any type of error occurs. The error related bits in the Standard Event Status Register are bits 2 through 5. The sum of the decimal weights of these bits is 60. Therefore, you can enable any of these bits to generate the summary bit by sending:

```
OUTPUT 707;"*ESE 60"
```

Whenever an error occurs, it sets one of these bits in the Standard Event Status Register. Because the bits are all enabled, a summary bit is generated to set bit 5 (ESB) in the Status Byte Register.

If bit 5 (ESB) in the Status Byte Register is enabled (via the *SRE command), an SRQ service request interrupt is sent to the external computer.

Disabled Standard Event Status Register bits respond but do not generate a summary bit.

Standard Event Status Register bits that are not enabled still respond to their corresponding conditions (that is, they are set if the corresponding event occurs). However, because they are not enabled, they do not generate a summary bit to the Status Byte Register.

Operation Status Register (OPR)

This register hosts the RUN bit (bit 3), the WAIT TRIG bit (bit 5), and the OVLR bit (bit 11).

- The RUN bit is set whenever the instrument is not stopped.
- The WAIT TRIG bit is set by the Trigger Armed Event Register and indicates that the trigger is armed.
- The OVLR bit is set whenever a 50Ω input overload occurs (54640-series only)
- If any of these bits are set, the OPER bit (bit 7) of the Status Byte Register is set. The Operation Status Register is read and cleared with the OPER? query. The register output is enabled or disabled using the mask value supplied with the OPEE command.

Arm Event Register (ARM)

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

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

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

Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. If the error queue overflows, the last error in the queue is replaced with error 350, Queue overflow. Any time the queue overflows, the least recent errors remain in the queue, and the most recent error is discarded. The length of the oscilloscope's error queue is 30 (29 positions for the error messages, and 1 position for the Queue overflow message).

The error queue is read with the SYSTEM:ERROR? query. Executing this query reads and removes the oldest error from the head of the queue, which opens a position at the tail of the queue for a new error. When all the errors have been read from the queue, subsequent error queries return 0, No error.

The error queue is cleared when:

- the instrument is powered up,
- the instrument receives the *CLS common command, or
- the last item is read from the error queue.

Output Queue

The output queue stores the oscilloscope-to-controller responses that are generated by certain instrument commands and queries. The output queue generates the Message Available summary bit when the output queue contains one or more bytes. This summary bit sets the MAV bit (bit 4) in the Status Byte Register.

The output queue may be read with the HP Basic ENTER statement.

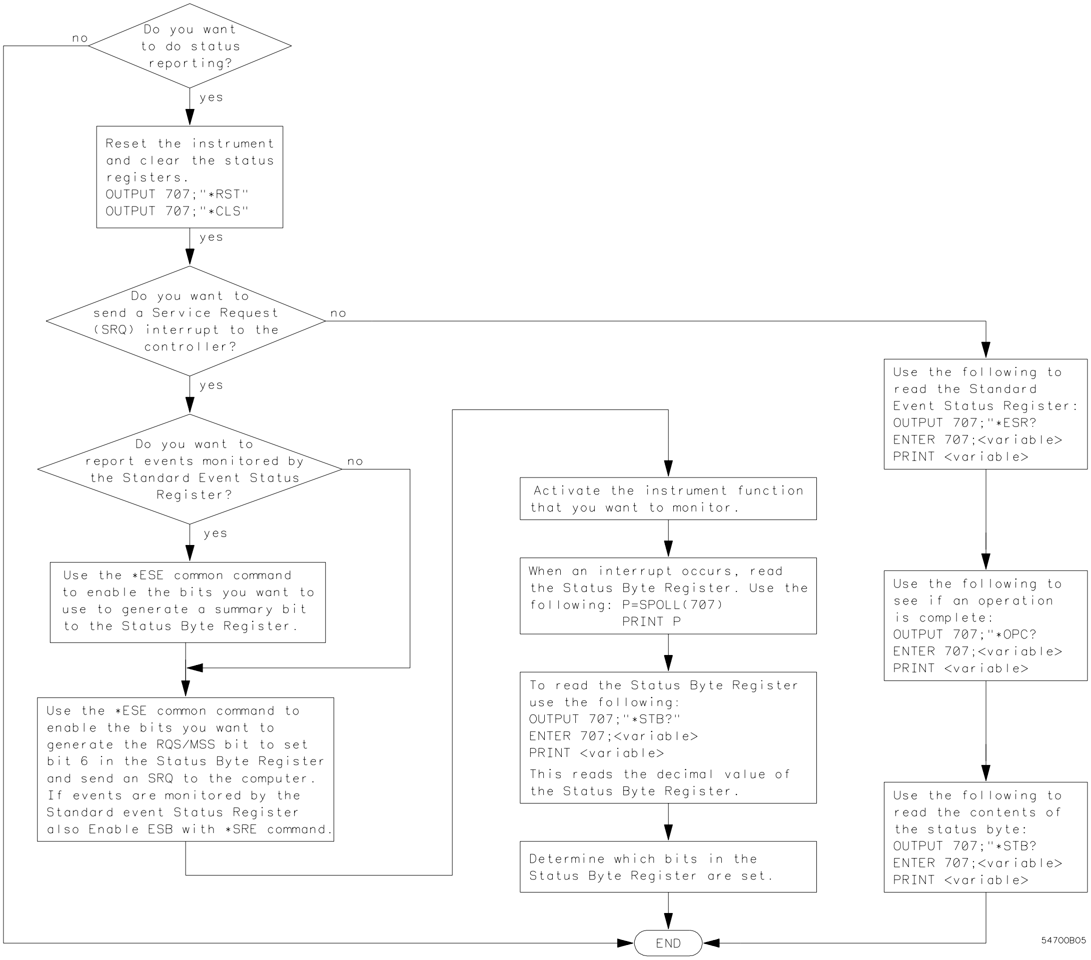
Message Queue

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

Clearing Registers and Queues

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

Figure 6-3


Do you want to do status reporting?
no
yes
Reset the instrument and clear the status registers. OUTPUT 707;"*RST" OUTPUT 707;"*CLS"
yes
Do you want to send a Service Request (SRQ) interrupt to the controller?
no
yes
Do you want to report events monitored by the Standard Event Status Register?
no
yes
Use the *ESE common command to enable the bits you want to use to generate a summary bit to the Status Byte Register.
Use the *ESE common command to enable the bits you want to generate the RQS/MSS bit to set bit 6 in the Status Byte Register and send an SRQ to the computer. If events are monitored by the Standard event Status Register also Enable ESB with *SRE command.
Activate the instrument function that you want to monitor.
When an interrupt occurs, read the Status Byte Register. Use the following: P=SPOLL(707) PRINT P
To read the Status Byte Register use the following: OUTPUT 707;"*STB?" ENTER 707;<variable> PRINT <variable> This reads the decimal value of the Status Byte Register.
Determine which bits in the Status Byte Register are set.
Use the following to read the Standard Event Status Register: OUTPUT 707;"*ESR? ENTER 707;<variable> PRINT <variable>
Use the following to see if an operation is complete: OUTPUT 707;"*OPC? ENTER 707;<variable> PRINT <variable>
Use the following to read the contents of the status byte: OUTPUT 707;"*STB? ENTER 707;<variable> PRINT <variable>
END
54700B05

Status Reporting Decision Chart

7

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

1 Insert the “Programmer’s Reference” floppy disk into the floppy disk drive of your PC.

2 Select **`Start | Run`** from the Windows taskbar, 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 54620_40-Series Programmer’s Reference

You can choose a different directory if desired. It also creates a Start Program group 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.

1 Log on to your Internet service.

2 Connect to www.agilent.com/find/5462xsw.

3 Under the "Software Updates" heading in the web page, click on the "54620/40-Series Oscilloscope Programming Reference Help File," then follow the instructions on the web page to download the help file.

To start the help file

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

The help file requires the program WINHELP.EXE for Microsoft Windows 95/98/NT or greater. 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.

8

Programmer’s Quick Reference

Introduction

The Programmer's Quick Reference provides the commands and queries with their corresponding arguments and returned formats for the 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.

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>.
I "or"	Indicates a choice of one element from a list. For example, <A> I <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 *Programmer’s Reference* help file.

Commands and Queries

The following tables facilitate easy access to each command and query for the 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. The NR1 and NR3 formats refer only to the Query Return values. Input arguments are not restricted by these formats.

These commands also show specific information about how command operates on a particular oscilloscope model. For additional information, refer to the online oscilloscopes *Programmer’s Reference* help file.

Command	**Query**	**Options and Query Returns**
:ACQuire:COMPlete <complete>	:ACQuire:COMPlete?	<complete> ::= 100; an integer in NR1 format
:ACQuire:COUNt <count>	:ACQuire:COUNT?	<count> ::= an integer from 1 to 16383 in NR1 format
:ACQuire:MODE <mode>	:ACQuire:MODE?	<mode> ::= {RTIMe I ETIMe}
n/a	**:ACQuire:POINts?**	2,000; an integer in NR1 format.
n/a	**:ACQuire:SRATe?**	<point_argument> ::=sample rate (samples/s) in NR3 format
:ACQuire:TYPE <type>	:ACQuire:TYPE?	<type> ::= {NORMal I AVERage I PEAK}
:ACTivity	:ACTivity?	<return value> ::= <edges>, <levels> <edges> ::= presence of edges (32-bit integer in NR1 format) <levels> ::= logical highs or lows (32-bit integer in NR1 format)
n/a	**:AER?**	{0 I 1}; an integer in NR1 format
:AUToscale	n/a	n/a
:BLANk <source>	n/a	<source> ::= {CHANnel<n>} I FUNCtion I MATH} for 546xxA <source> ::= {CHANnel<n> I DIGital0,...,DIGital15 I POD{1 I 2} I FUNCtion I MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
n/a	**:CALibrate:DATE?**	<return value> ::= <day>,<month>,<year>; all in NR1 format
:CALibrate:LABel <string>	:CALibrate:LABel?	<string> ::= quoted ASCII string up to 32 characters
n/a	**:CALibrate:SWITch?**	{PROTected I UNPRotected}
n/a	**:CALibrate:TIME?**	<return value> ::= <hours>,<minutes>,<seconds>; all in NR1 format
:CDISplay	n/a	n/a
:CHANnel<n>:BWLimit {{0 I OFF} I {1 I ON}}	:CHANnel<n>:BWLimit?	{0 I 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:COUPling <coupling>	:CHANnel<n>:COUPling?	<coupling> ::= {AC I DC I GND} for 54620-series <coupling> ::= {AC I DC} for 54640-series <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:DISPlay {{0 I OFF} I {1 I ON}}	:CHANnel<n>:DISPlay?	{0 I 1} <n> ::= 1-2 or 1-4 in NR1 format

Command	Query	Options and Query Returns
:CHANnel<n>:IMPedence <impedence>	:CHANnel<n>:IMPedence?	<impedence> ::= {ONEMeg} for 54620-series <impedence> ::= {ONEMeg | FIFTy} for 54640-series <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:INVert {{0 | OFF} | {1 | ON}}	:CHANnel<n>:INVert?	{0 | 1} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:LABel <string>	:CHANnel<n>:LABel?	<string>::= any series of 6 or less ASCII characters enclosed in quotation marks <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:OFFSet <offset> [suffix]	:CHANnel<n>:OFFSet?	<offset> ::= Vertical offset value in NR3 format. [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4; in NR1 format
:CHANnel<n>:PROBe <attenuation>	:CHANnel<n>:PROBe?	<attenuation> ::= Probe attenuation ratio in NR3 format <n> ::= 1-2 or 1-4r in NR1 format
:CHANnel<n>:PROBe:SKEW <skew_value>	:CHANnel<n>PROBe:SKEW?	<skew_value> ::= -100 ns to +100 ns in NR3 format <n> ::= 1-2 in NR1 format, for 54640-series only
:CHANnel<n>:PROTection[:CLEAR]	:CHANnel<n>:PROTection?	{NORM | TRIP} <n> ::= 1-2 in NR1 format, for 54640-series only
:CHANnel<n>:RANGe <range> [suffix]	:CHANnel<n>:RANGe?	<range> ::= Vertical full-scale range value in NR3 format. [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:SCALe <scale> [suffix]	:CHANnel<n>:SCALe?	<scale> ::= Vertical units per division value in NR3 format. [suffix] ::= {V | mV} <n> ::= 1-2 or 1-4 in NR1 format
:CHANnel<n>:UNITs <units>	:CHANnel<n>:UNITs?	<units> ::= {VOLTs | AMPeres} <n> ::= 1-2 or 1-4 in NR1 format
***CLS**	n/a	n/a
:DIGital<n>:DISPlay {{0 | OFF} | {1 | ON}}	:DIGital<n>:DISPlay?	{0 | 1} <n> ::= 0-15; an integer in NR1 format
:DIGital<n>:LABel <string>	:DIGital<n>:LABel?	<string>::= any series of 6 or less ASCII characters enclosed in quotation marks <n> ::= 0-15; an integer in NR1 format
:DIGital<n>:POSition <position>	:DIGital<n>:POSition?	<n> ::= 0-15; an integer in NR1 format <position> ::= 0-7 if display size = large, 0-15 if size = medium, 0-31 if size = small

:DIGital<n>:THReshold <value>[suffix]	:DIGital<n>:THReshold?	<n> ::= 0-15; an integer in NR1 format
		<value> ::= {CMOS | ECL | TTL | <user defined value>}
		<user defined value> ::= value in NR3 format from -8.00 to +8.00
		[suffix] ::= {V | mV | uV}

:DIGitize [<source>[,...,<source>]]	n/a	<source> ::= {CHANnel<n> | FUNCtion |MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | POD1 | POD2 | FUNCtion |MATH} for 546xxD <source> can be repeated up to 5 times. <n> ::= 1-2 or 1-4 in NR1 format
:DISPlay:CLEar	n/a	n/a

Command	**Query**	**Options and Query Returns**
:DISPlay:DATA [format][,][area] <display data>	:DISPlay:DATA? [format][,][area]	<format> ::= {TIFF} (command only) <area> ::= {GRATicule} (command only) <format> ::= {TIFF | BMP} (query only) <area> ::= {GRATicule | SCReen} (query only) <display data> ::= data in IEEE 488.2 # format
:DISPlay:LABel {{0 | OFF} | {1 | ON}}	:DISPlay:LABel?	{0 | 1}
:DISPlay:LABList #80000524 <binary block>	:DISPlay:LABList?	<binary block> ::= a time-ordered list of 75 labels. Each label can be a maximum of 6 characters followed by a comma.
:DISPlay:PERSistence <value>	:DISPlay:PERSistence?	<value> ::= {MINimum | INFinite}}
:DISPlay:SOURce <value>	:DISPlay:SOURce?	<value> ::= {PMEMory{0 | 1 | 2}}
:DISPlay:VECTors{{1 | ON} | {0 | OFF}}	:DISPlay:VECTors?	{1 | 0}
***DMC <macro label>,<macro definition>**	n/a	<macro label> ::= quoted ASCII string <macro definition> ::= block data in IEEE 488.2 # format
***EMC {{0 | OFF} | {1 | ON}}**	*EMC?	{0 | 1}
***ESE <mask>**	*ESE?	<mask> ::= 0 to 255; an integer in NR1 format Bit Weight Enables 7 128 PON - Power On 6 64 URQ - User Request 5 32 CME - Command Error 4 16 EXE - Execution Error 3 8 DDE - Device Dependent Error 2 4 QYE - Query Error 1 2 TRG - Trigger Query 0 1 OPC - Operation Complete
:EXTernal:BWLimit <bwlimit>	:EXTernal:BWLimit?	<bwlimit> ::= {0 | OFF}
:EXTernal:IMPedance <value>	:EXTernal:IMPedance?	<impedence> ::= {ONEMeg} for 54620-series <impedence> ::= {ONEMeg | FIFTy} for 54640-series
:EXTernal:PROBe <attenuation>	:EXTernal:PROBe?	<attenuation> ::= probe attenuation ratio in NR3 format
:EXTernal:PROTection[:CLEAR]	:EXTernal:PROTection?	{NORM | TRIP}
:EXTernal:RANGe <range>[<suffix>]	:EXTernal:RANGe?	<range> ::= vertical full-scale range value in NR3 format <suffix> ::= {V | mV}
:EXTernal:UNITs <units>	**:EXTernal:UNITs?**	<units> ::= {VOLTs | AMPeres}
n/a	***ESR?**	<status> ::= 0 to 255; an integer in NR1 format
:FUNCtion:CENTer <frequency>	:FUNCtion:CENTer?	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from 0 Hz to 25.00 GHz.
:FUNCtion:DISPlay {{0 | OFF} | {1 | ON}}	:FUNCtion:DISPlay?	{0 | 1}
:FUNCtion:OFFSet <offset>	:FUNCtion: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.
:FUNCtion:OPERation <operation>	:FUNCtion:OPERation?	<operation> ::= {SUBTract | MULTiply | INTegrate | DIFFerentiate | FFT}

Command	**Query**	**Options and Query Returns**
:FUNCtion:RANGe <range>	:FUNCtion:RANGe?	<range> ::= the full-scale vertical axis value in NR3 format. The range for ADD, SUBT, MULT is 8E-6 to 800E+3. The range for the INTegrate function is 8E-9 to 400E+3. The range for the DIFFerentiate function is 80E-3 to 8.0E12 (depends on current sweep speed). The range for the FFT function is 8 to 800 dBV.
:FUNCtion:REFerence <level>	:FUNCtion:REFerence?	<level> ::= the current reference level in NR3 format. The range of legal values is from 400.0 dBV to +400.0 dBV (depending on current range value).
:FUNCtion:SCALe <scale value>[<suffix>]	:FUNCtion:SCALe	<scale value> ::= integer in NR1 format <suffix> ::= {V | dB}
:FUNCtion:SOURce <source>	:FUNCtion:SOURce?	<source> ::= {CHANnel<n> | ADD | SUBT | MULT} <n> ::= 1-2 or 1-4 in NR1 format
:FUNCtion:SPAN <span>	:FUNCtion:SPAN?	<span> ::= the current frequency span in NR3 format. Legal values are 1 Hz to 100 GHz
:FUNCtion:WINDow <window>	:FUNCtion:WINDow?	<window> ::= {RECTangular | HANNing | FLATtop}
n/a	***GMC? <macro label>**	<macro label> ::= quoted ASCII string, block data in IEEE 488.2 # format
:HARDcopy:DESTination <destination>	:HARDcopy:DESTination	<destination> ::= {CENTronics | FLOPpy}
:HARDcopy:FACTors {{0 | OFF} | {1 | ON}}	:HARDcopy:FACTors?	{0 | 1}
:HARDcopy:FFEed {{0 | OFF} | {1 | ON}}	:HARDcopy:FFEed?	{0 | 1}
:HARDcopy:FORMat <device>	:HARDcopy:FORMat?	<format> ::= {TIFF | BMP | CSV | LASerjet | DESKjet | EPSon | SEIKo}
:HARDcopy:GRAYscale {{0 | OFF} | {1 | ON}}	:HARDcopy:GRAYscale?	{0 | 1}
n/a	***IDN?**	AGILENT TECHNOLOGIES,<model>, <serial number>,X.XX.XX <model> ::= the model number of the instrument <serial number> ::= the serial number of the instrument <X.XX.XX> ::= the software revision of the instrument
n/a	***LMC?**	<ascii string> ::= string list seperated by commas
n/a	***LRN?**	<learn_string> ::= current instrument setup as a block of data in IEEE 488.2 # format
:MARKer:MODE<mode>	:MARKer:MODE?	<mode> ::= {OFF | MEASure | MANual}
:MARKer:X1Position <position> [suffix]	:MARKer:X1Position?	<position> ::= X1 cursor position value in NR3 format [suffix] ::= {s | ms | ms | ns | ps | Hz | kHz | MHz} <return_value> ::= X1 cursor positon value in NR3 format
:MARKer:X1Y1source <source>	:MARKer:X1Y1 <source>	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= <source>
:MARKer:X2Position <position> [suffix]	:MARKer:X2Position?	<position> ::= X2 cursor position value in NR3 format [suffix] ::= {s | ms | ms | ns | ps | Hz | kHz | MHz} <return_value> ::= X2 cursor positon value in NR3 format
:MARKer:X2Y2source <source>	:MARKer:X2Y2 <source>	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= <source>

Command	Query	Options and Query Returns
n/a	**:MARker:XDELta?**	<return_value> ::= X cursors delta value in NR3 format
:MARKer:Y1Position <positon>[suffix]	:MARKer:Y1Position?	<positon> ::= Y1 cursor position value in NR3 format [suffix] ::= {V | mV | dB} <return_value> ::= Y1 cursor positon value in NR3 format
:MARKER:Y2Position <positon>[suffix]	:MARKer:Y2Position?	<positon> ::= Y2 cursor position value in NR3 format [suffix] ::= {V | mV | dB} <return_value> ::= Y2 cursor positon value in NR3 format
n/a	**:MARKer:YDELta?**	<return_value> ::= Y cursors delta value in NR3 format
:MEASure:CLEar	n/a	n/a
:MEASure:COUNter [<source>]	MEASure:COUNter? [<source>]	<source> ::= {CHANnel<n>} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 |} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::=counter frequency in Hertz in NR3 format
:MEASure:DEFine DELay, <edge spec1, edge spec2>	:MEASure:DEFine? DELay	edge_spec1 ::= [<slope>] <occurrence> edge_spec2 ::= [<slope>] <occurrence> <slope> ::= {+ | -} <occurrence> ::= integer

:MEASure:DEFine THResholds, <threshold spec>	:MEASure:DEFine? THResholds	<threshold spec> ::= {STANdard} | {<threshold mode>,<upper>, <middle>,<lower>}
		<threshold mode> ::= {PERCent | ABSolute}

:MEASure:DELay [<source1>] [,<source2>]	:MEASure:DELay? [<source1>] [,<source2>]	<source1,2> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DUTYcycle [<source>]	:MEASure:DUTYcycle? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= ratio of positive pulse width to period in NR3 format
:MEASure:FALLtime [<source>]	:MEASure:FALLtime? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= time in seconds between the lower and upper thresholds in NR3 format
:MEASure:FREQuency [<source>]	:MEASure:FREQuency? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= frequency in Hertz in NR3 format
:MEASure:NWIDth [<source>]	:MEASure:NWIDth? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= negative pulse width in seconds-NR3 format
:MEASure:OVERshoot [<source>]	:MEASure:OVERshoot? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= the percent of the overshoot of the selected waveform in NR3 format

Command	Query	Options and Query Returns
:MEASure:PERiod [<source>]	:MEASure:PERiod? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= waveform period in seconds in NR3 format
:MEASure:PHASe [<source1>] [,<source2>]	:MEASure:PHASe? [<source1>] [,<source2>]	<source1,2> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= the phase angle value in degrees in NR3 format
:MEASure:PREShoot [<source>]	:MEASure:PREShoot? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= the percent of preshoot of the selected waveform in NR3 format
:MEASure:PWIDth [<source>]	:MEASure:PWIDth? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= width of positive pulse in seconds in NR3 format
:MEASure:RISEtime [<source>]	:MEASure: RISEtime? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= rise time in seconds in NR3 format
:MEASure:SHOW {1 | ON}	:MEASure:SHOW?	{1}
:MEASure:SOURce [<source1>] [,<source2>]	:MEASure:SOURce?	<source1,2> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source1,2> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= {<source> | NONE}
:MEASure:TEDGe <slope><occurrence>[,<source>]	:MEASure:TEDGe?	<slope> ::= direction of the waveform <occurrence> ::= the transition to be reported. <source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= time in seconds of the specified transition
n/a	**:MEASure:TVALue? <value>, [<slope>]<occurrence> [,<source>]**	<value> ::= voltage level that the waveform must cross. <slope> ::= direction of the waveform when <value> is crossed. <occurrence> ::= transitions reported. <return_value> ::= time in seconds of specified voltage crossing in NR3 format <source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:MEASure:VAMPlitude [<source>]	:MEASure:VAMPlitude? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= the amplitude of the selected waveform in volts in NR3 format
:MEASure:VAVerage [<source>]	:MEASure:VAVerage? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= calculated average voltage in NR3 format

Command	Query	Options and Query Returns
:MEASure:VBASe [<source>]	:MEASure:VBASe? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <base_voltage> ::= voltage at the base of the selected waveform in NR3 format
:MEASure:VMAX [<source>]	:MEASure:VMAX? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= maximum voltage of the selected waveform in NR3 format
:MEASure:VMIN [<source>]	:MEASure:VMIN? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= minimum voltage of the selected waveform in NR3 format
:MEASure:VPP [<source>]	:MEASure:VPP? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= voltage peak-to-peak of the selected waveform in NR3 format
:MEASure:VRMS [<source>]	:MEASure:VRMS? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= calculated dc RMS voltage in NR3 format
n/a	**:MEASure:VTIMe? <vtime>[,<source>]**	<vtime> ::= displayed time from trigger in seconds in NR3 format <return_value> ::= voltage at the specified time in NR3 format <source> ::= {CHANnel<n> | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:MEASure:VTOP [<source>]	:MEASure:VTOP? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= voltage at the top of the waveform in NR3 format
:MEASure:XMAX [<source>]	:MEASure:XMAX? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= horizontal value of the maximum in NR3 format
:MEASure:XMIN [<source>]	:MEASure:XMIN? [<source>]	<source> ::= {CHANnel<n> | FUNCtion | MATH} <n> ::= 1-2 or 1-4 in NR1 format <return_value> ::= horizontal value of the maximum in NR3 format
:MERGe <pixel memory>	n/a	<pixel memory> ::= {PMEMory{0 | 1 | 2}}
***OPC**	*OPC?	ASCII ""1"" is placed in the output queue when all pending device operations have completed.
:OPEE <n>	:OPEE?	<n> ::= 16-bit integer in NR1 format
n/a	**:OPER?**	<n> ::= 16-bit integer in NR1 format
n/a	***OPT?**	<return_value> ::= n, A.XX.XX n identifies the module. XX.XX identifies the module software revision. N2757A, A.XX.XX

Command	Query	Options and Query Returns
OVLenable <mask>	OVLenable?	<mask> ::= 8-bit integer in NR1 format as shown, 54640-series only: Bit Weight Input 4 16 External Trigger 3 8 Channel 4 2 4 Channel 3 1 2 Channel 2 0 1 Channel 1
n/a	**OVLRegister?**	<value> ::= integer in NR1 format, 54640-series only. See OVLenable for <value>
***PMC**	n/a	n/a
:POD<n>:DISPlay{{0 | OFF} | {1 | ON}}	:POD<n>:DISPlay?	{0 | 1} <n> ::= 1-2 in NR1 format

:POD<n>:THReshold <type>[suffix]	:POD<n>:THReshold?	<n> ::= 1-2 in NR1 format
		<type> ::= {CMOS | ECL | TTL | <user defined value>}
		<user defined value> ::= value in NR3 format
		[suffix] ::= {V | mV | uV }

:PRINt [parameter][,parameter]]	:PRIN? [parameter][,parameter] x 4]	<parameter> ::= {HIRes | LORes | TIFF | PCL | BMP | PARallel | DISK | FACTors | NOFactors} <parameter> can be repeated up to 5 times.
***RCL <value>**	n/a	<value> ::= {0 | 1 | 2}
***RST**	n/a	See reset values in the online Programmer's Reference.
:RUN	n/a	n/a
***SAV <value>**	n/a	<value> ::= {0 | 1 | 2}
	:SERial?	<return value> ::= unquoted string containing serial number
:SINGle	n/a	n/a
***SRE <mask>**	*SRE?	<mask> ::= sum of all bits that are set, 0 to 255; an integer in NR1 format. <mask> ::= following values: Bit Weight Enables 7 128 OPER - Operation Status Register 6 64 Not Used 5 32 ESB - Event Status Bit 4 16 MAV - Message Available 3 8 Not used 2 4 MSG - Message 1 2 USR - User 0 1 TRG - Trigger

n/a	**:STATus? <display>**	{0 | 1}
		<display> ::= {CHANnel<n> | DIGital0,...,DIGital15 | FUNCtion |MATH};
		<n> ::= 1-2 or 1-4 in NR1 format

Command	Query	Options and Query Returns
n/a	***STB?**	<value> ::= 0 to 255; an integer in NR1 format, as shown in the following: Bit Weight Name Condition 7 128 OPER 0 = no enabled operations status conditions occurred 1 = an enabled operation status condition occurred 6 64 RQS/MS 0 = instrument has no reason for service 1 = instrument is requesting service 5 32 ESB 0 = no event status conditions occurred 1 = enabled event status condition occurred 4 16 MAV 0 = no output messages are ready 1 = an output message is ready 3 8 ---- 0 = not used 2 4 MSG 0 = no message has been displayed 1 = message has been displayed 1 2 USR 0 = no enabled user event conditions have occurred 1 = an enabled user event condition has occurred 0 1 TRG 0 = no trigger has occurred 1 = a trigger occurred
:STOP	n/a	n/a

:SYSTem:DATE <date>	:SYSTem:DATE?	<date> ::= <year>, <month>, <day>
		<year> ::= 4-digit year in NR1 format
		<month> ::= {1,..,12 | JANuary | FEBruary | MARch | APRil | MAY | JUNe | JULy | AUGust | SEPtember | OCTober | NOVember | DECember}
		<day> ::= {1,..31}

:SYSTem:DSP <string>	n/a	<string> ::= up to 254 characters as a quoted ASCII string
n/a	**:SYSTem:ERRor?**	<error> ::= an integer error code <error string> ::= quoted ASCII string. See error values in the online Programmer’s Reference.
:SYSTem:LOCK	:SYSTem:LOCK?	<value> ::= {ON | OFF}
:SYSTem:SETup <setup_data>	:SYSTem:SETup?	<setup_data> ::= data in IEEE 488.2 # format.
:SYSTem:TIME <time>	:SYSTem:TIME?	<time> ::= hours, minutes, seconds in NR1 format
n/a	**:TER?**	{0 | 1}
:TIMebase:MODE <value>	:TIMebase:MODE?	<value> ::= {MAIN | WINDow | XY | ROLL}
:TIMebase:POSition <pos>	:TIMebase:POSition?	<pos> ::= time from the trigger event to the display reference point in NR3 format

Command	Query	Options and Query Returns
:TIMebase:RANGe <range_value>	:TIMebase:RANGe?	<range_value> ::= 50 ns through 500 s in NR3 format for 54620-series <range_value> ::= 10 ns through 500 s in NR3 format for 54640-series
:TIMebase:REFerence {LEFT | CENTer | RIGHt}	:TIMebase:REFerence?	<return_value> ::= {LEFT | CENTer | RIGHt}
:TIMebase:SCALe <scale_value>	:TIMebase:SCALe?	<scale_value> ::= scale value in seconds in NR3 format
:TIMebase:WINDow:POSition <pos>	:TIMebase:WINDow:POSition?	<pos> ::= time from the trigger event to the delayed view reference point in NR3 format
:TIMebase:WINDow:RANGe <range_value>	:TIMebase:WINDow:RANGe?	<range value> ::= range value in seconds in NR3 format for the delayed window
:TIMebase:WINDow:SCALe <scale_value>	:TIMebase:WINDow:SCALe?	<scale_value> ::= scale value in seconds in NR3 format for the delayed window
***TRG**	n/a	n/a
:TRIGger:CAN:ACKNowledge <value>	:TRIGger:CAN:ACKNowledge?	<value> ::= {{0 | OFF} | {1 | ON}} with N2758A CAN Trigger module attached
:TRIGger:CAN:PATTern:DATA <value>, <mask>	:TRIGger:CAN:PATTern:DATA?	<value> ::= integer or <string> with N2758A CAN Trigger module attached <mask> ::= integer or <string> <string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F} for hexadecimal, or <string> ::= #Hnn...n where n ::= {0,...,9 | A,...,F} for hexadecimal, or <string> ::= #Bnn...n where n ::= {0 | 1} for binary
:TRIGger:CAN:PATTern:DATA: LENGth <length>	:TRIGger:CAN:PATTern:DATA: :LENGth?	<length> ::= integer from 1 to 8 in NR1 format with N2758A CAN Trigger module attached
:TRIGger:CAN:PATTern:ID <value>, <mask>	:TRIGger:CAN:PATTern:ID?	<value> ::= integer or <string> with N2758A CAN Trigger module attached <mask> ::= integer or <string> <string> ::= "0xnn...n" where n ::= {0,...,9 | A,...,F} for hexadecimal, or <string> ::= #Hnn...n where n ::= {0,...,9 | A,...,F} for hexadecimal, or <string> ::= #Bnn...n where n ::= {0 | 1} for binary
:TRIGger:CAN:PATTern:ID:MODE <value>	:TRIGger:CAN:PATTern:ID:MODE?	<value> ::= {STANdard | EXTended} with N2758A CAN Trigger module attached
:TRIGger:CAN:SAMPlepoint <value>	:TRIGger:CAN:SAMPlepoint?	<value> ::= {60 | 62.5 | 68 | 70 | 75 | 87.5} in NR3 format with N2758A CAN Trigger module attached
:TRIGger:CAN:SIGNal:BAUDrate <baudrate>	:TRIGger:CAN:SIGNal:BAUDrate?	<baudrate> ::= {10000 | 20000 | 33300 | 50000 | 62500 | 83300 | 100000 | 125000 | 250000 | 500000 | 800000 | 1000000}
:TRIGger:CAN:SIGNal:DEFinition <value>	:TRIGger:CAN:SIGNal:DEFinition?	<value> ::= {CANH | CANL | RX | TX | DIFFerential}
:TRIGger:CAN:SOURce <source>	:TRIGger:CAN:SOURce?	<source> ::= {CHANnel<n> | EXTernal} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 |} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:CAN:TRIGger <condition>	:TRIGger:CAN:TRIGer?	<condition> ::= {SOF} without N2758A CAN Trigger module attached <condition>::={SOF | DATA | ERRor | IDData | IDEither | IDRemote} with N2758A CAN Trigger module attached
:TRIGGER:DURation:GREaterthan <greater than time> [suffix]	:TRIGger:DURation:GREaterthan?	<greater than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}

Command	Query	Options and Query Returns
:TRIGGER:DURation:LESSthan <less than time> [suffix]	:TRIGger:DURation:LESSthan?	<less than time> ::= floating-point number from 5 ns to 10seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:DURation:PATTern <value>, <mask>	:TRIGger:DURation:PATTern?	<value> ::= integer or <string> <mask> ::= integer or <string> <string> ::= ""0xnnnnnn"" n ::= {0 ,...., 9 | A ,...., F}
:TRIGger:DURation:QUALifier <qualifier>	:TRIGger:DURation:QUALifier?	<qualifier> ::= {GREaterthan | LESSthan | INRange | OUTRange | TIMeout}
:TRIGger:DURation:RANGe <greater than time> [suffix], <less than time> [suffix]	:TRIGger:DURation:RANGe?	<greater than time> ::= min duration from 10 ns to 9.99 seconds in NR3 format <less than time> ::= max duration from 15 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:[EDGE:]COUPling {AC | DC | LF}	:TRIGger:COUPling?	{AC | DC | LF}
:TRIGger:[EDGE:]LEVel <level> [,<source>]	:TRIGger:[EDGE]:LEVel? [<source>]	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format. For external triggers, <level> ::= 2 volts with probe attenuation at 1:1 in NR3 format. For digital channels (546xxD only), <level> ::= 8 V. <source> ::= {CHANnel<n> | EXTernal} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | EXTernal } for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:[EDGE:]REJect {OFF | LF | HF}	:TRIGger:REJect?	{OFF | LF | HF}
:TRIGger:[EDGE:]SLOPe <polarity>	:TRIGger:[EDGE]:SLOPe?	<polarity> ::= {POSitive | NEGative}
:TRIGger:[EDGE:]SOURce <source>	:TRIGger:[EDGE]:SOURce?	<source> ::= {CHANnel<n> | EXTernal} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | EXTernal} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGGER:GLITch:GREaterthan <greater than time> [suffix]	:TRIGger:GLITch:GREaterthan?	<greater than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGGER:GLITch:LESSthan <less than time> [suffix]	:TRIGger:GLITch:LESSthan?	<less than time> ::= floating-point number from 5 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:GLITch:LEVel <level> [<source>]	:TRIGger:GLITch:LEVel?	For internal triggers, <level> ::= .75 x full-scale voltage from center screen in NR3 format. For external triggers , <level> ::= 2 volts with probe attenuation at 1:1 in NR3 format. For digital channels (546xxD only), <level> ::= 6 V. <source> ::= {CHANnel<n> | EXTernal} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:GLITch:POLarity <polarity>	:TRIGger:GLITch:POLarity?	<polarity> ::= {POSitive | NEGative}
:TRIGger:GLITch:QUALifier <qualifier>	:TRIGger:GLITch:QUALifier?	<qualifier> ::= {GREaterthan | LESSthan | RANGe}

Command	Query	Options and Query Returns
:TRIGger:GLITch:RANGe <greater than time> [suffix], <less than time> [suffix]	:TRIGger:GLITch:RANGe?	<greater than time> ::= start time from 10 ns to 9.99 seconds in NR3 format <less than time> ::= stop time from 15 ns to 10 seconds in NR3 format [suffix] ::= {s | ms | us | ns | ps}
:TRIGger:GLITch:SOURce <source>	:TRIGger:GLITch:SOURce?	<source> ::= {CHANnel<n> | EXTernal} for 546xxA; <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 } for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:HFReject {{0 | OFF} | {1 | ON}}	:TRIGger:HFReject?	{0 | 1}
:TRIGger:HOLDoff <holdoff_time>	:TRIGger:HOLDoff?	<holdoff_time> ::= 60 ns to 10 s in NR3 format
:TRIGger:IIC:PATTern:ADDRess <value>	:TRIGger:IIC:PATTern:ADDRess?	<value> ::= integer or <string> <string> ::= ""0xnn"" n ::= {0 ,...., 9 | A ,...., F}
:TRIGger:IIC:PATTern:DATA <value>	:TRIGger:IIC:PATTern:DATA?	<value> ::= integer or <string> <string> ::= ""0xnn"" n ::= {0 ,...., 9 | A ,...., F}
:TRIGger:IIC:[SOURce:]CLOCk <source>	:TRIGger:IIC:[SOURce:]CLOCk?	<source> ::= {CHANnel<n> | EXTernal} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 } for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:IIC:[SOURce:]DATA <source>	:TRIGger:IIC:[SOURce:]DATA?	<source> ::= {CHANnel<n> | EXTernal} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 } for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:IIC:TRIGger:QUALifier <value>	:TRIGger:IIC:TRIGger:QUALifer?	<value> ::={EQUal | NOTequal | LESSthan | GREaterthan}
:TRIGger:IIC:TRIGger[:TYPE] <type>	:TRIGger:IIC:TRIGger[:TYPE]?	<type> ::= {STARt | STOP | READ7 | READEeprom | WRITe7 | WRITe10 | NACK | RESTart}
:TRIGger:LIN:SIGNal:BAUDrate <baudrate>	:TRIGger:LIN:SIGNal:BAUDrate?	<baudrate> ::= {2400 | 9600 | 19200}
:TRIGger:LIN:SIGNal:DEFinition <value>	:TRIGger:LIN:SIGNal:DEFinition?	<value> ::= {LIN | RX | TX}
:TRIGger:LIN:SOURce <source>	:TRIGger:LIN:SOURce?	<source> ::= {CHANnel<n> | EXTernal} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:LIN:TRIGger <condition>	:TRIGger:LIN:TRIGger?	<condition> ::= {SYNCbreak}

:TRIGger:MODE <mode>	:TRIGger:MODE?	<mode> ::= {EDGE | GLITch | PATTern | CAN | DURation | IIC | SEQuence | SPI | TV | USB}
		<return_value> ::= {<mode> | <none>}
		<none> ::= query returns "NONE" if the :TIMebase:MODE is ROLL or XY

:TRIGger:NREJect {{0 | OFF} | {1 | ON}}	:TRIGger:NREJect?	{0 | 1}

:TRIGger:PATTern <value>, <mask> [,<edge source>, <edge>]	:TRIGger:PATTern?	<value> ::= 32-bit integer or <string>
		<mask> ::= 32-bit integer or <string>
		<string> ::= "0xnnnnnnn"; n ::= {0,...,9 | A,...,F}
		<edge source> ::= {DIGital0,..,DIGital15 | CHANnel<n> | NONE}
		<edge> ::= {POSitive | NEGative}
		<n> ::= 1-2 or 1-4 in NR1 format

:TRIGger:SEQuence:COUNt <count>	:TRIGger:SEQuence:COUNt?	<count> ::= integer in NR1 format

Command	Query	Options and Query Returns
:TRIGger:SEQuence:EDGE{1 | 2} <source>, <slope>	:TRIGger:SEQuence:EDGE{1 | 2}?	<source> ::= {CHANnel<n> | EXTernal} for the 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15} for the 546xxD <slope> ::= {POSitive | NEGative} <n> ::= 1-2 or 1-4 in NR1 format <return_value>::= query returns "NONE" if edge source is disabled

:TRIGGER:SEQuence:FIND <resource1>[,<operator> [,<resource2>]]	:TRIGGER:SEQuence:FIND?	<resource1> ::= {PATTern1 | EDGE1}
		<operator> ::= {ENTered | EXITed | AND | NONE}
		<resource2> ::= {PATTern1 | EDGE1 | NONE}

:TRIGger:SEQuence:PATTern {1 | 2}<value>, <mask>	:TRIGger:SEQuence:PATTern{1 | 2}?	<value> ::= integer or <string> <mask> ::= integer or <string> <string> ::= ""0xnnnnnnn"" n ::= {0 ,..., 9 | A ,..., F}

:TRIGGER:SEQuence:RESet <resource1>[,<operator> [,<resource2>]]	:TRIGGER:SEQuence:RESet?	<resource1> ::= {PATTern{1 | 2} | EDGE{1 | 2} | TIMer | NONE}
		<operator> ::= {ENTered | EXITed | AND | NONE}
		<resource2> ::= {PATTern{1 | 2} | EDGE{1 | 2} | TIMer | NONE}

:TRIGGER:SEQuence:TIMer <time_value>	:TRIGGER:SEQuence:TIMer?	<time_value> ::= time from 100 ns to 10 seconds in NR3 format

:TRIGGER:SEQuence:TRIGger <resource1>[,<operator> [,<resource2>]]	:TRIGGER:SEQuence:TRIGger?	<resource1> ::= {PATTern2 | EDGE2}
		<operator> ::= {ENTered | EXITed | AND | COUNt | NONE}
		<resource2> ::= {PATTern2 | EDGE2 | NONE}

:TRIGger:SPI:CLOCk:SLOPe <slope>	:TRIGger:SPI:CLOCk:SLOPe?	<slope> ::= {NEGative | POSitive}
:TRIGger:SPI:CLOCk:TIMeout <time_value>	:TRIGger:SPI:CLOCk:TIMeout?	<time_value> ::= time in seconds in NR1 format
:TRIGger:SPI:FRAMing <value>	:TRIGger:SPI:FRAMing?	<value> ::= {CHIPselect | NOTChipselect | TIMeout}
:TRIGger:SPI:PATTern:DATA <value>, <mask>	:TRIGger:SPI:PATTern:DATA?	<value> ::= integer or <string> <mask> ::= integer or <string> <string> ::= "0xnnnnnnn" where n ::= {0,...,9 | A,...,F}
TRIGger:SPI:PATTern:WIDth <width>	:TRIGger:SPI:PATTern:WIDth?	<width> ::= integer from 4 to 32 in NR1 format
:TRIGger:SPI:SOURce:CLOCk <source>	:TRIGger:SPI:SOURce:CLOCk?	<value> ::= {CHANnel<n> | EXTernal} for the 546xxA <value> ::= {CHANnel<n> | DIGital0,...,DIGital15} for the 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:SPI:SOURce:DATA <source>	:TRIGger:SPI:SOURce:DATA?	<value> ::= {CHANnel<n> | EXTernal} for the 546xxA <value> ::= {CHANnel<n> | DIGital0,...,DIGital15} for the 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:SPI:SOURce:FRAMe <source>	:TRIGger:SPI:SOURce:FRAMe?	<value> ::= {CHANnel<n> | EXTernal} for the 546xxA <value> ::= {CHANnel<n> | DIGital0,...,DIGital15} for the 546xxD <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:SWEep <sweep>	:TRIGger:SWEep?	<sweep> ::= {AUTLevel | AUTO | NORMal} for 54620-series <sweep> ::= {AUTO | NORMal} for 54640-series
:TRIGger:TV:LINE <line number>	:TRIGger:TV:LINE?	<line number> ::= integer in NR1 format.
:TRIGger:TV:MODE <tv mode>	:TRIGger:TV:MODE?	<tv mode> ::= {Field<n> | ALLFields | LINE | ALLLines | LINEField<n> | LINEAlt | LINEVert} <n> ::= 1-2 in NR1 format
:TRIGger:TV:POLarity <polarity>	:TRIGger:TV:POLarity?	<polarity> ::= {POSitive | NEGative}
:TRIGger:TV:SOURce <source>	:TRIGger:TV:SOURce?	<source> ::= {CHANnel<n>} <n> ::= 1-2 or 1-4 integer in NR1 format

Command	Query	Options and Query Returns
:TRIGger:TV:STANdard <standard>	:TRIGger:TV:STANdard?	<standard> ::= {GENeric | NTSc | PAL | PALM | SECam}
:TRIGger:TV:TVMODE <mode>	:TRIGger:TV:TVMODE?	<mode> ::= {Field<n> | ALLFields | LINE | ALLLines | LINEField<n> | LINEAlt | LINEVert} <n> ::= 1-2 in NR1 format
:TRIGger:USB:SOURce:DMINus <source>	:TRIGger:USB:SOURce:DMINus?	<source> ::= {CHANnel<n> | EXTernal} for the 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15} for the 546xxD <slope> ::= {POSitive | NEGative} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:USB:SOURce:DPLus <source>	:TRIGger:USB:SOURce:DPLus?	<source> ::= {CHANnel<n> | EXTernal} for the 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15} for the 546xxD <slope> ::= {POSitive | NEGative} <n> ::= 1-2 or 1-4 in NR1 format
:TRIGger:USB:SPEed <value>	:TRIGger:USB:SPEed?	<value> ::= {LOW | FULL}
:TRIGger:USB:TRIGer <value>	:TRIGger:USB:TRIGger?	<value> ::= {SOP | EOP | ENTersuspend | EXITsuspend | RESet}
n/a	***TST?**	<result> ::= 0 or non-zero value; an integer in NR1 format
:VIEW <source>	n/a	<source> ::= {CHANnel<n> | PMEMory{0 | 1 | 2} | FUNCtion | MATH} for 546xxA <source> ::= {CHANnel<n> | DIGital0,...,DIGital15 | PMEMory{0 | 1 | 2} | FUNCtion | MATH} for 546xxD <n> ::= 1-2 or 1-4 in NR1 format
***WAI**	n/a	n/a
:WAVeform:BYTeorder <value>	:WAVeform:BYTeorder?	<value> ::= {LSBFirst | MSBFirst}
na	**:WAVeform:COUNt?**	<count> ::= an integer from 1 to 16383 in NR1 format
n/a	**:WAVeform:DATA?**	<binary block length bytes>, <binary data> For example, to transmit 2000 bytes of data, the syntax would be: #800002000<2000 bytes of data><NL> 8 is the number of digits that follow 00002000 is the number of bytes to be transmitted <2000 bytes of data> is the actual data
:WAVeform:FORMat <value>	:WAVeform:FORMat?	<value> ::= {WORD | BYTE | ASCII}
:WAVeform:POINts <# points>	:WAVeform:POINts? [MAXimum]	<# points> ::= {100 | 250 | 500 | 1000 | 2000 | MAXimum}
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 BYTE format 1 for WORD format 2 for ASCii format <type> ::= an integer in NR1 format: 2 for AVERage type 0 for NORMal type 1 for PEAK detect type <count> ::= Average count, or 1 if PEAK detect type or NORMal; an integer in NR1 format
:WAVeform:SOURce <source>	:WAVeform:SOURce?	<source> ::= {CHANnel<n> | POD1 | POD2 | FUNCtion |MATH} <n> ::= 1-2 integer in NR1 format
n/a	**:WAVeform:TYPE?**	<return_mode> ::= {NORM | PEAK | AVER}

Command	Query	Options and Query Returns
:WAVeform:UNSigned {{0 | OFF} | {1 | ON}}	:WAVeform:UNSigned?	{0 | 1}
:WAVeform:VIEW <view>	:WAVeform:VIEW?	<view> ::= {MAIN}
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> ::= 0 (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

Index

Safety Notices

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."

Warnings

- 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.
- If you energize this instrument by an auto transformer (for voltage reduction or mains isolation), the common terminal must be 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.
- 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.
- 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.
- 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 use the instrument in a manner not specified by the manufacturer.

To clean the instrument

If the instrument requires cleaning: (1) Remove power from the instrument. (2) Clean the external surfaces of the instrument with a soft cloth dampened with a mixture of mild detergent and water. (3) Make sure that the instrument is completely dry before reconnecting it to a power source.

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.

Agilent Technologies Inc.
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901-2197, U.S.A.

Notices

Manual Part Number

54622-97038, September 2002

Print History

54622-97038, September 2002
54622-97030, March 2002
54622-97027. November 2001
54622-97026, December 2000
54622-97013, August 2000
54622-97001, May 2000

Agilent Technologies, Inc.
1601 California Street
Palo Alto, CA 94304 USA

Restricted Rights Legend

If software is for use in the performance of a U.S. Government prime contract or subcontract, Software is delivered and licensed as "Commercial computer software" as defined in DFAR 252.227-7014 (June 1995), or as a "commercial item" as defined in FAR 2.101(a) or as "Restricted computer software" as defined in FAR 52.227-19 (June 1987) or any equivalent agency regulation or contract clause. Use, duplication or disclosure of Software is subject to Agilent Technologies' standard commercial license terms, and non-DOD Departments and Agencies of the U.S. Government will receive no greater than Restricted Rights as defined in FAR 52.227-19(c)(1-2) (June 1987). U.S. Government users will receive no greater than Limited Rights as defined in FAR 52.227-14 (June 1987) or DFAR 252.227-7015 (b)(2) (November 1995), as applicable in any technical data.

Document Warranty

The material contained in this document is provided "as is," and is subject to being changed, without notice, in future editions. Further, to the maximum extent permitted by applicable law, Agilent disclaims all warranties, either express or implied, with regard to this manual and any information contained herein, including but not limited to the implied warranties of merchantability and fitness for a particular purpose. Agilent shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, use, or performance of this document or of any information contained herein. Should Agilent and the user have a separate written agreement with warranty terms covering the material in this document that conflict with these terms, the warranty terms in the separate agreement shall control.

Technology Licenses

The hardware and/or software described in this document are furnished under a license and may be used or copied only in accordance with the terms of such license.

WARNING

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

CAUTION

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

Trademark Acknowledgements

Windows and MS Windows are U.S. registered trademarks of Microsoft Corporation.