
Programmer's Guide

Publication Number 54645-97000
June 1996 (pdf version Nov 1998)

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

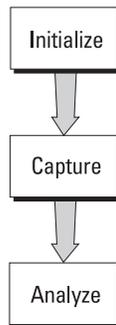
© Copyright Hewlett-Packard Company 1996
All Rights Reserved

HP 54645A Oscilloscope and HP 54645D Mixed-Signal Oscilloscope

Programming the Oscilloscope

When you attach an interface module to the rear of the HP 54645 A/D 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. Both HP-IB (also known as GP-IB, IEEE-488) and RS-232-C interfaces are available.

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



Initialize

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

- Program initialization defines and initializes variables, allocates memory, or tests system configuration.
- Controller initialization ensures that the interface to the oscilloscope (either HP-IB 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 is most likely invalid.

To collect data, you use the `:DIGitize` command. This command clears the waveform buffers and starts the acquisition process. Acquisition continues until 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 start the oscilloscope running, then use a wait loop in your program to ensure that the oscilloscope has completed at least one acquisition before you make a measurement. HP 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

The *HP 54645A/D Programmer's Guide* is your introduction to programming the HP 54645A Oscilloscope or HP 54645D Mixed-Signal 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.

To program the HP 54645A/D, you need an interface module, such as the HP 54650A, 54651A, or 54652B. 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, "Programming over HP-IB," discusses the general considerations for programming the instrument over an HP-IB 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 *HP 54645A/D 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 *HP 54645A/D 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 *HP 54645A/D User and Service 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 HP 82341A interface for IBM PC-compatible computers).

1	Introduction to Programming	
2	Programming Getting Started	
3	Programming over HP-IB	
4	Programming over RS-232-C	
5	Programming and Documentation Conventions	
6	Status Reporting	
7	Installing and Using the Programmer's Reference	
8	Programmer's Quick Reference	
	Index	

Contents

1 Introduction to Programming

- Talking to the Instrument 1-3
- Program Message Syntax 1-4
- Combining Commands from the Same Subsystem 1-7
- Duplicate Mnemonics 1-7
- Query Command 1-8
- Program Header Options 1-9
- Program Data Syntax Rules 1-10
- Program Message Terminator 1-12
- Selecting Multiple Subsystems 1-12

2 Programming Getting Started

- Initialization 2-3
- Autoscale 2-4
- Setting Up the Instrument 2-4
- Example Program 2-5
- Using the DIGitize Command 2-6
- Receiving Information from the Instrument 2-8
- String Variables 2-9
- Numeric Variables 2-10
- Definite-Length Block Response Data 2-11
- Multiple Queries 2-12
- Instrument Status 2-12

3 Programming over HP-IB

- Interface Capabilities 3-3
- Command and Data Concepts 3-3
- Addressing 3-4
- Communicating Over the Bus 3-5
- Lockout 3-6
- Bus Commands 3-6

Contents

4 Programming over RS-232-C

- Interface Operation 4-3
- Cables 4-3
- Minimum Three-Wire Interface with Software Protocol 4-4
- Extended Interface with Hardware Handshake 4-5
- Configuring the Interface 4-7
- Interface Capabilities 4-8
- Communicating Over the RS-232-C Bus 4-9
- Lockout Command 4-10

5 Programming and Documentation Conventions

- Command Set Organization 5-3
- The Command Tree 5-6
- Truncation Rules 5-10
- Infinity Representation 5-11
- Sequential and Overlapped Commands 5-11
- Response Generation 5-11
- Notation Conventions and Definitions 5-12
- Program Examples 5-13

6 Status Reporting

- Status Reporting Data Structures 6-5
- Status Byte Register (SBR) 6-9
- Service Request Enable Register (SRER) 6-11
- Trigger Event Register (TRG) 6-11
- Standard Event Status Register (SESR) 6-12
- Standard Event Status Enable Register (SESER) 6-13
- User Event Register (UER) 6-14
- Local Event Register (LCL) 6-14
- Operation Status Register (OPR) 6-14
- Limit Test Event Register (LTER) 6-15
- Mask Test Event Register (MTER) 6-16
- Histogram Event Register (HER) 6-17
- Arm Event Register (ARM) 6-17
- Error Queue 6-18

Output Queue 6-19
Message Queue 6-19
Key Queue 6-19
Clearing Registers and Queues 6-19

7 Installing and Using the Programmer's Reference

To install the help file under Microsoft Windows 7-3
To use the help text and example program files 7-3
To get updated help and program files via the Internet 7-4
To start the help file 7-5
To navigate through the help file 7-5

8 Programmer's Quick Reference

Introduction 8-2
Conventions 8-3
Suffix Multipliers 8-3
Commands and Queries 8-4

Index

Introduction to Programming

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

To program the HP 54645A/D oscilloscope you must add either an HP-IB (HP 54650A, for example) or RS-232-C (HP 54651A, for example) interface to the rear panel.

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

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

Other tasks are accomplished by combining these basic functions.

Languages for Program Examples

The programming examples for individual commands in this manual are written in HP BASIC 5.0 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, HP BASIC uses the OUTPUT statement for sending commands and queries. After a query is sent, the response is usually read in using the ENTER statement.

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

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

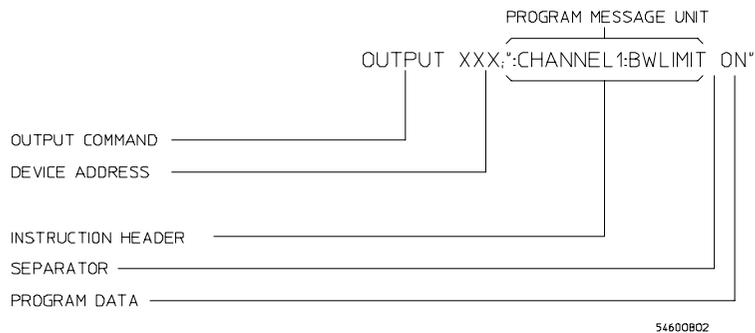
```
OUTPUT < device address > ;":ANALOG1:BWLIMIT ON"<terminator>
```

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. Figure 1 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, HP BASIC 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` in order to convert the examples. The instructions listed in this manual are always shown between quotation marks in the example programs.

Device Address

The location where the device address must be specified is also dependent on the programming language you are using. In some languages, this may be specified outside the output command. In 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 learnstring. There are only a few instructions which use block data.

Instructions are composed of two main parts:

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

Instruction Header

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

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

White Space (Separator)

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

Program Data

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

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

Header Types

There are three types of headers:

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

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

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

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

When program data must be included with the simple command header (for example, :DIGITIZE ANALOG1), 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 :ANALOG1:BWLIMIT ON)

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

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

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

Combining Commands from the Same Subsystem

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

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

(For example :ANALOG1:COUPLING DC;BWLIMIT ON)

Duplicate Mnemonics

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

```
: ANALOG1 : 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.

ANALOG1 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 causes the output buffer to be cleared and the current response to be lost. This also generates a query interrupted error in the error queue.

Program Header Options

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

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

```
TIMEBASE:DELAY 1US - long form
```

```
TIM:DEL 1US - short form
```

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

Command Syntax Programming Rules

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

Program Data Syntax Rules

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

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

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

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

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

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

Character Program Data

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

The available mnemonics for character program data are always included with the instruction's syntax definition. When sending commands, either the long form or short form (if one exists) may be used. 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 would be ignored, truncating the number. Numeric data parameters which accept fractional values are called real numbers.

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

Embedded Strings

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

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

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

Program Message Terminator

The program instructions within a data message are executed after the program message terminator is received. The terminator may be either an NL (New Line) character, an EOI (End-Of-Identify) asserted in the HP-IB 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>  
:ANALOG1:RANGE 0.4::TIMEBASE:RANGE 1
```

Combining Compound and Simple Commands

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



Programming Getting Started

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 HP BASIC 5.0 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 HP-IB, 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 *HP 54645A/D Oscilloscopes Programmer's Reference*.

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

Autoscale

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

The syntax for the autoscale function is:

```
:AUTOSCALE<terminator>
```

Setting Up the Instrument

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

```
:ANALOG1:PROBE X10;RANGE 16;OFFSET 1.00<terminator>  
:TIMEBASE:MODE NORMAL;RANGE 1E-3;DELAY 100E-6<terminator>
```

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

Example Program

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

```

10      CLEAR 707                                ! Initialize instrument interface
20      OUTPUT 707;"*RST"                        ! Initialize inst to preset state
30      OUTPUT 707;" :TIMEBASE:RANGE 5E-4"      ! Time base to 50 us/div
40      OUTPUT 707;" :TIMEBASE:DELAY 0"         ! Delay to zero
50      OUTPUT 707;" :TIMEBASE:REFERENCE CENTER" ! Display reference at center
60      OUTPUT 707;" :ANALOG1:PROBE X10"        ! Probe attenuation to 10:1
70      OUTPUT 707;" :ANALOG1:RANGE 1.6"       ! Vertical range to 1.6 V full scale
80      OUTPUT 707;" :ANALOG1:OFFSET -.4"      ! Offset to -0.4
90      OUTPUT 707;" :ANALOG1:COUPLING DC"     ! Coupling to DC
100     OUTPUT 707;" :TRIGGER:MODE NORMAL"      ! Normal triggering
110     OUTPUT 707;" :TRIGGER:LEVEL -.4"       ! Trigger level to -0.4
120     OUTPUT 707;" :TRIGGER:SLOPE POSITIVE"  ! Trigger on positive slope
130     OUTPUT 707;" :ACQUIRE:TYPE NORMAL"    ! Normal acquisition
140     OUTPUT 707;" :DISPLAY:GRID OFF"        ! Grid off
150     END

```

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

Using the DIGitize Command

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

Ensure New Data is Collected

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

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

Set :TIMEbase:MODE to NORMAl when using :DIGitize

:TIMEbase:MODE must be set to NORMAl to perform a :DIGitize 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 ANALOG1"<terminator>
OUTPUT 707; ":WAVEFORM:FORMAT BYTE"<terminator>
OUTPUT 707; ":ACQUIRE:COUNT 8"<terminator>
OUTPUT 707; ":WAVEFORM:POINTS 500"<terminator>
OUTPUT 707; ":DIGITIZE ANALOG1"<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 *HP 54645AD Oscilloscopes Programmer's Reference*.

Aborting a Digitize Operation Over HP-IB

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

Receiving Information from the Instrument

After receiving a query (command header followed by a question mark), the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the answer is transmitted across the interface to the designated listener (typically a controller). The input statement for receiving a response message from an instrument's output queue typically has two parameters; the device address, and a format specification for handling the response message. For example, to read the result of the query command :ANALOG1: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 causes the output buffer to be cleared and the current response to be lost. This also causes an error to be placed in the error queue.

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

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

String Variables

The output of the instrument may be numeric or character data depending on what is queried. Refer to the specific commands for the formats and types of data returned from queries.

Express String Variables Using Exact Syntax

In HP BASIC 5.0, 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$[30]
20 OUTPUT 707;" :ANALOG1:RANGE?"
30 ENTER 707;Rang$
40 PRINT Rang$
50 END
```

After running this program, the controller displays:

```
+8.00000E-01
```

Numeric Variables

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

```
10 OUTPUT 707;" :ANALOG1:RANGE?"  
20 ENTER 707;Rang  
30 PRINT Rang  
40 END
```

After running this program, the controller displays:

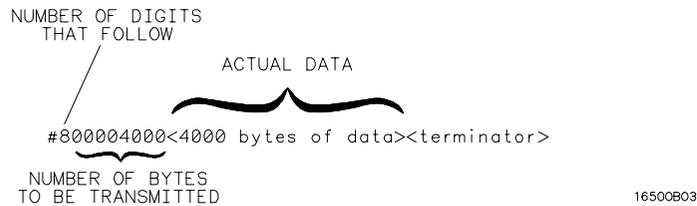
```
.8
```

Definite-Length Block Response Data

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

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

Figure 2-1



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

Multiple Queries

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

```
ENTER 707;Results$
```

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

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

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

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

Instrument Status

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



Programming over HP-IB

This section describes the HP-IB 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 HP-IB interface card you are using.

Interface Capabilities

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

Command and Data Concepts

The interface has two modes of operation:

- command mode
- data mode

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

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

Addressing

By using the front-panel menus, you must place the instrument in the “Connect to Computer” mode. Use the Print/Utility menu key on the front panel to do this after you have installed the HP-IB interface on the rear panel of the oscilloscope. Use this menu to set the HP-IB address for the oscilloscope.

- Each device on the HP-IB 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 HP-IB 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 HP-IB controllers.

Instrument Address (Selects Instrument)

Each instrument on an HP-IB 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 *HP 54645A/D Oscilloscopes Programmer's Reference* assume the oscilloscope is at device address 707.

See the documentation for your HP-IB interface card for more information on select codes and addresses.

Lockout

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

Restore Front-Panel Control

Cycling power also restores front panel control.

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

Bus Commands

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

Device Clear

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

Interface Clear (IFC)

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

Programming over RS-232-C

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

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

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

Interface Operation

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

When operating directly between two RS-232-C devices, certain considerations must be taken into account. For three-wire operation, an XON/XOFF software handshake must be used to handle handshaking between the devices. For extended hardware 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 hardware operation, the DCD and DSR inputs to the oscilloscope must remain high for proper operation.

With extended hardware 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 since you can ignore hardware handshake requirements. The oscilloscope uses the following connections on its RS-232-C interface for three-wire communication:

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

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

No Hardware Means to Control Data Flow

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

Extended Interface with Hardware Handshake

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

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

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

Programming over RS-232-C
Extended Interface with Hardware Handshake

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

By using the front-panel controls, you must place the RS-232-C interface in either the printer mode or the controller mode. Use the printer mode when you want the instrument to talk directly to a printer over RS-232-C without the aid of a controller. Use the controller mode when the instrument will operate in conjunction with a controller over RS-232-C.

Use the Print/Utility menu key on the front panel to do this after you have installed the RS-232-C interface on the rear panel of the oscilloscope. Use this menu to “Connect to Computer,” and assign the baud rate and handshake protocol.

Interface Capabilities

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

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

Protocol

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

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

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

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

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

Data Bits

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

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

Communicating Over the RS-232-C Bus

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

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

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

Lockout Command

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

Restoring Local Control

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

Programming and Documentation
Conventions

Programming and Documentation Conventions

This chapter covers conventions used in programming the instrument, as well as conventions used in the online *HP 54645A/D Oscilloscopes Programmer's Reference* and the remainder of this manual. This chapter also contains a detailed description of the command tree and command tree traversal.

Command Set Organization

The command set is divided into common commands, root level commands and sets of subsystem commands. Each of the groups of commands is described in the *HP 54645A/D Oscilloscopes Programmer's Reference*, which is supplied as an online help file for Microsoft Windows. See the chapter "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
ANALog	controls the channel display and vertical or Y-axis of the oscilloscope
CALibrate	provides utility commands for determining the state of the calibration factor protection switch
CHANnel	controls all oscilloscope functions associated with individual channels or groups of channels
Common	commands defined by IEEE 488.2 standard common to all instruments
DISPlay	controls how waveforms, graticule, and text are displayed and written on the screen
FUNction	controls functions in the Measurement/Storage Module
HARDcopy	provides commands to set and query the selection of hardcopy device and formatting options
MASK	controls the waveform monitoring test function available when using the Measurement/Storage Module
MEASure	selects automatic measurements to be made and controls time markers
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
TEST	allows instrument self-tests to be performed via the remote interface
TIMebase	controls all horizontal sweep functions
TRACe	controls features used with trace memories
TRIGger	controls the trigger modes and parameters for each trigger type
WAVEform	provides access to waveform data

Programming and Documentation Conventions
Command Set Organization

Table 5-1

Alphabetic Command Reference

Command	Subsystem Where Used	Command	Subsystem Where Used	Command	Subsystem Where Used
ACTivity	CHANnel	DSP	SYSTem	LABel	CALibrate
ADDReSS	HARDcopy	DUTycle	MEASure	LABel	CHANnel
ADVanced	TRIGger			LABEL	DISPlay
AER	Root level	EDGE	TRIGger	LABList	DISPlay
ALL	TEST	EDGE1	TRIGger:ADVanced	LESSthan	TRIGger:ADVanced
ASTore	Root level	EDGE2	TRIGger:ADVanced	LESSthan	TRIGger:GLITch
AUToscale	Root level	*EMC	Common	LEVEL	TRIGger:EDGE
		ERASe	Root level	LEVEL	TRIGger:GLITch
BLANK	Root level	ERRor	SYSTem	LINE	DISPlay
BWLimit	ANALog	*ESE	Common	LINE	TRIGger:TV
BYTeorder	WAVeform	*ESR	Common	*LMC	Common
				LOCK	SYSTem
CENTer	FUNCTion	FACTors	HARDcopy	*LRN	Common
CLEAR	TRACe	FAILmode	MASK		
*CLS	Common	FALLtime	MEASure	MATH	CHANnel
COLumn	DISPlay	FFEd	HARDcopy	MERGe	Root level
COMPLete	ACQuire	FIELD	TRIGger:TV	MODE	TIMEbase
CONNect	DISPlay	FORMat	WAVeform	MODE	TRACe
COUNT	ACQuire	FREQuency#	MEASure	MODE	TRIGger
COUNT	WAVeform			MOVE	FUNCTion
COUPling	ANALog	GLITch	TRIGger		
COUPling	TRIGger:EDGE	*GMC	Common	NREJect	TRIGger
CREATE	MASK	GRAYscale	HARDcopy	NUMBer	MASK
		GREaterthan	TRIGger:ADVanced	NWIDth#	MEASure
DATA	DISPlay	GREaterthan	TRIGger:GLITch		
DATA	MASK	GRID	DISPlay	OFFSet	ANALog
DATA	TRACe			OFFSet	FUNCTion
DATA	WAVeform	HOLDoff	TRIGger	*OPC	Common
DEFine	MEASure			OPEE	Root level
DELAy#	MEASure	*IDN	Common	OPER	Root level
DELAy	TIMEbase	INCRement	MASK	OPERation	FUNCTion<N>
DESTination	MASK	INVERse	DISPlay	OPERator	TRIGger:ADVanced
DEVIce	HARDcopy	INVert	ANALog	*OPT	Common
DIGitize	Root level			ORDer	DISPlay
DITHer	ACQuire	KEY	SYSTem	OVERshoot	MEASure
*DMC	Common				
		LABel	ANALog	PATTern	TRIGger
				PATTern1	TRIGger:ADVanced

These commands are used by the HP 54645A only.

Command	Subsystem Where Used	Command	Subsystem Where Used	Command	Subsystem Where Used
PATtern2	TRIGger:ADVanced	SOURce	FUNcTion	VBASe#	MEASure
PEAKs	FUNcTion	SOURce	MEASure	VDELta	MEASure
PERiod	MEASure	SOURce1	TRIGger:ADVanced	VIEW	FUNcTion<N>
PHASe#	MEASure	SOURce2	TRIGger:ADVanced	VIEW	Root level
PIXel	DISPlay	SOURce	TRIGger:EDGE	VIR#	TRIGger:TV
*PMC	Common	SOURce	TRIGger:GLITch	VMAX#	MEASure
PMODe	ANALog	SOURce	TRIGger:TV	VMIN#	MEASure
POINts	ACQuire	SOURce	WAVeform	VPP#	MEASure
POINts	WAVeform	SPAN	FUNcTion	VRMS#	MEASure
POLarity	TRIGger:TV	*SRE	Common	VSTArt	MEASure
POLarity	TRIGger:GLITch	STANdard	TRIGger:TV	VSTOp	MEASure
POSition	DISPlay	STATistics	MASK	VTIMe	MEASure
POSTfailure	MASK	STATus	Root level	VTOP	MEASure
PREAmble	WAVeform	*STB	Common		
PREShoot#	MEASure	STOP	Root level	*WAI	Common
PRINt	Root level	SWITch	CALibrate	WINDow	FUNcTion
PROBe	ANALog				
PWIDth#	MEASure				
		TDELta	MEASure	XINCrement	WAVeform
QUALifier	TRIGger:GLITch	TER	Root level	XORigin	WAVeform
		TEST	MASK	XREFerence	WAVeform
		TEXT	DISPlay		
RANGe	ANALog	THReshold	CHANnel	YINCrement	WAVeform
RANGe	FUNcTion<N>	THReshold	TRIGger	YORigin	WAVeform
RANGe	TIMebase	TOLerance	MASK	YREFerence	WAVeform
RANGe	TRIGger:ADVanced	*TRG	Common		
RANGe	TRIGger:GLITch	*TST	Common		
*RCL	Common	TSTArt	MEASure		
REFerence	FUNcTion	TSTOp	MEASure		
REFerence	TIMebase	TV	TRIGger		
REJect	TRIGger	TVHFrej	TRIGger:TV		
RISetime#	MEASure	TVMode	TRIGger:TV		
ROW	DISPlay	TVOLT	MEASure		
*RST	Common	TYPE	ACQuire		
RUN	Root level	TYPE	WAVeform		
*SAV	Common	VAMPplitude#	MEASure		
SAVE	MASK	VAUToscale	Root level		
SAVE	TRACe	VAVerage#	MEASure		
SCRatch	MEASure				
SETup	SYSTem				
SHOW	MEASure				
SINGle	Root level				
SLOPe	TRIGger:EDGE				
SOURce	DISPlay				

These commands are used by the HP 54645A only.

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 can be placed into three types:

- Common commands
- Root level commands
- Subsystem commands

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

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

Example:

```
*RST
```

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

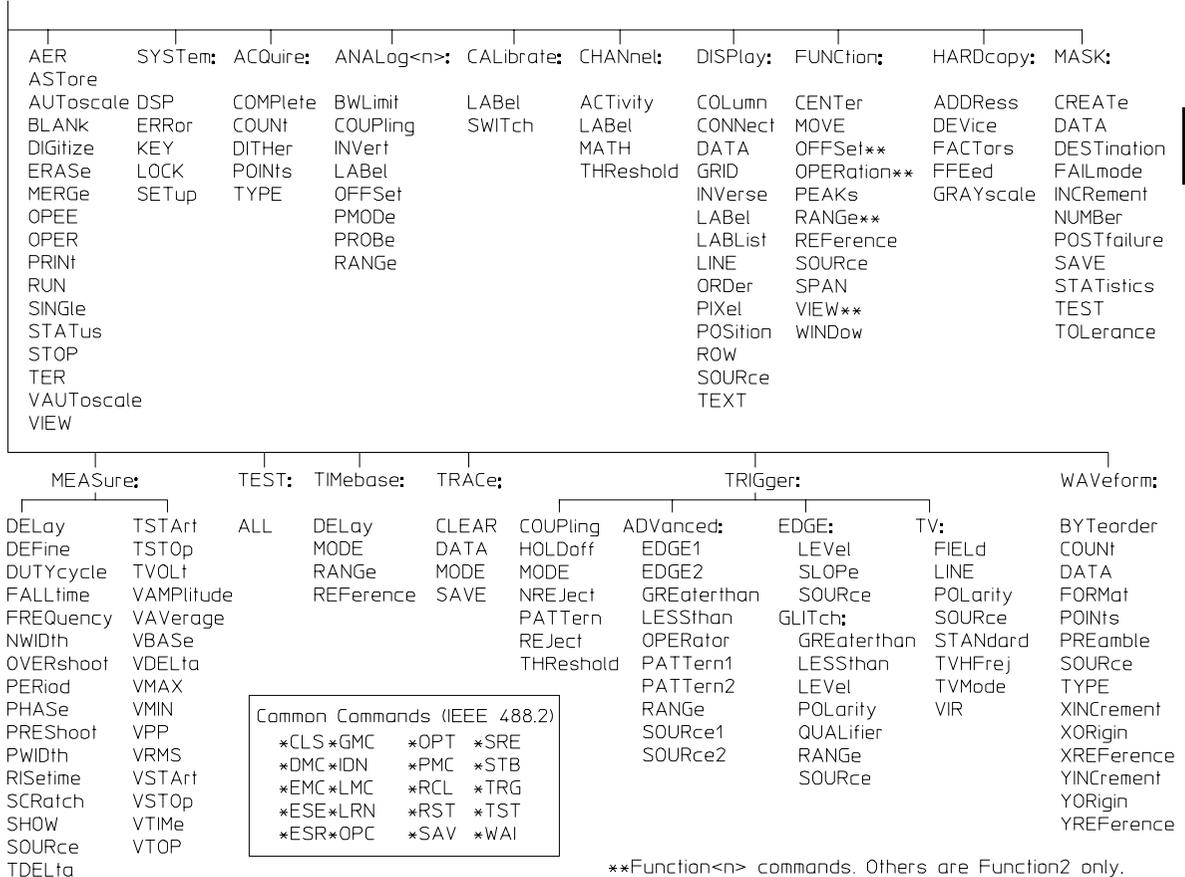
Example:

```
:AUTOSCALE
```

Programming and Documentation Conventions

The Command Tree

:root)



Note: Some of these commands are specific to either the HP 54645A or the HP 54645D.
Consult the online programmer's reference for more information.

54645b06

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 :ANALOG1: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 (ANALOG<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 HP BASIC 5.0. The quoted string is placed on the bus, followed by a carriage return and linefeed (CRLF).

Example 1:

```
OUTPUT 707;" :ANALOG1:RANGE 0.5 ;OFFSET 0"
```

The colon between ANALOG1 and RANGE is necessary because ANALOG1: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 ANALOG1 preceding it, since the ANALOG1:RANGE command sets the parser to the ANALOG1 node in the tree.

Example 2:

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

or

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

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

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

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

Example 3:

```
OUTPUT 707; ":TIMEBASE:REFERENCE CENTER ; :ANALOG1:OFFSET '0'"
```

The leading colon before ANALOG1 tells the parser to go back to the root of the command tree. The parser can then see the ANALOG1:OFFSET command.

Truncation Rules

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

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

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

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

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

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 *HP 54645A/D Oscilloscopes Programmer's Reference* in descriptions of remote operation:

Conventions

- < >** Angle brackets enclose words or characters that symbolize a program code parameter or an interface command.
- ::=** “is defined as.” For example, `<A> ::= ` indicates that `<A>` can be replaced by `` in any statement containing `<A>`.
 - | “or.” Indicates a choice of one element from a list. For example, `<A> | ` indicates `<A>` or ``, but not both.
- ...** An ellipsis (trailing dots) indicates that the preceding element may be repeated one or more times.
- []** Square brackets indicate that the enclosed items are optional.
- { }** When several items are enclosed by braces, one, and only one of these elements must be selected.

Definitions

- d ::=** A single ASCII numeric character, 0-9.
- n ::=** A single ASCII non-zero, numeric character, 1-9.
- <NL> ::=** Newline or Linefeed (ASCII decimal 10).
- <sp> ::=** <white space>
- <white space>**
 - ::=** 0 through 32 (decimal) except linefeed (decimal 10). The nominal value is 32 (the space character).

Program Examples

The program examples given for commands in the online *HP 54645A/D Oscilloscopes Programmer's Reference* were written using the HP BASIC 5.0 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; ":ANALOG1:RANGE .1"
- Commands in short form using an exponential format.
OUTPUT 707; ":ANAL1:RANG 1E-1"
- Commands using lower case letters, short forms, and a suffix.
OUTPUT 707; ":anal1: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.



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.

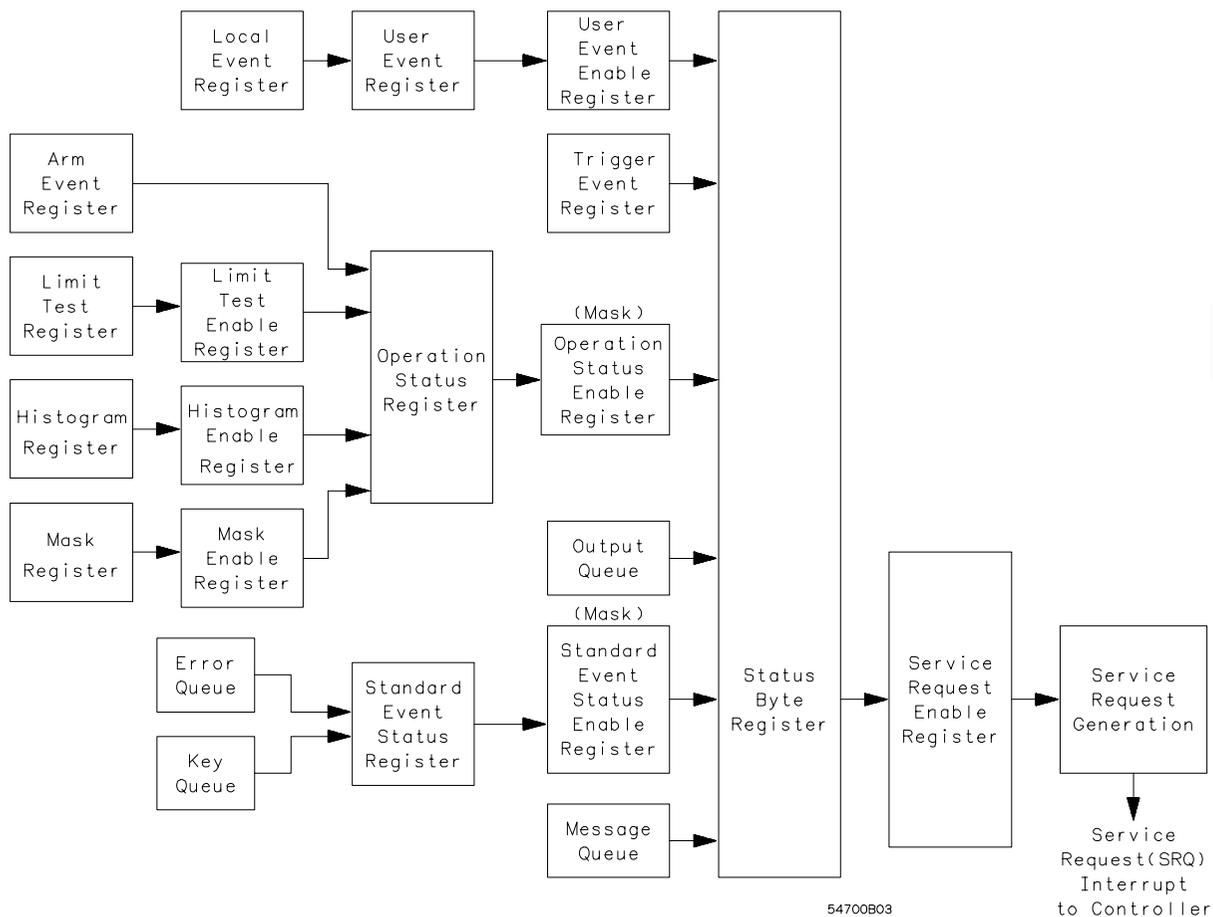
- To monitor an event, first clear the event, then enable the event. All of the events are cleared when you initialize the instrument.
- To generate a service request (SRQ) interrupt to an external controller, enable at least one bit in the Status Byte Register.

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

Figure 6-1



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.
USR	User Event Register	If any of the enabled conditions have occurred in the User Event Register.
TRG	Trigger	Whether a trigger has been received.
LCL	Local	If a remote-to-local transition occurs.
FAIL	Fail	That the specified test has failed.
COMP	Complete	That the specified test has completed.
LTEST	Limit Test	If any of the enabled conditions have occurred in the Limit Test Register.
MTEST	Mask Test	If any of the enabled conditions have occurred in the Mask Test Register.
HIST	Histogram	If any of the enabled conditions have occurred in the Histogram Register.
WAIT TRIG	Wait for Trigger	Instrument is armed and ready for trigger.

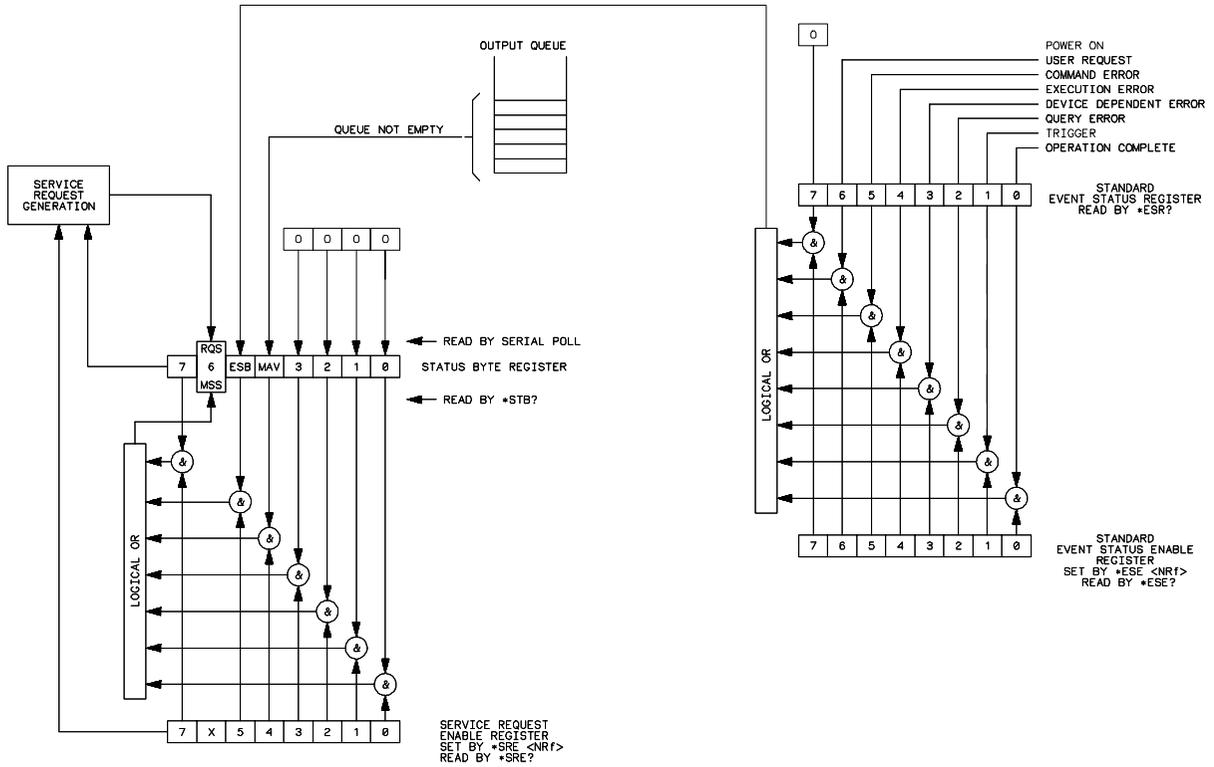
Status Reporting Data Structures

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.

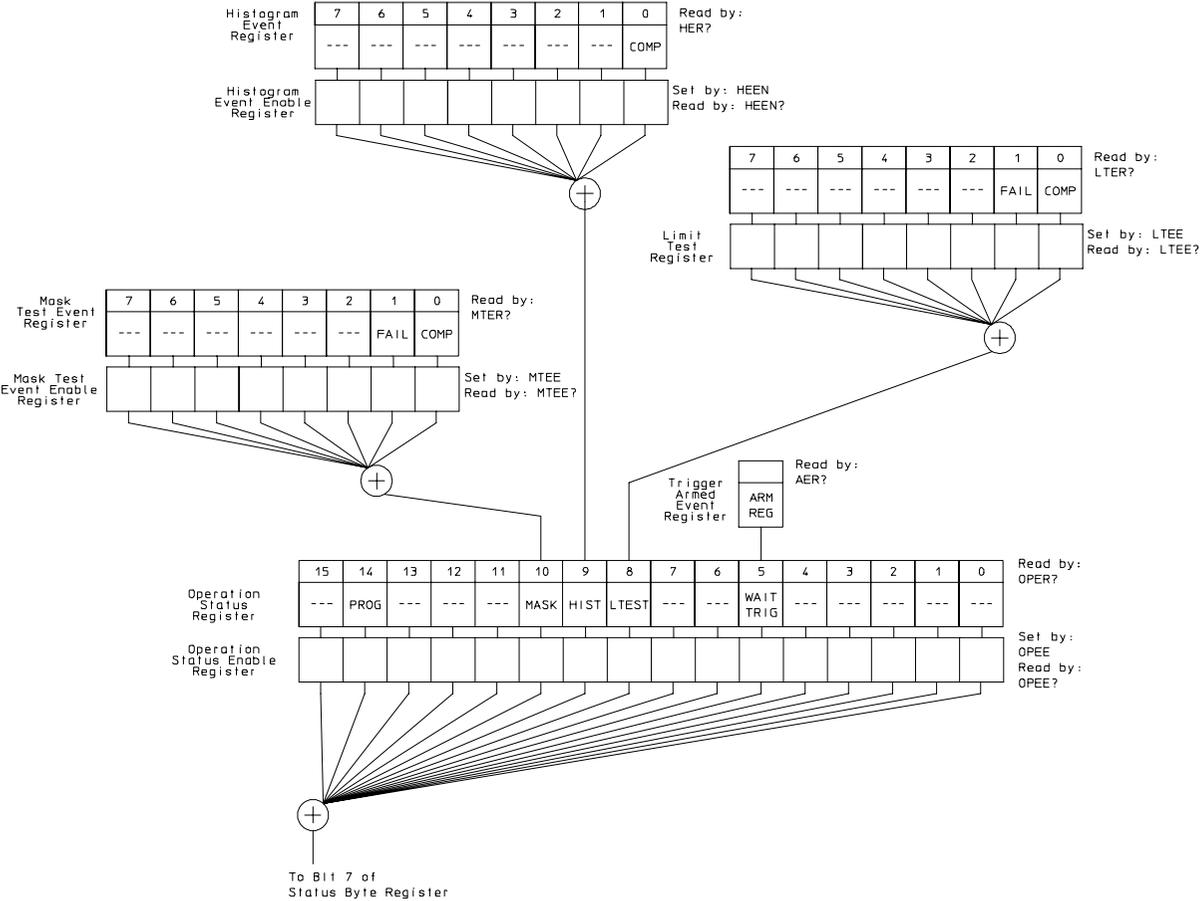
Status Reporting Status Reporting Data Structures

Figure 6-2



Status Reporting Data Structures

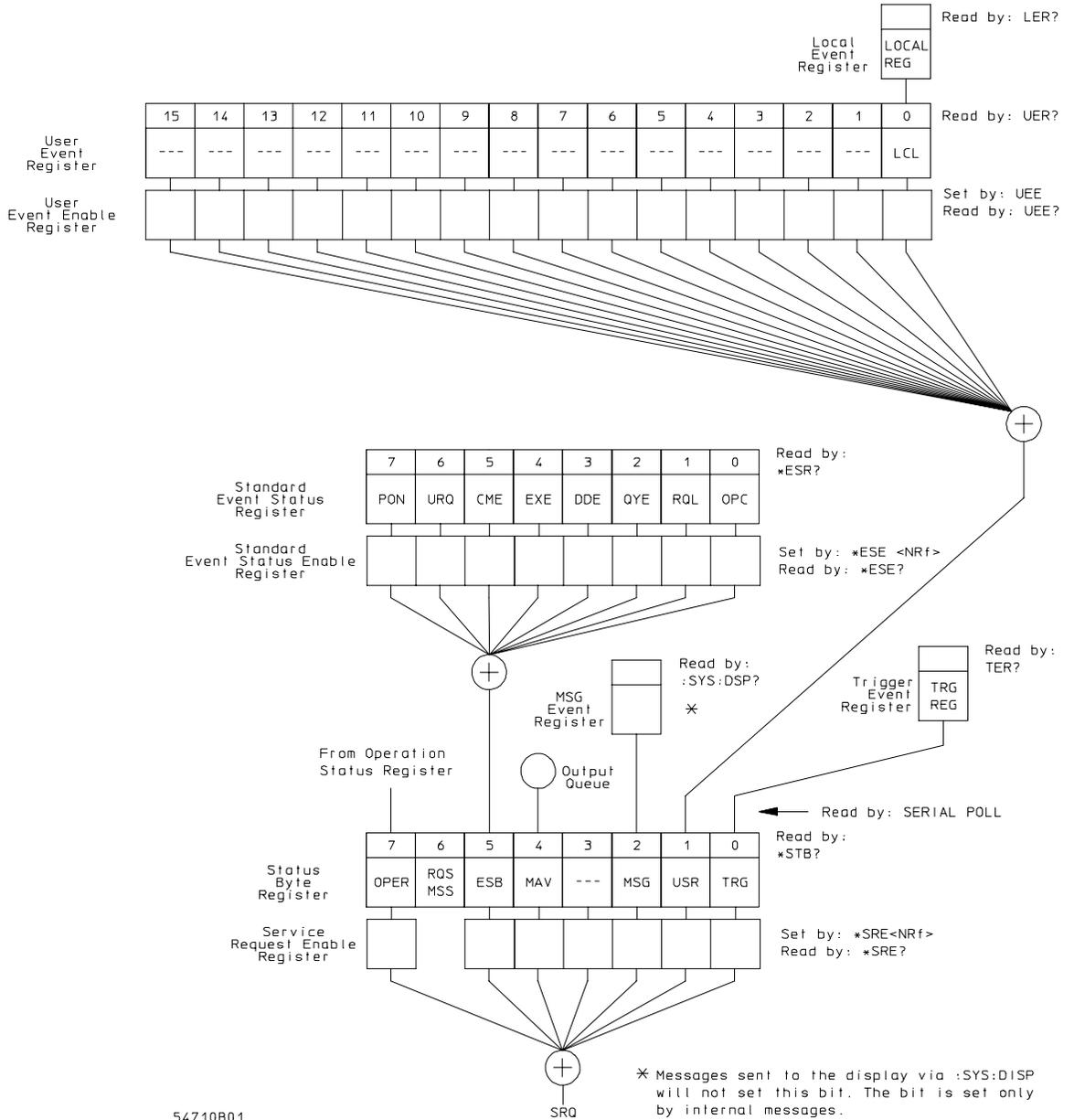
Figure 6-2 (Continued)



Status Reporting Data Structures

Status Reporting Status Reporting Data Structures

Figure 6-2 (Continued)



54710B01

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 HP-IB serial poll command. Both commands return the decimal-weighted sum of all set bits in the register. The difference between the two methods is that the 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.

Status Reporting

Status Byte Register (SBR)

Example

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

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

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

Example

The following example uses the HP BASIC serial poll (SPOLL) command 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, each time the trigger bit is set you must then clear the event register.

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.

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

Example

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

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

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

Standard Event Status Enable Register (SESER)

To make it possible for any of the Standard Event Status Register (SESR) bits to be able to generate a summary bit, first enable the bit. Enable the bit by using the *ESE (Event Status Enable) common command to set the corresponding bit in the Standard Event Status Enable Register (SESER).

Set bits are read with the *ESE? query.

Example

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

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

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

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

Function of SESR Bits

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

User Event Register (UER)

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

Local Event Register (LCL)

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

Operation Status Register (OPR)

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

- The WAIT TRIG bit is set by the Trigger Armed Event Register and indicates that the trigger is armed.
- The LTEST bit is set when a limit test fails or is completed and sets the corresponding FAIL or COMP bits in the Limit Test Event Register.
- The HIST bit is set when the COMP bit is set in the Histogram Event Register, indicating that the histogram measurement has satisfied the specified completion criteria.
- The MASK bit is set when the Mask Test either fails specified conditions or satisfies its completion criteria, setting the corresponding FAIL or COMP bits in the Mask Test Event Register.
- The PROG bit is reserved for future use.
- If any of these bits are set, the OPER bit (bit 7) of the Status Byte Register is set. The Operation Status Register is read and cleared with the OPER? query. The register output is enabled or disabled using the mask value supplied with the OPPE command.

Limit Test Event Register (LTER)

Bit 0 (COMP) of the Limit Test Event Register is set when the Limit Test completes. The Limit Test completion criteria are set by the LTEST:RUN command.

Bit 1 (FAIL) of the Limit Test Event Register is set when the Limit Test fails. Failure criteria for the Limit Test are defined by the LTEST:FAIL command.

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

When either the COMP or FAIL bits are set, they in turn set the LTEST bit (bit 8) of the Operation Status Register. You can mask the COMP and FAIL bits, thus preventing them from setting the LTEST bit, by defining a mask using the LTEE command.

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

Mask Test Event Register (MTER)

Bit 0 (COMP) of the Mask Test Event Register is set when the Mask Test completes. The Mask Test completion criteria are set by the MTEST:RUMode command.

Bit 1 (FAIL) of the Mask Test Event Register is set when the Mask Test fails. This will occur whenever any sample is recorded within any polygon defined in the mask.

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

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

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

Histogram Event Register (HER)

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

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

Arm Event Register (ARM)

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

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

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

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.

Key Queue

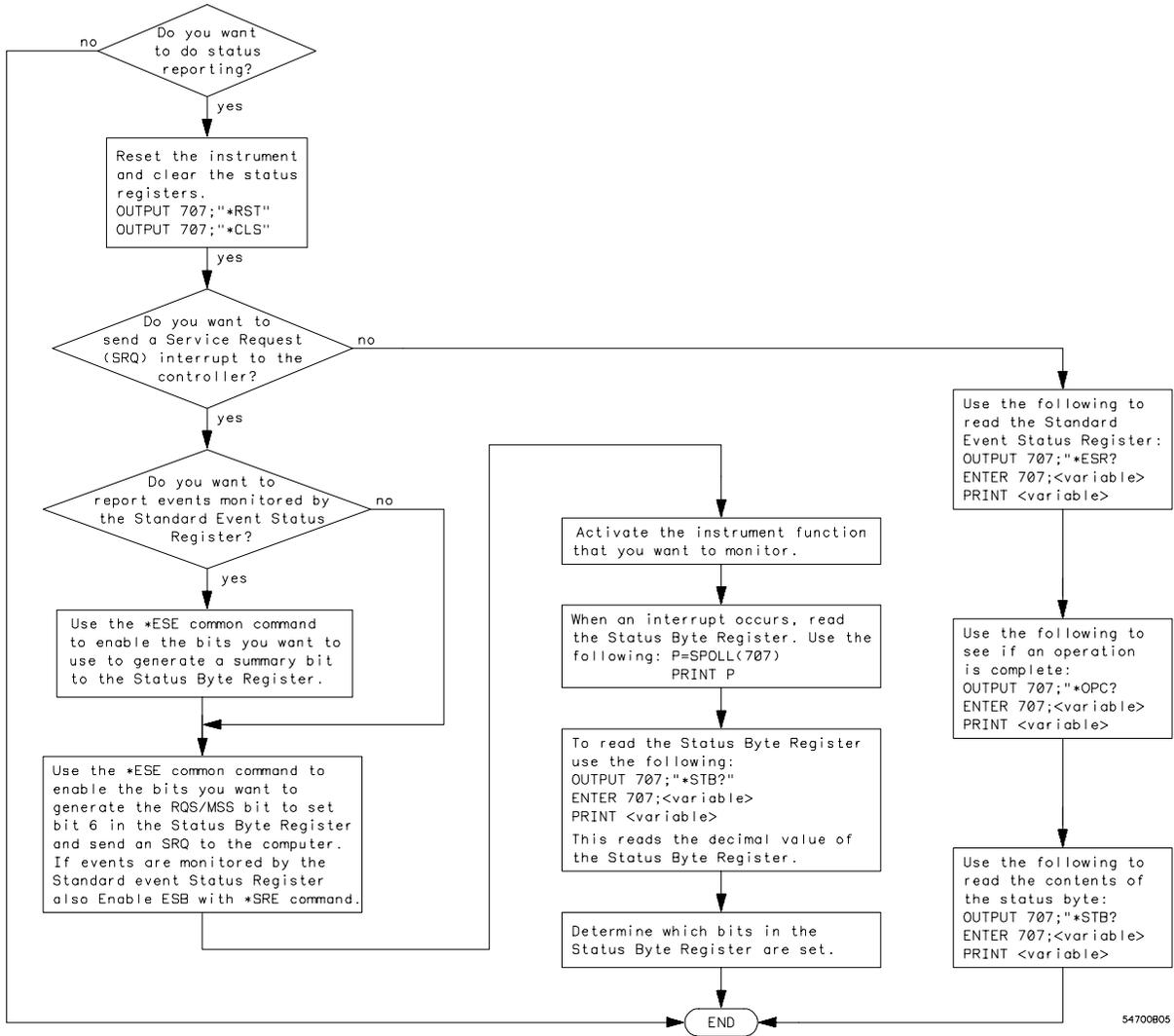
The key queue contains the key codes for the last 10 keys pressed on the front panel. This queue is first in, first out. If the key queue overflows, the oldest key codes are discarded as additional keys are pressed. The key queue is read with the SYSTEM:KEY? query.

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.

Status Reporting Clearing Registers and Queues

Figure 6-3



54700805

Status Reporting Decision Chart

Installing and Using the Programmer's Reference

Installing and Using the Programmer's Reference

The HP 54645A/D Oscilloscopes Programmer's Reference is supplied as an online help file readable with the Microsoft Windows 3.1 help viewer. A second diskette contains the help file as an ASCII text file and sample programs for the HP 54645A/D Oscilloscopes. This chapter explains how to install the help file on your system, discusses the text and program files, and explains how you can get the programs and help file via the Internet.

To install the help file under Microsoft Windows

The help file requires Microsoft Windows 3.1 and MS-DOS 3.3 or greater running on an IBM-compatible PC. The file uses the Microsoft Windows help viewer, WINHELP.EXE.

- 1 Start your PC and start Microsoft Windows.
- 2 Insert the 3.5" floppy disk labeled "HP 54645A/D Oscilloscopes Programmer's Reference/Example Programs for Windows 3.1 Applications" into the appropriate diskette drive (A: or B:) of your PC.
- 3 Select **File | Run** from the Program Manager, then type in the following:

```
<drive>:\install
```

where **<drive>** is either **A:** or **B:**.

- 4 Follow the instructions onscreen to complete the installation.

The installer copies the help file to a directory named `\hp54645`. You can choose a different directory if desired. It also creates a Program Manager group and icon that you can use to open the help file with the Microsoft Windows help viewer.

To use the help text and example program files

The help file is available as an ASCII text file that can be browsed with a text editor or text search tools. Also, there are sample programs (in ASCII text format) that show how to use the HP 54645 commands with the oscilloscope.

- Copy the help file text or sample programs from the diskette labeled "HP 54645 Oscilloscopes Programmer's Reference/Example Programs for Non-Windows Applications" to your system for viewing. See the file README.TXT on the diskette for more information about the diskette contents.

To get updated help and program files via the Internet

The latest versions of the help and example program files are available via anonymous ftp. You must have a connection to the Internet and have ftp software.

- 1** Log on to your Internet service.
- 2** Connect to host `col.hp.com` using anonymous ftp.

A sample set of commands might be as follows:

```
$ ftp col.hp.com
Name: anonymous
Password: myname@mycompany.com
```

- 3** Change to the directory containing HP 54645 files.

```
ftp> cd dist/hp54600/hp54645
```

- 4** Get sample programs or updated help files from the directory as desired.

For example, if you want the latest version of the HP 54645A/D Oscilloscopes Programmer's Reference online help file, you set the transfer mode to binary and get the file:

```
ftp> binary
ftp> get hp54645.hlp
```

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

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

To start the help file

- To open the help file under Microsoft Windows, double-click the “Programmer’s Reference” icon in the “HP 54645A/D Oscilloscopes Programmer’s Reference” program group in the Program Manager. The help file requires the program WINHELP.EXE for Microsoft Windows 3.1. The properties for the Program Manager icon are set to expect this file in the Windows directory.

To navigate through the help file

- Navigate through the help file by clicking on highlighted text and buttons.
See your Microsoft Windows documentation for more information, or select **Help | How to Use Help** in the Help window.

Introduction

The Programmer's Quick Reference provides the commands and queries with their corresponding arguments and returned formats for the HP 54645A/D 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 HP-IB command.
::= "is defined as."	<A>::= indicates that <A> can be replaced by in any statement containing <A>.
"or"	Indicates a choice of one element from a list. For example, <A> indicates <A> or 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 *HP 54645A/D Oscilloscopes Programmer's Reference*.

Commands and Queries

The following tables facilitate easy access to each command and query for the HP 54645A/D 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 the command operates on a particular oscilloscope model. For additional information, refer to the online Oscilloscopes Programmer's Reference.

Command	Query	Options and Query Returns
:ACQUIRE:COMPLETE <complete_argument>	:ACQUIRE:COMPLETE?	<complete_argument> ::= 0 to 100; an integer in NR1 format
:ACQUIRE:COUNT <count_argument>	:ACQUIRE:COUNT?	<count_argument> ::= 4, 8, 16, 32, 64, 128, or 256; an integer in NR1 format
:ACQUIRE:DITHER	:ACQUIRE:DITHER	{ON OFF}
n/a	:ACQUIRE:POINTS?	1,000 to 2,000,000; an integer in NR1 format.
:ACQUIRE:TYPE <acq_type>	:ACQUIRE:TYPE?	<acq_type> ::= {NORMAL AVERAGE PEAK REALTIME}
n/a	:AER?	0 1; an integer in NR1 format
:ANALOG<n>:BWLIMIT {ON OFF}	:ANALOG<n>:BWLIMIT?	{ON OFF} <n> ::= 1 or 2; an integer in NR1 format
:ANALOG<n>:COUPLING {AC DC GND}	:ANALOG<n>:COUPLING?	{AC DC GND} <n> ::= 1 or 2; an integer in NR1 format
:ANALOG<n>:INVERT {ON OFF}	:ANALOG<n>:INVERT?	{ON OFF} <n> ::= 1 or 2; an integer in NR1 format
:ANALOG<n>:LABEL <string>	:ANALOG:LABEL?	<string> ::= any series of ASCII characters enclosed in quotation marks <n> ::= 1 or 2
:ANALOG<n>:OFFSET <offset_argument>	:ANALOG<n>:OFFSET?	<offset_argument> ::= offset value in volts in <NR3> format. <n> ::= 1 or 2; an integer in NR1 format
:ANALOG<n>:PMODE {AUT MANUAL}	:ANALOG<n>:PMODE?	{AUT MAN} <n> ::= 1 or 2; an integer in NR1 format
:ANALOG<n>:PROBE <attenuation>	:ANALOG<n>:PROBE?	<attenuation> ::= X1, X10, X20, X100 for all oscilloscope models <n> ::= 1 or 2; an integer in NR1 format
:ANALOG<n>:RANGE <range_argument>	:ANALOG<n>:RANGE?	<range_argument> ::= Full-scale range value for channels 1 or 2 in NR3 format.

Command	Query	Options and Query Returns
:ASTore	n/a	n/a
:AUToscale	n/a	n/a
:BLANK <source>	n/a	<source> ::= {ANALog<n> DIGital0,...,DIGital15 PMEM{1 2}} <n> ::= 1 or 2; an integer in NR1 format
:CALibrate:LABel <string>	:CALibrate:LABel?	<string> ::= unquoted ASCII string up to 32 characters
n/a	:CALibrate:SWITCh?	{PROT UNPR}
:CHANnel:ACTivity	:CHANnel:ACTivity?	<edges> ::= 32-bit integer in NR1 format <levels> ::= 32-bit integer in NR1 format
:CHANnel:LABel <source text>, <string>	:CHANnel:LABel?	<source text> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} <string> ::= any series of ASCII characters enclosed in quotation marks
:CHANnel:MATH <operation>	:CHANnel:MATH?	<operation> ::= {OFF ADD SUBTRACT}
:CHANnel:THReshold <channel_group>,<threshold type>,[<value>]	:CHANnel:THReshold?	<channel group> ::= {POD1 POD2} <threshold type> ::= {CMOS ECL TTL USERdef} <value> ::= voltage type in NR3 format [Volt type] [Volt type] ::= {V MV (-3) UV (-6)}
*CLS	n/a	n/a
:DIGitize [<source>],[<source>]]	n/a	<source> ::= {ANALog1 ANALog2 POD1 POD2}
:DISPlay:COLumn <number>	:DISPlay:COLumn?	<number> ::= 0 through 63; an integer in NR1 format
:DISPlay:CONNect {ON OFF}	:DISPlay:CONNect?	{ON OFF}
:DISPlay:DATA <binary block data>	:DISPlay:DATA?	<binary block_data> ::= 16748 bytes of data in IEEE 488.2 # format
:DISPlay:GRID <value>	:DISPlay:GRID?	<value> ::= {OFF FRAME FULL} <value> ::= {OFF FRAME FULL TV} for HP 54645A with Option 005
:DISPlay:INVerse {ON OFF}	:DISPlay:INVerse?	{ON OFF}
:DISPlay:LABel <value>	DISPlay:LABel?	<value> ::= {ON OFF}
:DISPlay:LABList <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:LINE <string>	n/a	<string> ::= any series of ASCII characters enclosed in quotation marks
:DISPlay:ORDer <order list>	:DISPlay:ORDer?	<order list> ::= ::= DIGital<n>[,DIGital<n>] <n> ::= integer from 0 to 15

Programmer's Quick Reference Commands and Queries

Command	Query	Options and Query Returns																											
:DISPlay:PIXel <x>, <y>, <intensity>	:DISPlay:PIXel? <x>,<y>	<x> ::= x coordinate of the pixel to be set; an integer (0 to 511) in NR1 format <y> ::= y coordinate of the pixel to be set; an integer (0 to 303) in NR1 format <intensity> ::= an integer in NR1 format: 0 for pixel off 1 for pixel with half-bright on 2 for pixel with full-bright on																											
n/a	:DISPlay:POStion? <binary order string>	<binary order string> ::= Wave height ch 0, bottom pixed location ch 0 (main display) Wave height ch 1, bottom pixed location ch 1 (main display) ... Wave height ch 15, bottom pixed location ch 15 (main display) Wave height ch 0, bottom pixed location ch 0 (delayed display) Wave height ch. 1, bottom pixed location ch. 1 (delayed display) ... Wave height ch.15, bottom pixed location ch. 15 (delayed display)																											
:DISPlay:ROW <row number>	:DISPlay:ROW?	<row number> ::= 1 to 20; an integer in NR1 format																											
:DISPlay:SOURce <value>	:DISPlay:SOURce?	<value> ::= {PMEMory1 PMEMory2}																											
:DISPlay:TEXT BLANK	n/a	n/a																											
*DMC <macro label>,<macro definition>	n/a	<macro label> ::= quoted ASCII string <macro definition> ::= block data in IEEE 488.2 # format																											
*EMC {{OFF 0} {ON 1}}	*EMC?	{0 1}																											
:ERASe <value>	n/a	<value> ::= {PMEMory1 PMEMory2}																											
*ESE <mask_argument>	*ESE?	<mask_argument> ::= 0 to 255; an integer in NR1 format <table style="margin-left: 20px;"> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Enables</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>128</td> <td>PON - Power On</td> </tr> <tr> <td>6</td> <td>64</td> <td>URQ - User Request</td> </tr> <tr> <td>5</td> <td>32</td> <td>CME - Command Error</td> </tr> <tr> <td>4</td> <td>16</td> <td>EXE - Execution Error</td> </tr> <tr> <td>3</td> <td>8</td> <td>DDE - Device Dependent Error</td> </tr> <tr> <td>2</td> <td>4</td> <td>QYE - Query Error</td> </tr> <tr> <td>1</td> <td>2</td> <td>TRG - Trigger Query</td> </tr> <tr> <td>0</td> <td>1</td> <td>OPC - Operation Complete</td> </tr> </tbody> </table>	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
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																											
n/a	*ESR?	<status> ::= 0 to 255; an integer in NR1 format																											
:FUNctioN2:CENTer <frequency>	:FUNctioN2:CENTer?	<frequency> ::= the current center frequency in NR3 format. The range of legal values is from 0 Hz to 10.00 GHz.																											

Command	Query	Options and Query Returns
:FUNction2:MOVE {LEFT}	n/a	n/a
:FUNction<N>:OFFSet <offset>	:FUNction<N>:OFFSet?	<offset> ::= the value at center screen in NR3 format. The range of legal values is +-10 times the current sensitivity of the selected function. <N> ::= 1 or 2
:FUNction<N>:OPERation <operation>	:FUNction<N>:OPERation?	<operation> ::= {ADD SUBtract MULTiply} for :FUNction1:OPERation {INTegrate DIFFerentiate FFT} for :FUNction2:OPERation <N> ::= 1 or 2
n/a	:FUNction2:PEAKs? {FREQ1 DB1 FREQ2 DB2}	<measurement> ::= {FREQ1 DB1 FREQ2 DB2}. The measurement is the value of the peak specified in NR3 format.
:FUNction<N>:RANGE <range>	:FUNction<N>:RANGE?	<range> ::= the full-scale vertical axis value in NR3 format. The range for FUNction1 is 8E-6 to 8E+6. The range for the INTegrate function is 8E-9 to 400E+3. The range for the DIFFerentiate function is 8E-6 to 1.6E11. The range for the FFT function is 8 to 400 dBV. <N> ::= 1 or 2
:FUNction2:REfERENCE <level>	:FUNction2:REfERENCE?	<level> ::= the current reference level in NR3 format. The range of legal values is from ±160.0 dBV to +240.0 dBV in increments of 2.5 dBV.
:FUNction2:SOURce {CHANnel1 CHANnel2 FUNction1}	:FUNction2:SOURce?	{ANALog1 ANALog2 FUNction1}. The current reference level value is in NR3 format. The range of legal values is from ±160.0 dBV to +240.0 dBV in increments of 2.5 dBV.
:FUNction2:SPAN 	:FUNction2:SPAN?	 ::= the current frequency span in NR3 format. Legal values are 1.221 Hz to 9.766 GHz
:FUNction<N>:VIEW {ON OFF}	:FUNction<N>:VIEW?	{ON OFF} <N> ::= 1 or 2
:FUNction2:WINDow {RECTangular HANNing FLATtop EXPonent}	:FUNction2:WINDow?	{RECTangular HANNing FLATtop EXPonent}
n/a	*GMC? <macro label>	<macro label> ::= quoted ASCII string <macro definition> ::= block data in IEEE 488.2 # format
:HARDcopy:ADDRess <address>	:HARDcopy:ADDRess?	<address> ::= 0 to 31; an integer in NR1 format
:HARDcopy:DEvIce <device>	:HARDcopy:DEvIce?	<device> ::= {THINKjet LASerjet EPSON BWDeskjet PLOTter}
:HARDcopy:FACTors {{OFF 0} {ON 1}}	:HARDcopy:FACTors?	{OFF ON}
:HARDcopy:FFeEd {{OFF 0} {ON 1}}	:HARDcopy:FFeEd?	{OFF ON}
:HARDcopy:GRAYscale {{OFF 0} {ON 1}}	:HARDcopy:GRAYscale?	{OFF ON}

Programmer's Quick Reference Commands and Queries

Command	Query	Options and Query Returns
n/a	*IDN?	HEWLETT-PACKARD,<model>, 0, X.X <model> ::= the model number of the instrument <X.X> ::= the software revision of the instrument
n/a	*LMC?	<ascii string> ::= string list separated by commas
n/a	*LRN?	<learn_string> ::= a block of data in IEEE 488.2 # format
:MASK:CREATe	n/a	n/a
:MASK:DATA <mask_data>	:MASK:DATA?	<mask_data> ::= 1008 bytes of data that represent the currently selected mask template.
:MASK:DESTination {TRACe PRINter}	:MASK:DESTination?	{TRACe PRINter}
:MASK:FAILmode {IN OUT}	:MASK:FAILmode?	{IN OUT}
:MASK:INCRement {ON OFF}	:MASK:INCRement?	{ON OFF}
:MASK:NUMBer <number>	:MASK:NUMBer?	<number> ::= memory (1 or 2)
:MASK:POSTfailure {RUN STOP}	:MASK:POSTfailure?	{RUN STOP}
:MASK:SAVE {ON OFF}	:MASK:SAVE?	{ON OFF}
n/a	:MASK:STATistics?	<compares, failures, failure %> ::= current number of mask tests performed, number of failures detected, percentage of failures
:MASK:TEST {ON OFF}	:MASK:TEST?	{ON OFF}
:MASK:TOLerance <value>	:MASK:TOLerance?	<value> ::= the tolerance used when creating a mask template. The entered value can be from 0.00 to 20.0 percent.
:MEASure:DEFine <measurement>, <edge spec1, edge spec2>	:MEASure:DEFine?	<measurement> ::= DELay edge_spec1 ::= [<slope>] <occurrence> edge_spec2 ::= [<slope>] <occurrence> <slope> ::= {+ -} <occurrence> ::= integer from 1 to 5
:MEASure:DELay [<source>]	:MEASure:DELay? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= floating-point number delay time in seconds in NR3 format
:MEASure:DUTYcycle [<source>]	:MEASure: DUTYcycle? [<source>]	<source> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} <return_value> ::= ratio of positive pulse width to period in NR3 format
:MEASure:FALLtime [<source>]	:MEASure:FALLtime? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= time in seconds between the 10% and 90% voltage levels in NR3 format
:MEASure:FREQuency [<source>]	:MEASure:FREQuency? [<source>]	<source> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} <return_value> ::= frequency in Hertz in NR3 format

Command	Query	Options and Query Returns
:MEASure:NWIDth [<source>]	:MEASure:NWIDth? [<source>]	<source> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} <return_value> ::= negative pulse width in seconds-NR3 format
:MEASure:OVERshoot [<source>]	:MEASure:OVERshoot? [<source>]	<source> ::= {ANALog1 ANALog2} <voltage> ::= the percent of the overshoot of the selected waveform in NR3 format
:MEASure:PERiod [<source>]	:MEASure:PERiod? [<source>]	<source> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} <return_value> ::= waveform period in seconds in NR3 format
:MEASure:PHASe [<source>]	:MEASure:PHASe? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= the phase angle value in degrees in NR3 format
:MEASure:PREShoot [<source>]	:MEASure:PREShoot? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= the percent of preshoot of the selected waveform in NR3 format
:MEASure:PWIDth [<source>]	:MEASure:PWIDth? [<source>]	<source> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} <return_value> ::= width of positive pulse in seconds in NR3 format
:MEASure:RISEtime [<source>]	:MEASure:RISEtime? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= rise time in seconds in NR3 format
:MEASure:SCRatch	n/a	n/a
:MEASure:SHOW {ON OFF}	:MEASure:SHOW?	{ON OFF}
:MEASure:SOURce [<source>]	:MEASure:SOURce?	<source> ::= {ANALog1 ANALog2} for HP 54645A <source> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} for HP 54645D
n/a	:MEASure:TDELta?	<return_value> ::= time difference in seconds between start and stop markers in NR3 format
:MEASure:TSTArt <value> [suffix]	:MEASure:TSTArt?	<value> ::= time at the start marker in seconds in NR3 format [suffix] ::= {s, ms, μ s, ns, ps}
:MEASure:TSTOp <value> [suffix]	:MEASure:TSTOp?	<value> ::= time at the stop marker in seconds in NR3 format [suffix] ::= {s, ms, μ s, ns, ps}
n/a	:MEASure:TVOLT <tvolt_argument>, <slope><occurrence>	<tvolt_argument> ::= positive or negative voltage level that the waveform must cross. <slope> ::= direction of the waveform when <tvolt_argument> is crossed. <occurrence> ::= number of crossings to be reported. <return_value> ::= time in seconds of specified voltage crossing in NR3 format
:MEASure:VAMPLitude [<source>]	:MEASure:VAMPLitude? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= the amplitude of the selected waveform in volts in NR3 format
:MEASure:VAverage [<source>]	:MEASure:VAverage? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= calculated average voltage in NR3 format
:MEASure:VBASe [<source>]	:MEASure:VBASe? [<source>]	<source> ::= {ANALog1 ANALog2} <base_voltage> ::= voltage at the base of the selected waveform in NR3 format

Programmer's Quick Reference Commands and Queries

Command	Query	Options and Query Returns									
n/a	:MEASure:VDELta?	<return_value> ::= delta V value in volts in NR3 format									
:MEASure:VMAX [<source>]	:MEASure:VMAX? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= maximum voltage of the selected waveform in NR3 format									
:MEASure:VMIN [<source>]	:MEASure:VMIN? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= minimum voltage of the selected waveform in NR3 format									
:MEASure:VPP [<source>]	:MEASure:VPP? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= voltage peak-to-peak in NR3 format									
:MEASure:VRMS [<source>]	:MEASure:VRMS? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= calculated dc RMS voltage in NR3 format									
:MEASure:VSTArt <vstart_argument>	:MEASure:VSTArt?	<vstart_argument> ::= voltage value for VMarker 1 in NR3 format <return_value> ::= voltage at VMarker 1 in NR3 format									
:MEASure:VSTOp <vstop_argument>	:MEASure:VSTOp?	<vstop_argument> ::= voltage value for VMarker 2 in NR3 format <return_value> ::= voltage at VMarker 2 in NR3 format									
n/a	:MEASure:VTIME? <vtime_argument>	<vtime_argument> ::= displayed time from trigger in seconds in NR3 format <return_value> ::= voltage at the specified time in NR3 format									
:MEASure:VTOp [<source>]	:MEASure:VTOp? [<source>]	<source> ::= {ANALog1 ANALog2} <return_value> ::= voltage at the top of the waveform in NR3 format									
:MERGe <pixel memory>	n/a	<pixel memory> ::= {PMEMemory1 PMEMemory2}									
*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?	n identifies the module and option pairing. X.X identifies the module software revision. <table style="margin-left: 40px; border: none;"> <tr> <td>Module:</td> <td>No Opt. 005</td> <td>Opt. 005</td> </tr> <tr> <td>Basic Interface</td> <td>0,X.X</td> <td>50,X.X</td> </tr> <tr> <td>Measurement/Storage</td> <td>2,X.X</td> <td>52,X.X</td> </tr> </table>	Module:	No Opt. 005	Opt. 005	Basic Interface	0,X.X	50,X.X	Measurement/Storage	2,X.X	52,X.X
Module:	No Opt. 005	Opt. 005									
Basic Interface	0,X.X	50,X.X									
Measurement/Storage	2,X.X	52,X.X									
*PMC	n/a	n/a									
:PRINt [enhancement]	:PRINt? [enhancement]	[enhancement] ::= {HIREs{,LOREs{,TIFF{,PARAllel {,PCLcolor[,FACTors[,SOFTkey]]}}}}}									
*RCL <value>	n/a	<value> ::= {1 2 3 4 5 6 7 8 9 10}									
*RST	n/a	See reset values in the online Programmer's Reference.									
:RUN	n/a	n/a									
*SAV <value>	n/a	<value> ::= {1 2 3 4 5 6 7 8 9 10}									
:SINGle	n/a	n/a									

Programmer's Quick Reference
Commands and Queries

Command	Query	Options and Query Returns																																				
*SRE <mask>	*SRE?	<p><mask> ::= sum of all bits that are set, 0 to 255; an integer in NR1 format. <mask> ::= following values:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Enables</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>128</td> <td>OPER - Operation Status Register</td> </tr> <tr> <td>6</td> <td>64</td> <td>Not Used</td> </tr> <tr> <td>5</td> <td>32</td> <td>ESB - Event Status Bit</td> </tr> <tr> <td>4</td> <td>16</td> <td>MAV - Message Available</td> </tr> <tr> <td>3</td> <td>8</td> <td>Not used</td> </tr> <tr> <td>2</td> <td>4</td> <td>MSG - Message</td> </tr> <tr> <td>1</td> <td>2</td> <td>USR - User</td> </tr> <tr> <td>0</td> <td>1</td> <td>TRG - Trigger</td> </tr> </tbody> </table>	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									
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>	{ON OFF} <display> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15 PMEMory1 PMEMory2}																																				
n/a	*STB?	<p><value> ::= 0 to 255; an integer in NR1 format, as shown in the following:</p> <table border="1"> <thead> <tr> <th>Bit</th> <th>Weight</th> <th>Name</th> <th>Condition</th> </tr> </thead> <tbody> <tr> <td>7</td> <td>128</td> <td>OPER</td> <td>0 = no enabled operations status conditions occurred 1 = an enabled operation status condition occurred</td> </tr> <tr> <td>6</td> <td>64</td> <td>RQS/MS</td> <td>0 = instrument has no reason for service 1 = instrument is requesting service</td> </tr> <tr> <td>5</td> <td>32</td> <td>ESB</td> <td>0 = no event status conditions occurred 1 = enabled event status condition occurred</td> </tr> <tr> <td>4</td> <td>16</td> <td>MAV</td> <td>0 = no output messages are ready 1 = an output message is ready</td> </tr> <tr> <td>3</td> <td>8</td> <td>----</td> <td>0 = not used</td> </tr> <tr> <td>2</td> <td>4</td> <td>MSG</td> <td>0 = no message has been displayed 1 = message has been displayed</td> </tr> <tr> <td>1</td> <td>2</td> <td>USR</td> <td>0 = no enabled user event conditions have occurred 1 = an enabled user event condition has occurred</td> </tr> <tr> <td>0</td> <td>1</td> <td>TRG</td> <td>0 = no trigger has occurred 1 = a trigger occurred</td> </tr> </tbody> </table>	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
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:DSP <string>	n/a	<string> ::= quoted ASCII string																																				
n/a	:SYSTem:ERRor?	<p><error> ::= an integer error code <error string> ::= quoted ASCII string See error values in the online Programmer's Reference.</p>																																				
:SYSTem:KEY <key_code>	:SYSTem:KEY?	<p><key_code> ::= -1 to 50; an integer See key code values in the online Programmer's Reference.</p>																																				

Programmer's Quick Reference Commands and Queries

Command	Query	Options and Query Returns
:SYSTem:LOCK <value>	:SYSTem:LOCK?	<value> ::= {ON OFF}
:SYSTem:SETup <setup_data>	:SYSTem:SETup?	<setup_data> ::= a maximum of 218 bytes of data in IEEE 488.2 # format.
n/a	:TER?	<return_value> ::= 0 or 1
n/a	:TEST:ALL?	<value> ::= 16-bit integer in NR1 format
:TIMebase:DElay <delay_value>	:TIMebase:DElay?	<delay_value> ::= time from trigger to display reference in seconds. The display reference is left or center in NR3 format.
:TIMebase:MODE <value>	:TIMebase:MODE?	<value> ::= {MAIN DELayed XY ROLL}
:TIMebase:RANGe <range_value>	:TIMebase:RANGe?	<range_value> ::= the following values in NR3 format: 20 ns through 50 s for HP 54645A 50 ns through 500 s for HP 54645D
:TIMebase:REference {LEFT CENTer RIGHt}	:TIMebase:REference?	<return_value> ::= {LEFT CENTer RIGHt}
:TRACe:CLEAR <N>	n/a	<N> ::= the trace memory number (1 to 100)
:TRACe:DATA <N>,<trace_data>	:TRACe:DATA? <N>	<N> ::= the trace memory number (1 to 100) <trace_data> ::= a maximum of 16,748 bytes of data, setup, and label information that represents the current trace.
:TRACe:MODE <N> {ON OFF}	:TRACe:MODE? <N>	<N> ::= 1 to 100 <return_state> ::= {ON OFF}
:TRACe:SAVE <N>	n/a	<N> ::= the trace memory number (1 to 100)
*TRG	n/a	n/a
:TRIGGER:ADVanced:EDGE1 <rising value>,<falling value>	:TRIGger:ADVanced:EDGE1?	<rising value> ::= integer in NR1 format or <string> <falling value> ::= integer in NR1 format or <string> <string> ::= "0xnxxxxn" n ::= {0 1 2 4 8}
:TRIGGER:ADVanced:EDGE2 <rising value> , <falling value>	:TRIGger:ADVanced:EDGE2?	<rising value> ::= integer in NR1 format or <string> <falling value> ::= integer in NR1 format or <string> <string> ::= "0xnxxxxn" n ::= {0 1 ... 9 A B C D E F}
:TRIGGER:ADVanced:GREaterthan <lessthan time> [suffix]	:TRIGger:ADVanced:GREaterthan?	<greater than time> ::= floating-point number from 10 ns to 100 seconds in NR3 format [suffix] ::= {s ms (-3) μ s (-6) ns (-9) ps (-12)}
:TRIGGER:ADVanced:LESSthan <lessthan time> [suffix]	:TRIGger:ADVanced:LESSthan?	<less than time> ::= floating-point number from 15 ns to 100 seconds in NR3 format [suffix] ::= {s ms (-3) μ s (-6) ns (-9) ps (-12)}
:TRIGGER:ADVanced:OPERator <operator>	:TRIGger:ADVanced:OPERator?	<operator> ::= {AND OR THEN ENTered EXITed RANGe GREaterthan LESSthan}

Command	Query	Options and Query Returns
:TRIGGER:ADVanced:PATtern1 <value> , <mask>	:TRIGger:ADVanced:PATtern1?	<value> ::= integer in NR1 format or <string> <mask> ::= integer in NR1 format or <string> <string> ::= ::= "0xnxxxx" n ::= {0 1 ... 9 A B C D E F}
:TRIGGER:ADVanced:PATtern2 <value> , <mask>	:TRIGger:ADVancedPATtern2?	<value> ::= integer in NR1 format or <string> <mask> ::= integer in NR1 format or <string> <string> ::= ::= "0xnxxxx" n ::= {0 1 ... 9 A B C D E F}
:TRIGGER:ADVanced:RANGE <greater than time> <less than time> [suffix]	:TRIGger:ADVanced: RANGE?	<greater than time> ::= floating-point number from 10 ns to 100 seconds in NR3 format <less than time> ::= floating-point number from 15 ns to 100 seconds in NR3 format [suffix] ::= {s ms (-3) μ s (-6) ns (-9) ps (-12)}
:TRIGGER:ADVanced:SOURce1 <source>	:TRIGger:ADVanced:SOURce1?	<source> ::= {PATtern1 NPATtern1 PATtern2 NPATtern2 EDGE1 EDGE2 PAT1ANDEDGE1 PAT2ANDEDGE2}
:TRIGGER:ADVanced:SOURce2 <source>	:TRIGger:ADVanced:SOURce2?	<source> ::= {PATtern1 NPATtern1 PATtern2 NPATtern2 EDGE1 EDGE2 PAT1ANDEDGE1 PAT2ANDEDGE2}
:TRIGger:COUpling {AC DC}	:TRIGger:COUpling?	{AC DC}
:TRIGger:[EDGE]:LEVel <level_argument>	:TRIGger:[EDGE]:LEVel?	For internal triggers, <level_argument> ::= \pm .75 x full-scale voltage from center screen in NR3 format. For external triggers (HP 54645A only), <level_argument> ::= \pm 2 volts with probe attenuation at 1:1 in NR3 format. For digital channels (HP 54645D only), <level_argument> ::= \pm 6 V.
:TRIGger:[EDGE]:SLOPe {NEGative POSitive}	:TRIGger:[EDGE]:SLOPe?	{NEG POS}
:TRIGger:[EDGE]:SOURce <source>	:TRIGger:[EDGE]:SOURce?	<source> ::= {ANALog1 ANALog2 EXTernal LINE} for HP 54645A {ANALog1 ANALog2 DIGital0,...,DIGital15 LINE} for HP 54645D
:TRIGGER:GLITch: GREaterthan <lessthan time> [suffix]	:TRIGger:GLITch: GREaterthan?	<greater than time> ::= floating-point number from 8 ns to 100 seconds in NR3 format [suffix] ::= {s ms (-3) μ s (-6) ns (-9) ps (-12)}
:TRIGGER:GLITch: LESSthan <lessthan time> [suffix]	:TRIGger:GLITch: LESSthan?	<less than time> ::= floating-point number from 8 ns to 100 seconds in NR3 format [suffix] ::= {s ms (-3) μ s (-6) ns (-9) ps (-12)}
:TRIGger:GLITch:LEVel <level_argument>	:TRIGger:GLITch:LEVel?	For internal triggers, <level_argument> ::= \pm .75 x full-scale voltage from center screen in NR3 format. For external triggers (HP 54645A only), <level_argument> ::= \pm 2 volts with probe attenuation at 1:1 in NR3 format. For digital channels (HP 54645D only), <level_argument> ::= \pm 6 V.
:TRIGger:GLITch:POLarity {POSitive NEGative}	:TRIGger:GLITch:POLarity?	{POS NEG}

Programmer's Quick Reference Commands and Queries

Command	Query	Options and Query Returns
:TRIGger:GLITch:QUALifier {GREaterthan LESSthan RANGe}	:TRIGger:GLITch:QUALifier?	{GREaterthan LESSthan RANGe}
:TRIGger:GLITch:RANGe <greater than time>, <less than time> [suffix]	:TRIGger:GLITch:RANGe?	<greater than time> ::= number from 8 ns to 100 seconds in NR3 format <less than time> ::= number from 8 ns to 100 seconds in NR3 format [suffix] ::= {s ms (-3) μ s (-6) ns (-9) ps (-12)}
:TRIGger:GLITch:SOURce <source>	:TRIGger:GLITch:SOURce?	<source> ::= {ANALog1 ANALog2 EXTernal} for HP 54645A {ANALog1 ANALog2 DIGital0,...,DIGital15} for HP 54645D
:TRIGger:HOLDoff <holdoff_time>	:TRIGger:HOLDoff?	<holdoff_time> ::= 200 ns to 20 s in NR3 format
:TRIGger:MODE <mode>[, <type>] NORMal SINGle TV	:TRIGger:MODE?	<mode> ::= {AUTLevel AUTO NORMal} <type> ::= {EDGE TV GLITch} for HP 54645A ::= {EDGE TV GLITch ADVanced PATtern} for HP 54645D
:TRIGger:NREJect {OFF ON}	:TRIGger:NREJect?	{OFF ON}
:TRIGger:PATtern <value>, <mask>, [<source>, <edge>]	:TRIGger:PATtern?	<value> ::= integer or <string> <mask> ::= integer or <string> <string> ::= "0xnxxxxn" n ::= {0 1 2 ... 8 9 A B C D E F} <source> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15} <edge> ::= {RISing FALLing}
:TRIGger:REJect {OFF LF HF}	:TRIGger:REJect?	{OFF LF HF}
:TRIGger:THReshold <channel group>, <threshold type>, [<value>]	:TRIGger:THReshold?	<channel group> ::= {POD1 POD2} <threshold type> ::= {CMOS ECL TTL USERdef} <value> ::= voltage for USERdef (floating-point number) [Volt type] ::= {V mV (-3) μ V (-6)}
:TRIGger:TV:FIELD {ALTernate ONE TWO VERTical}	:TRIGger:TV:FIELD?	{ALTernate ONE TWO VERTical}
:TRIGger:TV:LINE <line_number>	:TRIGger:TV:LINE?	<line_number> ::= integer in NR1 format.
:TRIGger:TV:POLarity {POSitive NEGative}	:TRIGger:TV:POLarity?	{POSitive NEGative}
:TRIGger:TV:SOURce <source>	:TRIGger:TV:SOURce?	<source> ::= {ANALog1 ANALog2}
:TRIGger:TV:STANdard {GENeric NTSC PAL SECam}	:TRIGger:TV:STANdard?	{GENeric NTSC PAL SECam} NOTE: To trigger on a PAL-M signal, use NTSC. The line and field rates are identical.
:TRIGger:TV:TVHFrej {OFF ON}	:TRIGger:TV:TVHFrej?	{OFF ON}

Command	Query	Options and Query Returns
:TRIGger:TV:TVMode <mode>	:TRIGger:TV:TVMode?	<mode> ::= {LINE FIELD1 FIELD2 VERTical} {LINE FIELD1 FIELD2 VERTical ALLLINES ALLFLDS} with the Option 005 Enhanced TV/Video Trigger
:TRIGger:TV:VIR {ON OFF}	:TRIGger:TV:VIR?	{ON OFF}
*TRG n/a	n/a	n/a
n/a	*TST?	<result> ::= 0 or non-zero value; an integer in NR1 format 0 indicates the test passed. Non-zero indicates the test failed.
VAUToscale	n/a	n/a
:VIEW <source text>	n/a	<source text> ::= {ANALog1 ANALog2 DIGital0,...,DIGital15 PMEemory1 PMEemory2}
*WAI	n/a	n/a
:WAVeform:BYTeorder <value>	:WAVeform:BYTeorder?	<value> ::= {LSBFirst MSBFirst}
:WAVeform:COUNT <count_argument>	:WAVeform:COUNT?	<count_argument> ::= {1 4 8 16 32 64 128 256}
n/a	:WAVeform:DATA?	<binary block length bytes>, <binary data> For example, to transmit 4000 bytes of data, the syntax would be: #800004000<4000 bytes of data><NL> 8 is the number of digits that follow 00004000 is the number of bytes to be transmitted <4000 bytes of data> is the actual data
:WAVeform:FORMat <value>	:WAVeform:FORMat?	<value> ::= {WORD BYTE}
:WAVeform:POINts <mode> [,<#points>]	:WAVeform:POINts?	<mode> ::= {NORMal ALL} If <mode> ::= NORMal, <#points> ::= {100 200 250 400 500 800 1000 2000 4000} If <mode> ::= ALL, <#points> must match value returned by the :ACQ:POINts? query
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 <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

Programmer's Quick Reference Commands and Queries

Command	Query	Options and Query Returns
:WAVeform:SOURce <source> or :WAVeform:SOURce:FUNCTion <channel>	:WAVeform:SOURce?	<source> ::= {ANALog1 ANALog2 POD1 POD2} <channel> ::= {1 2}
n/a	:WAVeform:TYPE?	<return_mode> ::= {NORMal PEAK AVERAge}
n/a	:WAVeform:XINCrement?	<return_value> ::= x-increment in the current preamble in NR3 format
n/a	:WAVeform:XORigin?	<return_value> ::= x-origin value in the current preamble in NR3 format
n/a	:WAVeform:XREFerence?	<return_value> ::= x-reference value in the current preamble in NR1 format
n/a	:WAVeform:YINCrement?	<return_value> ::= y-increment value in the current preamble in NR3 format
n/a	:WAVeform:YORigin?	<return_value> ::= y-origin in the current preamble in NR3 format
n/a	:WAVeform:YREFerence?	<return_value> ::= y-reference value in the current preamble in NR1 format

Index

A

Address 707, 2-9
Addressing, 3-4
 the instrument, 3-5
alpha argument, 5-10
Arguments, 1-5
Arm Event Register (ARM), 6-17
availability of measured data, 6-2

B

BASIC, 1-3
Baud rate, 4-8
Bit Definitions in Status Reporting, 6-4
Block data, 1-5, 2-11
Block Diagram
 Status Reporting Overview, 6-3

C

Cable
 RS-232-C, 4-3
carriage return, 5-8
Character data, 1-10
Character program data, 1-10
Clear To Send (CTS), 4-6
clearing
 error queue, 6-18
 registers and queues, 6-19 to 6-20
 Standard ESE Register, 6-12
 TRG bit, 6-11, 6-17
Combining commands, 1-7
Command, 1-5
 Common Commands, 5-6
 error (CME), 6-4
 Lockout, 4-10
 Root Level Commands, 5-6
 structure, 2-5
 Subsystem Commands, 5-8
 tree, 5-6 to 5-9
 types, 5-6
command tree, 5-6
Common command header, 1-6
common commands, 5-6
Communication, 1-3
compound header, 1-6, 5-8
Controller mode
 RS-232-C, 4-7
Controllers, 1-3
conventions, 5-2

D

Data bits, 4-8 to 4-9
 8-Bit mode, 4-9
Data Carrier Detect (DCD), 4-6
Data Communications Equipment, 4-3
Data Set Ready (DSR), 4-6
Data Structures and Status Reporting,
6-5 to 6-8
Data Terminal Equipment, 4-3
Data Terminal Ready (DTR), 4-5, 4-8
DCE, 4-3
Decision Chart-Status Reporting, 6-20
Definite-length response data, 2-11
Device address, 1-4
 HP-IB, 3-5
Device Dependent Error (DDE), 6-4
DIGitize Command, 2-6
documentation conventions, 5-2
DTE, 4-3
DTR (Data Terminal Ready), 4-8
Duplicate mnemonics, 1-7

E

Embedded strings, 1-3, 1-5, 1-11
Enter statement, 1-3
EOI, 1-12
Error Queue & Status Reporting, 6-18
ESB (Event Status Bit), 6-4
ESR (Standard Event Status Register),
6-12
Event
 monitoring, 6-2
 Status Bit (ESB), 6-4
 Status Enable (*ESE), 6-13
Example Program, 2-5
example programs, 7-3
Execution Error (EXE), Status Bit, 6-4
Exponents, 1-10

F

Fractional values, 1-10

H

Headers, 1-5 to 1-6, 5-10
help file, 7-2
Histogram Event Register, 6-17
Host language, 1-5
HP-IB, 3-5

I

IEEE 488.1, 4-2
IEEE 488.2, 4-2, 5-6
 Standard, 1-2
 Status Data Structure Model, 6-2
Infinity Representation, 5-11
Initialization, 2-3
 event status, 6-2
installing help file, 7-2
Instruction
 headers, 1-5
Instruction syntax, 1-4
Instructions, 1-4
Instrument address
 HP-IB, 3-5
Interface
 capabilities of RS-232-C, 4-8
 select code for HP-IB, 3-5
Interface Capabilities, 3-3

K

Key Queue, 6-19

L

LCL (Local Event Register), 6-14
leading colon, 5-8 to 5-9
Limit Test Event Register, 6-15
linefeed, 5-6
 CRLF, 5-8
Local Event Register (LCL), 6-14
lockout, 3-6, 4-10
Long form, 1-9
Lowercase, 1-9
LTER (Limit Test Register), 6-15

M

Mask Test Event Register, 6-16
Master Summary Status (MSS), 6-4
MAV (Message Available), 6-4
menu, Utility/Print, 3-4, 4-7
Message (MSG), Status Bit, 6-4
Message Available (MAV), 6-4
Message Queue, 6-19
mnemonic, 5-10
monitoring events, 6-2
Multiple
 numeric variables, 2-12

-
- program commands, 1-12
 - program data, 1-10
 - queries, 2-12
 - subsystems, 1-12
- N**
- NL, 1-12, 5-6
 - Notation Conventions and Definitions, 5-12
 - Numeric
 - data, 1-10
 - program data, 1-10
 - variables, 2-10
- O**
- online help file, 7-2
 - Operation
 - Complete (OPC), 6-4
 - status, 6-2
 - Operation19
 - Status Register (OPR), 6-14
 - OPR (Operation Status Register), 6-14
 - Output
 - command, 1-4
 - OUTPUT statement, 1-3
 - Output11
 - queue, 6-19
 - Overlapped Commands, 5-11
- P**
- Parameters, 1-5
 - Parity, 4-8
 - Parser, 2-3
 - Power On (PON), Status Bit, 6-4
 - Printer mode
 - RS-232-C, 4-7
 - Program
 - data, 1-5, 1-10
 - example, 2-5
 - syntax, 1-4
 - Program message, 5-9
 - syntax, 1-4
 - terminator, 1-12, 5-8
 - unit separator, 5-8
 - programmer's reference, 7-2
 - programming conventions, 5-2
 - Protocol, 4-8
 - DTR (Data Terminal Ready), 4-8
 - XON/XOFF, 4-8
- Q**
- Query, 1-5, 1-8
 - command, 1-8
 - response, 2-8, 5-11
 - Query Error (QYE), Status Bit, 6-4
 - Question mark, 1-8
- R**
- Receive Data (RD), 4-4, 4-6
 - reference for programmers, 7-2
 - Registers
 - Histogram Event, 6-17
 - Limit Test Event, 6-15
 - Mask Test Event, 6-16
 - Standard Event Status Enable, 6-13
 - reliability of measured data, 6-2
 - Request Control (RQC), 6-4
 - Request Service (RQS), 6-4
 - Request To Send (RTS), 4-6
 - Response
 - data, 2-11
 - Generation, 5-11
 - Root Level commands, 5-6
 - RQC (Request Control) Status Bit, 6-4
 - RQS (Request Service) Status Bit, 6-4
 - RS-232-C, 4-2
- S**
- sample programs, 7-3
 - Separator, 1-5
 - Sequential commands, 5-11
 - serial poll
 - of Status Byte Register, 6-10
 - SPOLL example, 6-10
 - serial poll (SPOLL) in example, 6-10
 - Service Request Enable Register, 6-11
 - setting
 - bits in SRE Register, 6-11
 - LCL bit, 6-14
 - Standard ESE Register, 6-13
 - TRG bit, 6-11
 - Short form, 1-9
 - Simple command header, 1-6
 - Spaces, 1-5
 - SRE (Service Request Enable Register), 6-11
 - Standard
 - Event Status Enable Register, 6-13
 - Event Status Register, 6-12
 - Status Data Structure Model, 6-2
 - Status, 2-12
 - of an operation, 6-2
 - registers, 2-12
 - Status Byte Register, 6-9 to 6-10
 - and serial polling, 6-10
 - Status Reporting, 6-2
 - Bit Definitions, 6-4
 - Data Structures, 6-5 to 6-8
 - Decision Chart, 6-20
 - Stop bits, 4-8
 - String variables, 2-9
 - Subsystem commands, 5-6
 - summary bits, 6-9
- T**
- Talking to the instrument, 1-3
 - Terminator, 1-12
 - Three-wire Interface, 4-4
 - Transmit Data (TD), 4-4, 4-6
 - Transmit On/Transmit Off, 4-8
 - TRG bit in the status byte, 6-11
 - Trigger (TRG), 6-4
 - Event Register, 6-11
 - Truncation Rules, 5-10
- U**
- UER (User Event Register), 6-14
 - Uppercase, 1-9
 - User
 - Event Register (UER), 6-14
 - Request (URQ), 6-4
- W**
- White space, 1-5
- X**
- XON/XOFF, 4-8
-

© Copyright Hewlett-Packard Company 1996
All Rights Reserved.

Microsoft is a registered trademark of Microsoft Corporation.

Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

Document Warranty

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

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

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

Safety

This apparatus has been designed and tested in accordance with IEC Publication 348, Safety Requirements for Measuring Apparatus, and has been supplied in a safe condition. This is a Safety Class I instrument (provided with terminal for protective earthing). Before applying power, verify that the correct safety precautions are taken (see the following warnings). In addition, note the external markings on the instrument that are described under "Safety Symbols."

Warning

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

Safety Symbols



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



Hazardous voltage symbol.



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

WARNING

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

CAUTION

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

Product Warranty

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

For warranty service or repair, this product must be returned to a service facility designated by Hewlett-Packard.

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

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

Limitation of Warranty

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

No other warranty is expressed or implied. Hewlett-Packard specifically disclaims the implied warranties of merchantability or fitness for a particular purpose.

Exclusive Remedies

The remedies provided herein are the buyer's sole and exclusive remedies.

Hewlett-Packard shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory.

Assistance

Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.

For any assistance, contact your nearest Hewlett-Packard Sales Office.

Certification

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

About this edition

This is the first edition of the *HP 54645A/D Oscilloscopes Programmer's Guide*.

Publication number
54645-97000

Printed in USA.

Edition dates are as follows:

First edition, June 1996

New editions are complete revisions of the manual. Update packages, which are issued between editions, contain additional and replacement pages to be merged into the manual by you. The dates on the title page change only when a new edition is published.

A software or firmware code may be printed before the date. This code indicates the version level of the software or firmware of this product at the time the manual or update was issued. Many product updates do not require manual changes; and, conversely, manual corrections may be done without accompanying product changes. Therefore, do not expect a one-to-one correspondence between product updates and manual updates.

The following list of pages gives the date of the current edition and of any changed pages to that edition.

All pages original edition