Programmer's Reference

Publication number 54542-97038
First edition, November 1995

This book applies directly to firmware revision 3.XX.

For Safety information, Warranties, and Regulatory information, see the pages behind the index

HP 54520 and 54540 Series Oscilloscopes

In This Book

This programmer's reference contains general information, remote programming information, programming examples, and oscilloscope commands for programming the HP54520A and HP 54540A Series Oscilloscopes.

Instruments covered by the Programmers Reference.

This Programmer's Reference contains programming information for the following models:

- HP 54520A - 2 channel 500MSa/s.
- HP 54522A - 2 channel 2GSa/s.
- HP 54540A - 4 channel 500MSa/s.
- HP 54542A - 4 channel 2GSa/s.

What is in the Programmer's Reference?

The Programmer's Reference is organized as follows:

Chapter 1 through Chapter 3 contains general information about programming basics, interface requirements, and documentation conventions

Chapter 4 contains programming examples.

1 Introduction to Programming an Instrument

2 Programming an Instrument

3 Programming and Documentation Conventions

4 Example Programs

5 Common Commands

6 Root Level Commands

7 SYSTem Subsystem

8 ACQuire Subsystem

9 CALibrate Subsystem

10 CHANnel Subsystem

11 DISK Subsystem

12 DISPlay Subsystem

13 FUNCtion Subsystem

14 HARDcopy Subsystem

Chapters 5 through 24, explain each command in the command set for the oscilloscope. These chapters are organized in subsystems with each subsystem representing a front-panel menu.

The commands explained in this part give you access to common commands, root level commands, and individual subsystem commands. These chapters are designed to provide a concise description of each command.

Chapter 25 explains status reporting and how it can be used to monitor the flow of your programs and measurement process.

Chapter 26 contains error message descriptions.

Chapter 27 is the algorithms of the automatic measurements.

15 MARKer Subsystem
16 MEASure Subsystem
17 Multiple Memory (MMEMory) Subsystem
18 Pixel Memory (PMEMory) Subsystem
19 SEQuential Subsystem
20 TIMebase Subsystem
21 TRIGger Subsystem
22 Waveform Memory (WMEMory) Subsystem
23 WAVeform Subsystem
24 EXTernal Trigger Subsystem
25 Status Reporting
26 Error Messages
27 Algorithms
Index

Contents

1 Introduction to Programming an Instrument

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

2 Programming an Instrument

Initialize 2-3
Order of Commands 2-4
Capture 2-7
Analyze 2-9
Response Header Options 2-10
Response Data Formats 2-11
String Variables 2-12
Numeric Variables 2-13
Definite-Length Block Response Data 2-14
Multiple Queries 2-14
Instrument Status 2-15

3 Programming and Documentation Conventions

Truncation Rules 3-3
The Command Tree 3-4
Infinity Representation 3-9
Sequential and Overlapped Commands 3-9
Response Generation 3-9
Notation Conventions and Definitions 3-10
Syntax Diagrams 3-11
Program Examples 3-11
Command Set Organization 3-12

4 Example Programs

HP Basic Initialize Program 4–4
HP Basic Digitize Program 4–6
Microsoft QuickBASIC Initialize Program 4–10
Microsoft QuickBASIC Digitize Program 4–13
Microsoft QuickBASIC Initialize Program 4–19
Microsoft QuickC Initialize Program 4–23
Microsoft QuickC Digitize Program 4–26
Microsoft QuickC Initialize Program 4–32
Instrument BASIC for Windows Initialize Program 4–36
Instrument BASIC for Windows Digitize Program 4–38

5 Common Commands

Common Commands 5–5
*CLS (Clear Status) 5–6
*DMC (Define Macro) 5–7
*EMC (Enable Macro) 5–8
*ESE (Event Status Enable) 5–9
*ESR? (Event Status Register) 5–11
*GMC? (Get Macro Contents) 5–13
*IDN? (Identification Number) 5–14
*LMC? (Learn Macro) 5–15
*LRN? (Learn) 5–16
*OPC (Operation Complete) 5–17
*OPT? (Option) 5–18
*PMC (Purge Macro) 5–19
*RCL (Recall) 5–20
*RST (Reset) 5–21
*SAV (Save) 5–23
*SRE (Service Request Enable) 5–24
*STB? (Status Byte) 5–26
*TRG? (Trigger) 5–28
*TST? (Test) 5–29
*WAI (Wait) 5–30

6 Root Level Commands

AUToscale 6-6
BEEPer 6-7
BLANk 6-8
BNC 6-9
DIGitize 6-10
ERASe 6-11
LER? (Local Event Register) 6-12
LTER? (Limit Test Event Register) 6-13
MENU 6-14
MERGe 6-15
PCFRequency 6-16
PLOT 6-17
POWerup 6-18
PRINt? 6-19
RUN 6-20
SERial (Serial Number) 6-21
STATus? 6-22
STOP 6-23
STORe 6-24
TER? (Trigger Event Register) 6-25
VIEW 6-26

7 SYSTem Subsystem

DATE 7-5
DSP 7-6
ERRor? 7-7
HEADer 7-10
KEY 7-11
LONGform 7-13
PIMacro 7-14
SETup 7-15
TIME 7-16
UTILity 7-17

8 ACQuire Subsystem

Acquire Type and Count 8-4
COMPlete 8-7
COUNt 8-9
POINts 8-10
TYPE 8-12

9 CALibrate Subsystem

DATA:ASCii? 9-3
SETup? 9-4
TNULl 9-5

10 CHANnel Subsystem

COUPling 10-5
DISPlay 10-6
ECL 10-7
HFReject 10-8
LFReject 10-9
OFFSet 10-10
PROBe 10-11
RANGe 10-13
SETup? 10-14
TTL 10-15

11 DISK Subsystem

CDIRectory 11-5
DELete 11-6
DIRectory? 11-7
FORMat 11-8
LOAD 11-9
MDIRectory 11-10
PWD? 11-11
SIMage 11-12
STORe 11-13
File Formats 11-14

12 DISPlay Subsystem

COLumn 12-6
CONNect 12-7
DATA 12-8
FORMat 12-10
GRATicule 12-11
INVerse 12-12
LINE 12-13
MASK 12-14
PERSistence 12-16
ROW 12-18
SCReen 12-19
SETup? 12-20
SOURce 12-21
STRing 12-22
TEXT 12-23
{MARKer | TMARker | VMARker} 12-24

13 FUNCtion Subsystem

ADD 13-6
DIFF 13-7
DISPlay 13-8
FFT 13-9
FREQuency 13-10
INTegrate 13-11
INVert 13-12
LEVel 13-13
MAGNify 13-14
MODE? 13-15
MULTiply 13-16
OFFSet 13-17
ONLY 13-18
PEAK 13-19
POINts 13-21
RANGe 13-24
SETup? 13-25

SPAN 13-27
SUBTract 13-28
VERSus 13-29
WINDow 13-30

14 HARDcopy Subsystem

LENGth 14-6
MODE 14-7
PAGE 14-8
PLOT:AREA 14-9
PLOT:INITialize 14-10
PLOT:{PEN|COLor} 14-11

15 MARKer Subsystem

DISPlay 15-5
MODE 15-6
SETup? 15-7
X1Position 15-8
X2Position 15-9
X1Y1source 15-10
X2Y2source 15-11
XDELta? 15-12
Y1Position (Command ignored in waveform mode) 15-13
Y2Position (Command ignored in waveform mode) 15-14
YDELta? 15-15

16 MEASure Subsystem

Measurement Setup 16-12
User-Defined Measurements 16-12
Measurement Error 16-12
Making Measurements 16-13
ALL? 16-15
COMPare 16-16
CURSor? 16-18
DEFine 16-19

DELay 16-21
DESTination 16-22
DUTycycle 16-24
ESTArt 16-25
ESTOp 16-27
EANalysis 16-29
FALLtime 16-31
FREQuency 16-32
LIMittest 16-33
LOWer 16-34
MODE 16-36
MWINdow 16-37
NWIDth 16-38
OVERshoot 16-39
PERiod 16-40
POSTfailure 16-41
PREShoot 16-42
PWIDth 16-43
RESults? 16-44
RISetime 16-46
SCRatch 16-47
SOURce 16-48
STATistics 16-49
STATistics:MODE 16-50
TDELta? 16-51
TMAX? 16-52
TMIN? 16-53
TSTArt 16-54
TSTOp 16-55
TVOLt? 16-56
UNITs 16-57
UPPer 16-58
VACRms 16-60
VAMPlitude 16-61
VAVerage 16-62
VBASe 16-63
VDCRms 16-64

VDELta? 16–65
VFIFty 16–66
VMAX 16–67
VMIN 16–68
VPP 16–69
VRELative 16–70
VSTArt 16–72
VSTOp 16–73
VTIMe? 16–74
VTOP 16–75
WCOMpare 16–76
WCOMpare:ALLowance 16–77
WCOMpare:COMPare 16–78
WCOMpare:DESTination 16–79
WCOMpare:HALLowance 16–81
WCOMpare:POSTfailure 16–82
WCOMpare:VALLowance 16–83
WCOMpare:WTESt 16–84

17 Multiple Memory (MMEMory) Subsystem

DISPlay 17–4
FNUMber 17–5
SOURce 17–6
STORe 17–7

18 Pixel Memory (PMEMory) Subsystem

CLEar 18–3
DISPlay 18–4
MERGe 18–5
SETup? 18–6

19 SEQuential Subsystem

DISPlay 19-5
EXCLude 19-6
INCLude 19-8
NPOints 19-10
NSEGments 19-11
SETup? 19-13
SNUMber 19-14
SOURce 19-15
TTAGs? 19-16
TTDifference? 19-17

20 TIMebase Subsystem

DELay 20-4
MODE 20-5
RANGe 20-6
REFerence 20-7
RLENgth 20-8
SAMPle 20-10
SAMPle:CLOCk 20-11
SETup? 20-12

21 TRIGger Subsystem

Trigger Mode 21-8
The EDGE Trigger Mode 21-9
The Pattern Trigger Mode 21-10
The State Trigger Mode 21-11
The Delay Trigger Mode 21-12
The TV Trigger Mode 21-13
The Glitch Trigger Mode 21-14
CENTered 21-15
CONDition 21-16
COUPling 21-18
DELay 21-19
DELay:SLOPe 21-20

DELay:SOURce 21-21
FIELd 21-22
GLITch 21-23
GLITch:CENTered 21-24
GLITch:HOLDoff 21-25
GLITch:LEVel 21-26
GLITch:SOURce 21-27
GLITch:WIDth 21-28
HOLDoff 21-29
LEVel 21-30
LINE 21-31
LOGic 21-32
MODE 21-33
NREJect 21-37
OCCurrence 21-38
OCCurrence:SLOPe 21-39
OCCurrence:SOURce 21-40
PATH 21-41
POLarity 21-42
QUALify 21-43
SETup 21-44
SLOPe 21-48
SOURce 21-49
STANdard 21-50

22 Waveform Memory (WMEMory) Subsystem

DISPlay 22-4
GET 22-5
PROTect 22-6
SETup? 22-7
XOFFset 22-8
XRANge 22-9
YOFFset? 22-10
YRANge? 22-11

23 WAVeform Subsystem

Waveform Data and Preamble 23–4
Data Acquisition Types 23–5
Data Conversion 23–9
Data Format for Remote Interface Transfer 23–10
DATA 23–12
FORMat 23–14
POINts? 23–15
PREamble 23–16
SOURce 23–18
TYPE? 23–19
XINCrement? 23–20
XORigin? 23–21
XREFerence? 23–22
YINCrement? 23–23
YORigin? 23–24
YREFerence? 23–25

24 EXTernal Trigger Subsystem

COUPling 24–4
HFReject 24–5
LFReject 24–6
PROBe 24–7
RANGe 24–8
SETup? 24–9

25 Status Reporting

Serial Poll 25–6

26 Error Messages

27 Algorithms

Measurement Setup 27–3
Making Measurements 27–3
Top-Base 27–4
Edge Definition 27–4
Algorithm Definitions 27–5

1

Introduction to Programming an Instrument

Introduction

This chapter introduces you to the basics of remote programming. The programming instructions explained in this manual conform to the IEEE 488.2 Standard Digital Interface for Programmable Instrumentation. These programming instructions provide a means of remotely controlling the oscilloscopes.

There are four basic operations that can be done with a controller and an oscilloscope via the remote interface. You can:

- Set up the instrument and start measurements.
- Retrieve setup information and measurement results.
- Digitize a waveform and pass the data to the controller.
- Send measurement data to the instrument.

Other more complicated tasks are accomplished with a combination of these four basic functions.

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

Talking to the Instrument

In general, computers acting as controllers communicate with the instrument by sending and receiving messages over a remote interface. Instructions for programming the oscilloscope 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 9000 Series 200/300 BASIC uses the OUTPUT statement for sending commands and queries to the oscilloscope. 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 BASIC statement sends a command which turns the command headers on:

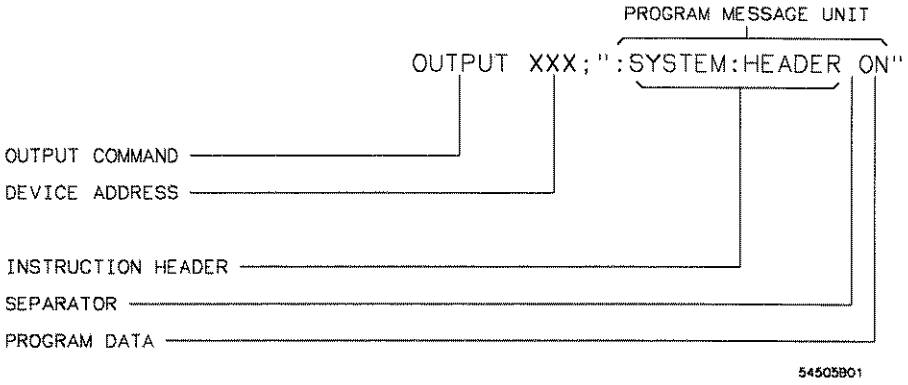
```
OUTPUT <device_address> ;":SYSTEM:HEADER ON"<terminator>
```

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 have an understanding of 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 1-1 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 language you are using. Throughout this manual HP 9000 Series 200/300 BASIC 5.0 is used in the programming examples. If you are using another language, you will need to find the equivalents of BASIC commands like OUTPUT, ENTER, and CLEAR in order to convert the examples. The instructions listed in this manual for the oscilloscopes are always shown between quotes in the example programs.

Device Address

The location where the device address must be specified is also dependent on which host language you are using. In some languages, this may be specified outside the output command. In 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 you use varies according to how you have configured the bus for your application.

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>. 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 figure 3-1 illustrates how all the mnemonics can be joined together to form a complete header (see the "Programming and Documentation Conventions" chapter).

The example in figure 1-1 shows 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. The syntax is:

```
<program_mnemonic><terminator>
```

When program data must be included with the simple command header (for example, :DIGITIZE CHANNEL1), white space is added to separate the data from the header. The syntax is:

```
<program_mnemonic><separator><program_data><terminator>
```

Compound Command Header Compound command headers are a combination of two or more program mnemonics. The first mnemonic selects the subsystem, and the last mnemonic selects the function within that subsystem. Additional mnemonics may appear between the subsystem mnemonic and the function mnemonic when there are additional levels within the subsystem that must be traversed. 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 :SYSTEM:LONGFORM ON)

To traverse down a level of a subsystem to execute a subsystem within that subsystem:

```
:<subsystem>:<subsystem>:<function><separator>
<program_data><terminator>
```

(For example :TRIGGER:DELAY:SOURCE CHANNEL1)

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

```
*<command_header><terminator>
```

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

Combining Commands from the Same Subsystem

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

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

(For example :SYSTEM:LONGFORM ON;HEADER ON)

Duplicate Mnemonics

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

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

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

```
:TIMEBASE:RANGE 1
```

- sets the horizontal time base to 1 second full scale.

CHANnel1 and TIMebase are subsystem selectors and determine which range is being modified.

Query Command

Command headers immediately followed by a question mark (?) are queries. After receiving a query, the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the answer is transmitted across the bus to the designated listener (typically a controller). For example, the query :TIMEBASE:RANGE? places the current time base setting in the output queue. In 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, with the query actually activating the measurement. For example, the command :MEASURE:RISETIME? instructs the instrument to measure the rise time of your waveform and place the result in the output queue.

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

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 an 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. Either of the following examples turn the headers on.

```
:SYSTEM:HEADER ON   - long form
:SYST:HEAD ON       - 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.

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

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, :TRIGGER:DELAY TIME,1.23E-01 has two program data: TIME and 1.23E-01.

There are two main types of program data which are used in oscilloscope 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 auto, trigger, or single. The character program data in this case may be AUTO, TRIGGER, or SINGLE. :TIMEBASE:MODE SINGLE sets the time base mode to single.

The available mnemonics for character program data are always included with the instruction's syntax definition. When sending commands, either the long form or short form (if one exists) may be used. Upper-case and lower-case letters may be mixed freely. When receiving responses, upper-case letters are used exclusively. The use of long form or short form in a response depends on the setting you last specified with the SYSTem:LONGform command.

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

Program Message Terminator

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

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

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><separator><data>;:
<program_mnemonic><separator><data><terminator>
:CHANNEL1:RANGE 0.4;:TIMEBASE:RANGE 1
```

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

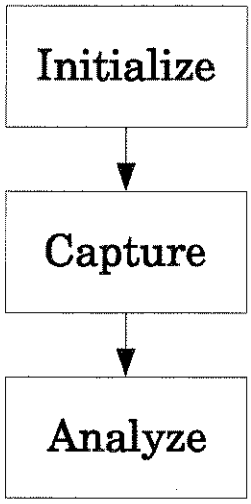
2

Programming an Instrument

Introduction

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

Fundamentally, this process can be broken down into three basic functions as shown below:



54542B78

A detailed description for each of these functions is explained in the following paragraphs.

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

Initialize

Initialization

The first step in programming is to initialize the remote interface and oscilloscope. This ensures consistent, repeatable performance of the program. Without these initialization procedures, a program may run one time, but not function properly the next time. This may be due to some setting change made either from the front panel or remote interface which drastically changed the configuration of the oscilloscope.

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

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

When you are using Remote Interface, 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. For example:

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

> The actual commands and syntax for initializing the instrument are discussed in the "Common Commands" chapter.
>
> Refer to your controller manual and programming language reference manual for information on initializing the interface.

Configuration

The second step of initialization is to configure the oscilloscope. The timebase, channel, and trigger subsystems are set for the desired measurement.

Autoscale The AUTOSCALE feature of Hewlett-Packard digitizing oscilloscopes 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 Autoscale is:

```
:AUTOSCALE<terminator>
```

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

```
:CHANNEL1:RANGE 0.64;OFFSET 0.25<terminator>
:TIMEBASE:RANGE 1E-6;DELAY 20E-9;MODE TRIGGERED<terminator>
:TRIGGER:LEVEL 0.25;SLOPE POSITIVE<terminator>
```

This example sets the vertical to 0.64 volts full-scale (80 mV/div) centered at 0.25 V. The horizontal time is set to 1 ms full-scale with 20 ns delay. The time base mode is set to triggered, and the trigger circuit is set to trigger at 0.25 volts on a positive slope.

Acquisition/Completion Criteria

Finally, specify the ACQuire subsystem, From the front panel, the only setting that can be made is the mode (such as normal, average, or envelope). But over the bus, you can set the number of points to be acquired and the completion criteria -- both impact throughput.

Order of Commands

The order of the commands in a program may be critical to the task you are trying to accomplish. If your commands do not follow the proper order, later commands may cancel out earlier commands and conditions you thought were setup are no longer valid. Table 2-1 lists the order of some commonly used commands in the TIMebase, CHANnel, TRIGger, ACQuire, and WAVeform subsystems. For more information, refer to the specific commands in this manual.

To use the table, select the appropriate commands you need from one subsystem at a time, working from top to bottom. Then move to the next subsystem, working from left to right. It is not necessary to use all of the commands, but the order in which you use the commands is critical.

Table 2-1

Order of Commands

	First	Second	Third	Fourth	Fifth
Subsystem:	**:TIMebase**	**:CHANnel<n>**	**:TRIGger**	**:ACQuire**	**:WAVeform**
First	:MODE AUTO TRIGgered SINGle	:PROBe	:MODE EDGE PATTern STATe DELay GLITch TV	:TYPE NORMal AVERage ENVelope RAWData	:FORMat ASCii WORD BYTE COMPressed
Second	:SAMPling REALtime REPetitive	:RANGe	:SOURce CHANnel<n> EXTernal	:COMPlete	:SOURce CHANnel<n> WAVeform {1|2|3|4}
Third	:REFerence LEFT CENTer RIGHt	:OFFSet	:LEVel	:POINts	
Fourth	:RANGe	:COUPling AC DC LFReject	:SLOPe POSitive NEGative	:COUNt	
Fifth	:DELay				

* <n> = 1 or 2 for the HP 54520A or HP 54522A.
<n> = 1 through 4 for the HP 54540A or HP 54542A

Example Program

This program demonstrates the basic command structure used to program the oscilloscope. It assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe.

```
10 CLEAR 707 ! Initialize instrument interface
20 OUTPUT 707;"*RST" ! Initialize instrument to preset state
30 ! BNC output defaults to the probe mode at 496Hz after *RST
40 OUTPUT 707;":TIMEBASE:MODE TRIGGERED"       ! Triggered time base mode
50 OUTPUT 707;":TIMEBASE:RANGE 5E-4"     ! Time base to 50 us/div
60 OUTPUT 707;":TIMEBASE:DELAY 0" ! Delay to zero
70 OUTPUT 707;":TIMEBASE:REFERENCE CENTER"! Display reference at center
80 OUTPUT 707;":CHANNEL1:PROBE 10"       ! Probe attenuation to 10:1
90 OUTPUT 707;":CHANNEL1:RANGE 1.6"      ! Vertical range to 1.6 V full scale
100 OUTPUT 707;":CHANNEL1:OFFSET -.4"    ! Offset to -0.4
110 OUTPUT 707;":CHANNEL1:COUPLING DC"   ! Coupling to DC
120 OUTPUT 707;":TRIGGER:MODE EDGE"      ! Edge triggering
130 OUTPUT 707;":TRIGGER:LEVEL -.4"      ! Trigger level to -0.4
140 OUTPUT 707;":TRIGGER:SLOPE POSITIVE"       ! Trigger on positive slope
150 OUTPUT 707;":DIGITIZE CHAN1"  ! Capture data
160 OUTPUT 707;":MEASURE:VPP?"    ! Perform Vpeak measurement
170 ENTER 707;Value ! Enter measurement result
180 PRINT "Vpp = ";Value   ! Print measurement result
190 END
```

Program Overview

Line 10 Initializes the instrument interface to a known state.

Line 20 Initializes the instrument to a preset state.

Lines 40 through 70 Set the time base mode to triggered with the horizontal time at 50 ms/div with 0 s of delay referenced at the center of the graticule.

Lines 80 through 110 Set the vertical range to 1.6 volts full scale centered at -0.4 volts with 10:1 probe attenuation and DC coupling.

Lines 120 through 140 Configures the instrument to trigger at -0.4 volts on a positive edge.

Line 150 Captures the measurement data.

Line 160 through 180 Runs the measurement, acquires the data, and prints the results.

Capture

Even though the oscilloscope has been properly initialized and set up, measurements cannot yet be made. It must be noted that when the oscilloscope is responding to remote interface commands, it is in an interrupt mode, and the data acquisition process is being held off. Changes to the oscilloscope's configuration invalidate previously collected data. As a result, if a measurement is requested immediately, there may not be sufficient waveform data available to analyze. Therefore, it is recommended that the next program step is to capture the data using the DIGitize command.

The DIGitize command captures data as specified by you. It is a macro that first clears the waveform data buffers and then starts the acquisition process. Acquisition continues until the criteria, such as number of averages, completion criteria, and number of points, is satisfied. Once the criteria is satisfied, the acquisition process stops. The signal is displayed on the oscilloscope, and the captured data can be measured, stored in memory, or transferred to the computer for analysis. Additional commands are stored in a buffer of the oscilloscope until the DIGitize is completed.

An alternative to the DIGitize command is to let the oscilloscope run with a wait in the program before doing any measurements. This method is not recommended because the length of the wait is an indeterminant value. The DIGitize command gives confidence to the measurement by ensuring complete data capture. In addition, the DIGitize stops the acquisition process, so all measurements are made on the data being displayed, not on a continually changing set of data. In some cases however, such as using statistics, the DIGitize command may not yield meaningful results.

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.

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

When the DIGitize command is sent to an instrument, the specified channel signal is digitized with the current ACQuire parameters. To transfer waveform data across the bus, you must setup the WAVeform:SOURce and :WAVeform:FORMat parameters for the waveform data prior to sending the :WAVeform:DATA? query.

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;":TIMEBASE:SAMPLE REPETITIVE"<terminator>
OUTPUT 707;":ACQUIRE:TYPE AVERAGE"<terminator>
OUTPUT 707;":ACQUIRE:COMPLETE 100"<terminator>
OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"<terminator>
OUTPUT 707;":WAVEFORM:FORMAT ASCII"<terminator>
OUTPUT 707;":ACQUIRE:COUNT 4"<terminator>
OUTPUT 707;":ACQUIRE:POINTS 500"<terminator>
OUTPUT 707;":DIGITIZE CHANNEL1"<terminator>
OUTPUT 707;":WAVEFORM:DATA?"<terminator>
```

This setup places the instrument into the repetitive averaged mode with four averages and defines the data record to be 500 points. This means that when the DIGitize command is received, the waveform is not complete into memory until 500 points have been averaged at least four 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 ASCii WORD, BYTE, or COMPressed.

A digitize operation may be aborted by pressing the "BLUE" key on the front panel, or by sending a Device Clear over the bus. Digitize operations with :ACQuire:TYPE RAWData can only be aborted with the front panel BLUE key.

Analyze

Once the data has been captured, measurements can be made. The measurements can include the IEEE standard parametric measurements (such as VPP, Frequency, PWIDth) or the positioning and reading of voltage and time markers. The waveform data can be transferred from the oscilloscope into the computer for special analysis, if desired.

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 bus 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 :SYSTEM:LONGFORM? you would execute the BASIC statement:

```
ENTER <device_address> ;Setting$
```

where <device_address> represents the address of your device. This would enter the current setting for the LONGFORM command 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 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.

Response Header Options

The format of the returned ASCII string depends on the current settings of the SYSTem HEADer and LONGform commands. The general format is:

```
<header><separator><data><terminator>
```

The header identifies the data that follows and is controlled by issuing a :SYSTem:HEADer ON/OFF command. If the state of the header command is OFF, only the data is returned by the query.

The format of the header is controlled by the :SYSTem:LONGform ON/OFF command. If long form is OFF, the header is in its short form and the header varies in length depending on the particular query. The separator between the header and the data always consists of one space.

The following examples show some possible responses for a :MEASure:FREQuency? frequency measurement query:

```
<data><terminator>                                    (with HEADER OFF )
:MEAS:FREQ<separator><data><terminator>               (with HEADER ON/LONGFORM OFF )
:MEASURE:FREQUENCY<separator><data><terminator>       (with HEADER ON/LONGFORM ON )
```

> A command or query may be sent in either long form or short form, or in any combination of long form and short form. The HEADer and LONGform commands only control the format of the returned data and have no effect on the way commands are sent.
>
> Refer to the "System Subsystem" chapter for information on turning the HEADer and LONGform commands on and off.

Response Data Formats

Both numeric and character data are returned as a series of ASCII characters, as described in the following sections. Mnemonics in the program data are returned in the same format as the header, as specified by the LONGform command. Like the headers, the mnemonics are always in upper-case.

The following examples are possible responses to the :TRIGger:SLOPe? query:

```
:TRIGGER:SLOPE POSITIVE<terminator>      (with HEADER ON/LONGFORM ON)
:TRIG:SLOP POSITIVE<terminator>          (with HEADER ON/LONGFORM OFF)
POSITIVE<terminator>                       (with HEADER OFF/LONGFORM ON)
POS<terminator>                          (with HEADER OFF/LONGFORM OFF)
```

Refer to the individual commands in this manual for information on the format (alpha or numeric) of the data returned from each query.

String Variables

If you want to transfer the headers for queries, you must bring the returned data into a string variable. Reading queries into string variables is simple and straightforward, requiring little attention to formatting. For example:

```
ENTER <device_address>;Result$
```

places the output of the query in the string variable Result$.

> In HP BASIC 5.0, string variables are case sensitive and must be expressed exactly the same each time they are used.

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.

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

The following example shows the data being returned to a string variable with headers on:

Example

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

After running this program, the controller will typically display:

```
CHANNEL1:RANGE +8.00000E-01
```

Numeric Variables

If you do not need to see the headers when a numeric value is returned from the instrument, then you can use a numeric variable. When you are receiving numeric data into a numeric variable, turn the headers off.

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

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

Example

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

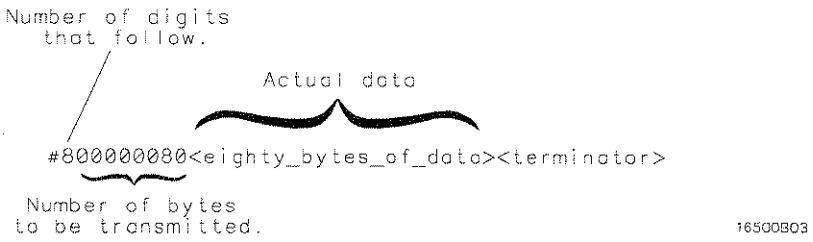
After running this program, the controller will typically display: .8

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 80 bytes of data, the syntax would be:

Figure 2-1



Definite-Length Block Response Data

The "8" states the number of digits that follow, and "00000080" 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 :TIMEebase:RANGe?;DELay? with HEADer and LONGform on would be:

```
:TIMEBASE:RANGE <range_value>;:TIMEBASE:DELAY <delay_value>
```

If you do not need to see the headers when the numeric values are returned, read the result of the query into numeric variables. For example, use the following program message to read the query :TIMebase:RANGe;DELay? into multiple numeric variables:

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

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

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. The "Status Reporting" chapter explains how to check the status of the instrument.

3

Programming and Documentation Conventions

Introduction

This chapter covers conventions which are used in programming the instrument, as well as conventions used in the remainder of this manual. This chapter contains a detailed description of the command tree and command tree traversal.

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 table 3-1.

Table 3-1 Mnemonic Truncation

Long Form	Short Form
RANGE	RANE
PATTERN	PATT
TIME	TIME
DELAY	DEL

The Command Tree

The command tree in figure 3-1 shows all of the commands in the oscilloscope and the relationship of the commands to each other. The IEEE 488.2 common commands are not listed as part of the command tree since they do not affect the position of the parser within the tree. When a program message terminator (<NL>, linefeed - ASCII decimal 10) or a leading colon (:) is sent to the instrument, the parser is set to the "root" of the command tree.

Command Types

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

- Common commands.
- Root level commands.
- Subsystem commands.

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

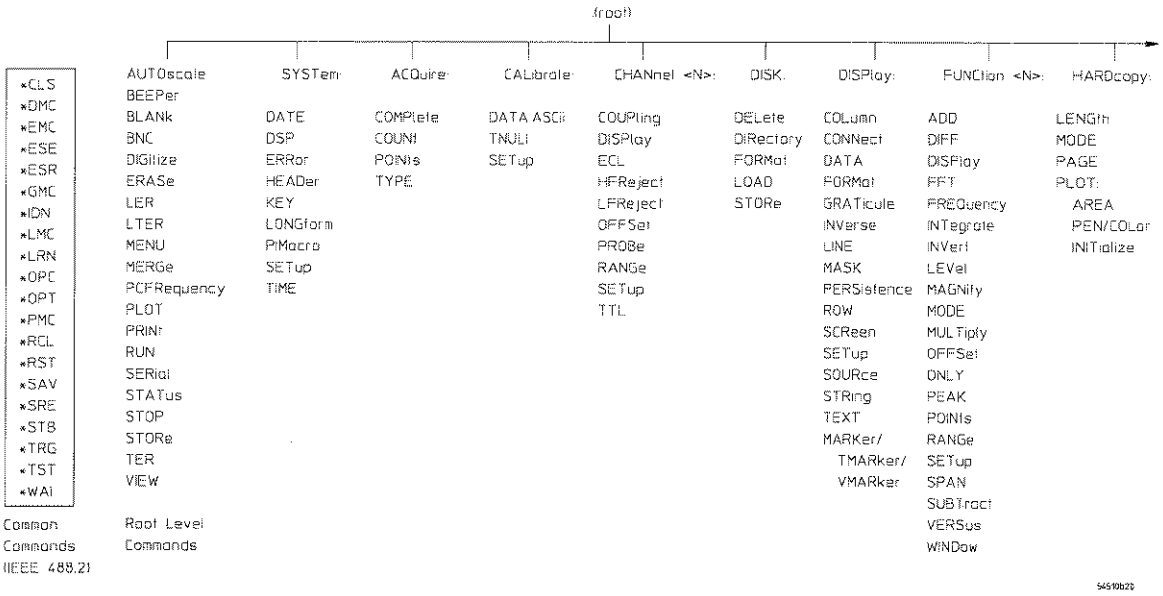
Common commands are independent of the subsystem commands, and do not affect the position of the parser within the command tree.

```
*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. These commands differ from root level commands in that root level commands place the parser back at the root of the command tree.

```
:AUTOSCALE
```

Figure 3–1



Command Tree

Figure 3–2

Command Tree (continued)

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 in figure 3-1 would be :CHANnel1:RANGe. This is referred to as 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, figure 3-1, use the last mnemonic in the compound header as a reference point (for example, RANGe). Then find the last colon above that mnemonic (:CHANnel<n>:). That is the point where the parser resides. Any command below that point can be sent within the current program message without sending the mnemonics which appear above them (for example, OFFSet).

Examples

The OUTPUT statements in the examples are written using HP BASIC 5.0 on a HP 9000 Series 200/300 Controller. The quoted string is placed on the bus, followed by a carriage return <CR> and linefeed <LF>.

Example 1

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

Comments: The colon between CHANnel1 and RANGe is necessary because CHANnel1:RANGe is a compound command. The semicolon between the RANGe command and the OFFSet command is the required

program message unit separator. The OFFSet command does not need CHANnel1 preceding it, since the CHANnel1:RANGe command sets the parser to the CHANnel1 node in the tree.

Example 2

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

or

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

Comments: In the first line of example 2, the "subsystem selector" is implied for the DELay command in the compound command. The DELay command must be in the same program message as the REFerence command, since the program message terminator places the parser back at the root of the command tree.

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

Example 3

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

Comments: The leading colon before CHANnel1 tells the parser to go back to the root of the command tree. The parser can then see the CHANnel1:OFFSet command.

Infinity Representation

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

Sequential and Overlapped Commands

IEEE 488.2 makes the distinction between sequential and overlapped commands. Sequential commands finish their task before the execution of the next command starts. Overlapped commands run concurrently. Commands following an overlapped command may be started before the overlapped command is completed. All of the commands of the oscilloscope 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 instrument is capable of buffering multiple queries.

Notation Conventions and Definitions

The following conventions and definitions are used in this manual in descriptions of remote HP-IB operation:

Conventions

<>	Angular brackets enclose words or characters that symbolize a program code parameter or an HP-IB command.
::=	"is defined as." For example, <A> ::= <B> indicates that <A> can be replaced by <B> in any statement containing <A>.
|	"or." Indicates a choice of one element from a list. For example, <A> | <B> indicates <A> or <B> but not both.
. . .	An ellipsis (trailing dots) indicate 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 may 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>	::= space character.
<white_space>	::= space(s) and tabs.

Syntax Diagrams

Chapters 5 through 23 contain syntax diagrams showing the proper syntax for each command. All characters contained in a circle or oblong are literals, and must be entered exactly as shown. Words and phrases contained in rectangles are names of items used with the command and are described in the accompanying text of each command. Each line can only be entered from one direction as indicated by the arrow on the entry line. Any combination of commands and arguments that can be generated by following the lines in the proper direction is syntactically correct. An argument is optional if there is a path around it. Where there is a rectangle which contains the word "space," a white space character must be entered. White space is optional in many other places.

Program Examples

The program examples given for each command in chapters 5 through 23 were written on an HP 9000 Series 200/300 controller using the HP BASIC 5.0 programming language. The programs always assume the oscilloscope is at address 707. If a printer is used, it is always assumed to be at address 701.

In these examples, pay 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 channel 1 range value of 100 mV, that value could be sent using a decimal (.1), or an exponential (1e-1 or 1.0E-1), or a suffix (100 mV or 100MV).

As an example, set channel 1 range to 100 mV by sending one of the following:

- Commands in long form using the decimal format.

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

- Commands in short form using an exponential format.

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

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

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

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

If you want to observe the headers for the queries, you must bring the returned data into a string variable. Generally, you should dimension all string variables before reading the data.

If you do not need to see the headers and a numeric value is returned from the oscilloscope, then you should use a numeric variable. In this case the headers should be turned off.

Command Set Organization

The command set for the oscilloscope is divided into common commands, root level commands and subsystem commands. Each of the groups of commands is described in the following chapters. Each of the chapters contain a brief description of the subsystem, a set of syntax diagrams for the commands, and the commands for each subsystem in alphabetic order.

The commands are shown in the long form and short form using upper and lowercase letters. As an example, AUToscale indicates that the long form of the command is AUTOSCALE and the short form of the command is AUT. Each command listing contains a description of the command and its arguments, the command syntax, and a programming example.

The subsystems are listed below:

SYSTem Controls some basic functions of the oscilloscope.

ACQuire Sets the parameters for acquiring and storing data.

CALibrate Sets the time nulls (channel-to-channel skew) and returns the instrument's calibration data.

CHANnel Controls Y-axis oscilloscope functions (corresponds to front panel vertical controls).

DISK Controls all built-in floppy disk drive functions.

DISPlay Controls how waveforms, voltage and time markers, graticule, and text are displayed and written on the screen.

FUNCtion Controls the waveform math functions of the oscilloscope.

HARDcopy Controls the parameters used during the plotting or printing of waveforms.

MARKer Controls x- and y-marker functions.

MEASure Selects the automatic measurements to be made.

MMEMory Controls multiple memory functions.

PMEMory Controls pixel memory functions.

SEQuential Controls sequential memory mode configuration and display parameters.

TIMebase Controls X-axis oscilloscope functions (corresponds to front panel horizontal controls).

TRIGger Controls the trigger modes and parameters for each trigger mode.

WMEMory Controls waveform memory functions.

WAVeform Provides access to waveform data, including active data from channels and functions as well as static data from waveform memories.

4

Example Programs

Introduction

This chapter contains example programs using the command set for the oscilloscope. Each command has a separate output statement for clarity. To optimize speed, switch to the concatenated short form numerics.

Throughout these examples, the oscilloscope is assumed to be at address 7, the hardcopy devices at address 1. The HP-IB system bus address is assumed to be at 7, and the National interface address is derived using the IBDEV command. The input signal used is the PROBE COMP signal from the front panel of the instrument. This signal is connected to channel 1 used in 10-1 probe.

The programs are categorized by programming language used, and by the interface module used. Examples in the four different platforms are shown:

- HP BASIC 5.0 running on an HP Series 200/300 controller.
- Microsoft QuickBASIC 4.5 running on IBM PC compatible.
- Microsoft QuickC 2.5 running on IBM PC compatible.
- Instrument-BASIC (I-BASIC) for windows running on IBM PC compatible.

Two 3.5 inch floppy disks are included with this manual, one in DOS format for the PC environment, and one in LIF format for the HP Series 200/300 controller environment.

- The DOS disk contains all the HP I-BASIC, Microsoft QuickBASIC, and Microsoft QuickC programs, as well as copies of the HP-BASIC programs (HP-BASIC will not execute on DOS).
- The LIF disk contains only the HP-BASIC programs.

The sample programs are stored on the DOS disk in the following sample directories:

HP-BASIC Contains examples in HP BASIC 5.0 running on an HP Series 200/300 controller.

QB-HPIB Contains examples in Microsoft QuickBASIC 4.5 running on IBM PC compatible using the HP 82335A HP-IB interface module.

QB-NI Contains examples in Microsoft QuickBASIC 4.5 running on IBM PC compatible using the National Instruments PCII or PCIIA IEEE 488.2 interface module.

QC-HPIB Contains examples in Microsoft QuickC 2.5 running on IBM PC compatible using the HP 82335A HP-IB interface module.

QC-NI Contains examples in Microsoft QuickC 2.5 running on IBM PC or compatibles using the National Instruments PCII or PCIIA IEEE 488.2 interface module.

I-BASIC Contains examples in I-BASIC for Windows running on IBM PC compatible using the HP 82335A HP-IB interface module.

To use the QuickBASIC and QuickC programs on the remote interface, you must have the HP 82335A Remote Interface Command Library installed in your PC. See the manual for HP 82335A for information on how to use the library with QuickBASIC and QuickC.

There are more example files contained on the diskette. In each sub-directory, there is a text file called README. It contains the latest information concerning the example files.

HP Basic Initialize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize the input signal (in this case channel 1), then measure frequency and print the result. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe.

```
10    !INIT                    HP Basic program
20    !
30    !MAIN PROGRAM
40    !
50    CLEAR SCREEN
60    PRINT "This example program will perform the following tasks:"
70    PRINT "          a.  initialize the interface and scope"
80    PRINT "          b.  digitize signal                    "
90    PRINT "          c.  measure and print the frequency    "
100   PRINT
110   PRINT "The program assumes the system is configured as:"
120   PRINT "          HP-IB address = 7"
130   PRINT "          scope address = 7"
140   PRINT "          signal attached to channel 1 with 10:1 probe"
150   PRINT
160   PRINT "If the addresses are not correct for your configuration, change"
170   PRINT "the ASSIGN statements in the Initialize function."
180   PRINT
190   PRINT "Press Continue when ready to start program, or Shitf/Break to
      terminate."
200   PAUSE
210   GOSUB Initialize                    !initialize interface and scope
220   GOSUB Get_waveform                  !digitize signal
230   GOSUB Measure                       !measure and print frequency
240   STOP
250   !
260   !INITIALIZE INTERFACE AND SCOPE
270   !
280 Initialize:                           !
290   ASSIGN @Scope TO 707                !scope address
300   ASSIGN @Isc TO 7                    !HP-IB address
310   CLEAR @Isc                          !clear HPIB interface
320   OUTPUT @Scope;"*RST"                !set scope to default config
330   OUTPUT @Scope;":AUTOSCALE"          !AUTOSCALE
340   OUTPUT @Scope;":SYST:HEADER OFF"    !turn headers off
```

```
350   OUTPUT @Scope;":CHAN1:PROBE 10"     !set probe to 10:1
355   CLEAR SCREEN                        !clear screen
360   RETURN
370     !
380     !DIGITIZE waveform to acquire data and stop scope for further
390     !measurement.  Measurement is NOT displayed on front panel.
400     !
410 Get_waveform:                                 !clear screen
430   OUTPUT @Scope;":DIGITIZE CHAN1"             !macro to acquire data & stop
440   RETURN
450     !
460     !have scope to a frequency measurement and read results into
470     !controller.
480     !
490 Measure:                                      !
500   OUTPUT @Scope;":MEASURE:FREQUENCY?"         !FREQUENCY query
510   ENTER @Scope;Value                          !read from scope
520   PRINT "FREQUENCY = ";Value;"Hz"
521   OUTPUT @Scope;":MEAS:VPP?"                  !Vpp query
522   ENTER @Scope;Value
523   PRINT "Vpp = ";Value;"V"
530   RETURN
540   END
```

HP Basic Digitize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize and acquire the input data (in this case channel 1), then store the acquired data on disk. The stored data is then retrieved and the input signal is displayed on the computer screen. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe.

```
10    !DIG            HP Basic program
20    !
30    !MAIN PROGRAM
40    !
50    REAL Preamble(1:10)                !array for preamble information
60    !
70    CLEAR SCREEN
80    PRINT "This example program will perform the following tasks:"
90    PRINT "              a.  initialize interface and scope"
100   PRINT "              b.  digitize and acquire data"
110   PRINT "              c.  store data to disk"
120   PRINT "              d.  retrieve data from disk"
130   PRINT "              e.  draw signal on computer"
140   PRINT
150   PRINT "This program assumes the system is configured as:"
160   PRINT "    HP-IB address = 7"
170   PRINT "    scope address = 7"
180   PRINT "    signal attached to channel 1"
190   PRINT
200   PRINT "If the addresses are not correct, change the ASSIGN "
210   PRINT "statements in the Initialize function."
220   PRINT
230   PRINT "Press Continue when ready to start"
240   PAUSE
250   GOSUB Initialize                   !initialize interface and scope
260   GOSUB Get_waveform                 !dig & acquire signal
270   GOSUB Save_waveform                !store to disk
280   !
290   CLEAR SCREEN
300   PRINT "The waveform data and preamble information have now been"
310   PRINT "read from the scope and stored in the computer's disk."
320   PRINT "This information will now be retrieved from the disk, and"
330   PRINT "will be used to plot the waveform, calculate and plot the"
340   PRINT "integral, as well as calculate scaling information."
```

```
350   PRINT "Press CONTINUE to continue."
360   PAUSE
370   GOSUB Retrieve_wave                  !retrieve from disk
380   GOSUB Graph                          !draw waveform
390   STOP
400   !
410   !INITIALIZATION INTERFACE AND SCOPE
420 Initialize:                            !
430   PRINT "INITIALIZE"
440   ASSIGN @ TO 7                        !Interface Select Code = 7
450   ASSIGN @ TO 707                      !scope address
460   CLEAR @                              !clear HP-IB interface
470   OUTPUT @Scope;"*RST"                 !set scope to default config
480   OUTPUT @Scope;":AUTOSCALE"           !AUTOSCALE
490   OUTPUT @Scope;":system:header off" !turn headers off
500     !
510     !The following 4 commands are the default configuration setting
520     !that *RST sets up; but, they are included here for clarity and
530     !completeness.  This ensures scope is configured for DIGITIZE, if
540     !*RST was not done.
550     !
560   OUTPUT @Scope;":WAVEFORM:SOURCE CHAN1"  !set source to channel 1
570   OUTPUT @Scope;":ACQUIRE:TYPE NORMAL"    !set to normal acquisition mode
580   OUTPUT @Scope;":ACQUIRE:COMPLETE 100"   !set complete criteria
590   OUTPUT @Scope;":ACQUIRE:POINTS 512"     !set # of pts to 512
600     !
610   OUTPUT @Scope;":WAVEFORM:FORMAT COMP"   !acquired data in 8-bit format
620   CLEAR SCREEN
630   RETURN
640  !
650   !
660   !complete scope configuration; DIGITIZE and acquire waveform data
670   !and preamble (voltage/timing) information into computer.
680   !
690 Get_waveform:                          !
700   OUTPUT @Scope;":DIGITIZE CHAN1"          !macro to acquire data & stop
710   OUTPUT @Scope;":WAVEFORM:DATA?"          !query scope for data
720   ENTER @Scope USING "#,1A,";One_char$     !strip off header & size
730   IF One_char$="#" THEN Found_pound
740   PRINT "BAD DATA"
750   STOP
760 Found_pound:     !read record length from header
770   ENTER @Scope USING "#,1D";Digits         !get length of record
780   ENTER @Scope USING "#,"&VAL$(Digits)&"D";Length
```

```
790   PRINT "reading ";Length;" bytes from scope"
800   !
810   !allocate an array for the waveform data.  After the array has been
820   !read in, one extra byte read is done to input the line feed (10)
830   !attached to the end of the scope's output buffer.
840   !
850   ALLOCATE INTEGER Waveform(1:Length)
860   ENTER @Scope USING "#,B";Waveform(*)     !read waveform information
870   ENTER @Scope USING "-K,B";End$           !get last byte (line feed)
880   OUTPUT @Scope;":WAVEFORM:PREAMBLE?"      !query for preamble
890   ENTER @Scope;Preamble(*)                 !read preamble information
900   RETURN
910   !
920   !
930   !plot waveform data to display
940   !
950 Graph:                                     !
960   GCLEAR                                   !initialize graphics
970   CLEAR SCREEN
980   GINIT
990   GRAPHICS ON
1000  VIEWPORT 0,130,35,100
1010  WINDOW 1,Preamble(3),0,255
1020  FRAME
1030  PEN 4
1040  MOVE 0,0
1050  FOR I=1 TO Preamble(3)                   !plot data points
1060    MOVE I,Waveform(I)
1070    DRAW I,Waveform(I)
1080  NEXT I
1090  Srange=Preamble(3)*Preamble(5)/10
1100  Vrange=256*Preamble(8)
1110  Offset=(128-Preamble(10))*Preamble(8)+Preamble(9)
1120  PRINT TABXY(0,18),"Vertical=";Vrange/8;" V/div";TAB(50),"Offset =
";Offset;"V"
1130  PRINT TABXY(0,19),"Time=";Srange;" s/div"
1140  RETURN
1150    !
1160    !save waveform data and preamble information to computer disk
1170    !
1180 Save_waveform:                            !
1190  ON ERROR GOTO 1210
1200  PURGE "WAVESAMPLE"
1210  OFF ERROR
```

```
1220  CREATE BDAT "WAVESAMPLE",1,4080
1230  ASSIGN @Path TO "WAVESAMPLE"
1240  OUTPUT @Path;Waveform(*),Preamble(*)
1250  RETURN
1260   !
1270   !retrieve waveform data and preamble information from disk
1280   !
1290 Retrieve_wave:                          !
1300  ASSIGN @Path TO "WAVESAMPLE"
1310  ENTER @Path;Waveform(*),Preamble(*)
1320  RETURN
1330  END
```

Microsoft QuickBASIC Initialize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize the input signal (in this case channel 1), then measure frequency and print the results. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the HP 82335A HP-IB interface module is being used.

```
DECLARE SUB get.waveform ()
DECLARE SUB measure ()
DECLARE SUB Initialize ()
DECLARE SUB send (cmd$)
'Filename  :  QB-HPIB\INIT.bas

REM $INCLUDE: 'c:\hpib\qbsetup'
DIM SHARED isc&, scope&

CLS                                          'clear screen
PRINT "This example program will perform the following tasks:"
PRINT "          a.  initialize the interface and scope"
PRINT "          b.  digitize signal"
PRINT "          c.  measure and print the frequency and Vpp"
PRINT
PRINT "The sample program assumes the following address settings:"
PRINT "      HP-IB interface select code = 7"
PRINT "      scope address = 707"
PRINT "Set these variables as needed for your system."
PRINT
PRINT "Connect the 10:1 probe to channel 1 and to the probe comp signal"
PRINT "      on the front panel."
PRINT
PRINT "PRESS: 1 TO CONTINUE"
PRINT "       2 TO TERMINATE"
PRINT "?";
BEEP
LINE INPUT cmd$
IF cmd$ = "2" THEN SYSTEM
isc& = 7                                     'HP-IB interface select code
scope& = isc& * 100 + 7                      'isc& *100 + scope address

CALL Initialize                              'initialize interface and scope
CALL get.waveform                            'tell scope to acquire data
```

```
CALL measure                              'measure frequency of input signal
END

SUB get.waveform
   send (":digitize channel1")              'tell scope to acquire data
END SUB

SUB Initialize
SHARED pcib.err, pcib.baserr, noerr

 CALL ioreset(isc&)                              'set interface to start-up state
 IF pcib.err <> noerr THEN ERROR pcib.baserr 'check for error
 timeout! = 20!
 CALL iotimeout(isc&, timeout!)          'set timeout to 10 seconds
 IF pcib.err <> noerr THEN ERROR pcib.baserr 'check for error
 CALL ioclear(isc&)                              'clear devices attached to the
interfaces to known state
 IF pcib.err <> noerr THEN ERROR pcib.baserr 'check for error
 CLS
 send ("*cls")                                   'clear status registers
 send ("*rst")                                   'reset the scope
 send (":autoscale")                             'autoscale the input signal
 send (":system:header off")                     'turn headers off
 send (":chan1:probe 10")                        'set probe to 10:1
   '
   'The following acquire subsystem commands are used to specify
   'the acquisition mode, percent completion, and number of points
   'for the DIGITIZE.  These are the defaults set by the *RST, but
   'are included here to stress the fact that you can control them.
   '
   send (":acquire:type normal")                 'normal acquisition mode
   send (":acquire:complete 100")                '100% completion criteria
   send (":acquire:points 512")                  '500 points acquired
   CLS                                           'clear PC screen
END SUB

SUB measure
SHARED pcib.err, pcib.baserr, noerr
   send (":measure:frequency?")                  'Query for result of frequency
measurement
   CALL ioenter(scope&, freq!)                   'Get the number from the scope
   IF pcib.err <> noerr THEN ERROR pcib.baserr 'Check if any errors occur
   PRINT "Frequency = "; freq!; " Hz"            'Print out the result
   send (":measure:vpp?")                        'query for result of Vpp
```

```
   CALL ioenter(scope&, vpp!)                    'get result from scope
   IF pcib.err <> noerr THEN ERROR pcib.baserr
   PRINT "Vpp = "; vpp!; " V"
END SUB

SUB send (cmd$)
SHARED pcib.err, pcib.baserr, noerr
   CALL iooutputs(scope&, cmd$, LEN(cmd$))       'send command to scope
   IF pcib.err <> noerr THEN ERROR pcib.baserr 'check for any error
END SUB
```

Microsoft QuickBASIC Digitize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize and acquire the input data (in this case channel 1), then perform a Vpp and frequency measurement. The waveform is also stored on disk. The stored waveform is then retrieved and displayed on the computer screen. Then the integral is calculated and displayed on the computer screen. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the HP 82335A HP-IB interface module is being used.

```
DECLARE SUB initialize ()
DECLARE SUB capture.waveform ()
DECLARE SUB measure ()
DECLARE SUB get.waveform ()
DECLARE SUB save.waveform ()
DECLARE SUB retrieve.waveform ()
DECLARE SUB graph ()
DECLARE SUB integrate ()
DECLARE SUB send (cmd$)

'Filename : \QB-HPIB\DIGI.bas

REM $INCLUDE: 'c:\hpib\qbsetup'
DIM SHARED preamble!(10)            'preamble data array
DIM SHARED waveform%(8000)          'waveform data array
DIM SHARED scope&, isc&             'scope/hpib address variable

'MAIN PROGRAM
CLS                                 'clear screen
PRINT "This sample program will perform the following tasks:"
PRINT "       a.  initialize the HP-IB interface and scope"
PRINT "       b.  digitize the signal"
PRINT "       c.  do a Vpp and frequency"
PRINT "       d.  transfer waveform data to PC"
PRINT "       e.  store waveform data on PC disk"
PRINT "       f.  retrieve waveform data from PC disk"
PRINT "       g.  draw signal on PC display"
PRINT "       h.  calculate and display integral"
PRINT
PRINT "The sample program assumes the following address settings:"
PRINT "     HP-IB interface select code = 7"
PRINT "     scope address = 7"
```

```
PRINT "Set these variables as needed for your system."
PRINT
PRINT "Connect the 10:1 probe to channel 1 and to the probe comp signal"
PRINT "      on the front panel."
PRINT
PRINT "PRESS:  1 to CONTINUE "
PRINT "        2 to TERMINATE"
BEEP
PRINT "?";
LINE INPUT cmd$
IF cmd$ = "2" THEN SYSTEM
isc& = 7                              'HP-IB Interface Select Code
scope& = isc& * 100 + 7               'isc& * 100 + scope address

CALL initialize                       'initialize interface & scope
CALL capture.waveform                 'digitize signal
CALL measure                          'do Vpp and Frequency measurement
CALL get.waveform                     'bring waveform & preamble into computer
CALL save.waveform                    'store preamble & waveform to disk
CALL retrieve.waveform                'retrieve preamble & waveform from disk
CALL graph                            'plot waveform data & scaling info
CALL integrate                        'calculate & plot integral
END

SUB capture.waveform
  send (":digitize channel1")           'capture data for channel 1
END SUB

SUB get.waveform
SHARED PCIB.ERR, PCIB.BASERR, NOERR

 send (":waveform:format word")         '16 bit word format
 send (":waveform:preamble?")           'query for config information
 max.len% = 10
 actual.len% = 0
 CALL ioentera(scope&, SEG preamble!(1), max.len%, actual.len%)
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

 send (":waveform:data?")               'query for waveform data
 max.len% = 8000                        'max size of data block
 actual.len% = 0
 CALL ioenterab(scope&, SEG waveform%(1), max.len%, actual.len%, 2)
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
```

```
 max.len% = 1                          'get new line char at end of data block
 actual.len% = 0
 msg$ = SPACE$(max.len%)               'allocate the space
 CALL ioenters(scope&, msg$, max.len%, actual.len%)
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
END SUB

SUB graph
 SCREEN 9                                    'set screen mode to EGA 64k
                                             '640 x 350 x 16
 VIEW (1, 1)-(500, 256), , 15                'set viewport and draw the border
 WINDOW (0, 0)-(10, 8)                       'prepare to draw the grid
 COLOR 8                                     'set color
 FOR i% = 1 TO 9
   LINE (i%, 0)-(i%, 8)
 NEXT i%
 FOR i% = 1 TO 7
   LINE (0, i%)-(10, i%)
 NEXT i%
 WINDOW (1, 0)-(preamble!(3), 32640)         'use window to set coordinates
 COLOR 10
 FOR i% = 1 TO preamble!(3)
  PSET (i%, waveform%(i%))                   'draw data point
 NEXT i%
 WIDTH 80, 25
 VIEW PRINT 20 TO 24
 send (":chan1:range?")                      'query scope for vertical
sensitivity
 CALL ioenter(scope&, vrange!)
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
 LOCATE 20, 1
 PRINT vrange! / 8; " V/div"
 send (":chan1:offset?")
 CALL ioenter(scope&, offset!)
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
 PRINT "Offset = "; offset!; " V"
 send (":timebase:range?")
 CALL ioenter(scope&, trange!)
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
 LOCATE 20, 41
 PRINT trange! / 10; " S/div"
 send (":timebase:delay?")
 CALL ioenter(scope&, delay!)
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
```

```
 LOCATE 21, 41
 PRINT " Delay = "; delay!; " S"
 PRINT "press a key to see the integral of the signal plotted"
 LINE INPUT dummy$
END SUB

SUB initialize
SHARED PCIB.ERR, PCIB.BASERR, NOERR
 CLS
 CALL ioreset(isc&)                    'reset HP-IB interface
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
 timeout.val! = 20!
 CALL iotimeout(isc&, timeout.val!)  'set timeout to 10 second
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
 CALL ioclear(isc&)                  'clear interface & scope
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR
 send ("*cls")                       'clear status registers
 send ("*rst")                       'reset scope to default configuration
 send (":autoscale")                 'perform autoscale to find signal
 send (":chan1:probe 10")            'set probe to 10:1
 send (":system:header off")         'for 54500: turn headers off
 send (":acquire:type normal")       'acquisition type normal (default)
 send (":acquire:points 512")        'set number of points to 512
 send (":waveform:source channel1") 'source of data is channel 1 (default)
END SUB

SUB integrate
DIM math!(preamble(3))                      'define integral array
math!(0) = 0                                'initialize for summation
FOR i% = 1 TO preamble!(3)
  math!(i%) = math!(i% - 1) + (waveform%(i%) - preamble!(10)) * preamble!(8)
+ preamble!(9)
NEXT i%
max! = math!(1)
min! = math!(1)
FOR i% = 1 TO preamble!(3)
   IF math!(i%) > max! THEN max! = math!(i%)
   IF math!(i%) < min! THEN min! = math!(i%)
NEXT i%
WINDOW (1, min!)-(preamble!(3), max!)       'set up proper scale
COLOR 12
FOR i% = 1 TO preamble!(3)
   PSET (i%, math!(i%))
NEXT i%
```

```
BEEP

END SUB

SUB measure
SHARED PCIB.ERR, PCIB.BASERR, NOERR

  send (":measure:vpp?")                    'query scope for Vpp
  CALL ioenter(scope&, vpp.value!)          'input Vpp
  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

  send (":measure:frequency?")              'query scope for frequency
  CALL ioenter(scope&, frequency!)          'input frequency
  IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

  PRINT "Vpp = "; vpp.value!                'print Vpp
  PRINT "frequency = "; frequency; " Hz"    'print frequency
  PRINT
  BEEP
  INPUT "PRESS Enter TO CONTINUE:", a$      'read keypad

END SUB

SUB retrieve.waveform
 OPEN "wave.dat" FOR BINARY AS #2
 FOR count% = 1 TO 10
    GET #2, , preamble!(count%)
 NEXT count%
 FOR count% = 1 TO preamble!(3)
    GET #2, , waveform%(count%)
 NEXT count%
 CLOSE #2

END SUB

SUB save.waveform
 OPEN "wave.dat" FOR BINARY AS #2
 FOR count% = 1 TO 10
    PUT #2, , preamble!(count%)
 NEXT count%
 FOR count% = 1 TO preamble!(3)
    PUT #2, , waveform%(count%)
 NEXT count%
 CLOSE #2
```

```
END SUB

SUB send (cmd$)
SHARED PCIB.ERR, PCIB.BASERR, NOERR

 CALL iooutputs(scope&, cmd$, LEN(cmd$))
 IF PCIB.ERR <> NOERR THEN ERROR PCIB.BASERR

END SUB
```

Microsoft QuickBASIC Initialize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize the input signal (in this case channel 1), then measure frequency and Vpp and print the results. The program assumes that the PROB COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the National Instruments PCII or PCIIA IEEE 488.2 interface module is being used.

```
DECLARE [illegible] ()
DECLARE [illegible] ()
DECLARE [illegible] (MSG$)
DECLARE [illegible] ()
DECLARE [illegible].scope (cmd$)
'Filename: [illegible] INIT.BAS
'
REM $INCLUDE: 'GPIB-PC\QBDECL.BAS'

DIM SHARED [illegible]                    'scope address variable
'
'MAIN PROGRAM
'
CLS
PRINT "This example program will perform the following tasks:"
PRINT "          a.  initialize the interface and scope"
PRINT "          b.  digitize signal"
PRINT "          c.  measure and print the frequency and Vpp"
PRINT
PRINT "The sample program assumes the following address settings:"
PRINT "      scope address = 7"
PRINT "Set the scope or program variable as needed for your system."
PRINT "IBDEV is used to set up GPIB bus.  IBFIND could also be used."
'use ibdev or ibfind to set scope address
         'gpib card = 0
         'scope = 7
         'secondary address = don't care
         'timeout = 13 implies 10 s wait;  see IBTIMEOUT FOR VALUES
         'end of message enabled = 1
         'end of string mode disabled = 0
         'name of scope = scope%
CALL IBDEV(0, 7, 0, 13, 1, 0, scope%)
PRINT
PRINT "PRESS: 1 TO CONTINUE"
```

```
PRINT "       2 TO TERMINATE"
PRINT "?";
BEEP
LINE INPUT cmd$
IF cmd$ = "2" THEN SYSTEM
CALL initialize                    'initialize interface & scope
CALL digitize                      'have scope acquire a signal
CALL measure                       'find Vpp and Frequency
CALL IBLOC(scope%)
  IF (IBSTA% AND EERR) THEN CALL GPIBERR("IBLOC ERROR")
CALL IBONL(scope%, 0)
  IF (IBSTA% AND EERR) THEN CALL GPIBERR("IBONL ERROR")
END

SUB digitize
send.to.scope (":digitize channel1")         'tell scope to acquire data
END SUB
SUB GPIBERR (MSG$) STATIC

PRINT MSG$

PRINT "IBSTA=&H"; HEX$(IBSTA%); "<";
IF IBSTA% AND EERR THEN PRINT " ERR";
IF IBSTA% AND TIMO THEN PRINT " TIMO";
IF IBSTA% AND EEND THEN PRINT " END";
IF IBSTA% AND SRQI THEN PRINT " SRQI";
IF IBSTA% AND RQS THEN PRINT " RQS";
IF IBSTA% AND CMPL THEN PRINT " CMPL";
IF IBSTA% AND LOK THEN PRINT " LOK";
IF IBSTA% AND RREM THEN PRINT " REM";
IF IBSTA% AND CIC THEN PRINT " CIC";
IF IBSTA% AND AATN THEN PRINT " ATN";
IF IBSTA% AND TACS THEN PRINT " TACS";
IF IBSTA% AND LACS THEN PRINT " LACS";
IF IBSTA% AND DTAS THEN PRINT " DTAS";
IF IBSTA% AND DCAS THEN PRINT " DCAS";
PRINT " "
PRINT "IBERR="; IBERR%;
IF IBERR% = EDVR THEN PRINT " EDVR <DOS Error>"
IF IBERR% = ECIC THEN PRINT " ECIC <Not CIC>"
IF IBERR% = ENOL THEN PRINT " ENOL <No Listener>"
IF IBERR% = EADR THEN PRINT " EADR <Address error>"
IF IBERR% = EARG THEN PRINT " EARG <Invalid argument>"
IF IBERR% = ESAC THEN PRINT " ESAC <Not Sys Ctrlr>"
```

```
IF IBERR% = EABO THEN PRINT " EABO <Op. aborted>"
IF IBERR% = ENEB THEN PRINT " ENEB <No GPIB board>"
IF IBERR% = EOIP THEN PRINT " EOIP <Async I/O in prg>"
IF IBERR% = ECAP THEN PRINT " ECAP <No capability>"
IF IBERR% = EFSO THEN PRINT " EFSO <File sys. error>"
IF IBERR% = EBUS THEN PRINT " EBUS <Comand error>"
IF IBERR% = ESTB THEN PRINT " ESTB <Status byte lost>"
IF IBERR% = ESRQ THEN PRINT " ESRQ <SRQ stuck on>"
IF IBERR% = ETAB THEN PRINT " ETAB <Table Overflow>"
PRINT "IBCNT="; IBCNT%
'  Call the IBONL function to disable the device DVM.
CALL IBONL(DVM%, 0)
STOP
END SUB

SUB initialize
DIM reading AS STRING * 30
CLS
CALL IBCLR(scope%)
send.to.scope ("*rst")              'reset scope to default configuration
send.to.scope ("*cls")              'clear status registers
send.to.scope (":autoscale")        'perform autoscale to find signal
'
'The following 3 commands are added for clarity and completeness
'but are the default values set by the *RST command.
send.to.scope (":acquire:points 512")        'set # of points to 512
send.to.scope (":acquire:type normal")       'normal acquisition mode
send.to.scope (":acquire:complete 100")      'completion criteria set
END SUB

SUB measure
CLS
send.to.scope (":MEAS:VPP?")          'vpp query
reading$ = SPACE$(100)
CALL ibrd(scope%, reading$)
   IF (IBSTA% AND EERR) THEN CALL GPIBERR("read of vpp ERROR")
vpp = VAL(reading$)                   'convert string
PRINT "vpp = "; vpp; " V"
send.to.scope (":MEAS:FREQ?")          'vpp query
CALL ibrd(scope%, reading$)
   IF (IBSTA% AND EERR) THEN CALL GPIBERR("read of freq ERROR")
freq = VAL(reading$)                   'convert string
PRINT "freq = "; freq; " Hz"
PRINT
```

```
END SUB

SUB send.to.scope (cmd$)
CALL IBWRT(scope%, cmd$)
   IF (IBSTA% AND EERR) THEN CALL GPIBERR("send.to.scope ERROR")
END SUB
```

Microsoft QuickC Initialize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize the input signal (in this case channel 1), then measure frequency and Vpp and print the results. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the HP 82335A HP-IB interface module is being used.

```
#include "c:\hpib\chpib.h"
#include "c:\hpib\cfunc.h"
#include <stdio.h>
#include <graph.h>

send (char *cmd);
initialize();
get_waveform();
measure();
/*Filename : \QC-HPIB\INIT.C */
#define ISC     7L                      /* select code of HP-IB interface */
#define SCOPE   707L                    /* isc*100 + scope address */

main()
{
   char ch;
   _clearscreen(_GCLEARSCREEN);              /* clear screen of PC     */

   printf ("This example program will perform the following tasks:\n");
   printf ("     1.  initialize the interface and scope\n");
   printf ("     2.  digitize the signal\n");
   printf ("     3.  measure and print the frequency and VPP\n");
   printf ("\nConnect 10:1 probe from channel 1 to probe comp signal\n");
   printf ("on the front panel.\n");
   printf ("\nThe sample program assumes the following address settings:\n");
   printf ("     HP-IB interface select code = 7\n");
   printf ("     scope address = 7\n");
   printf ("Set these constants in the define statements for your system\n");
   printf ("\nPRESS Enter TO CONTINUE OR Ctrl C TO TERMINATE\n");
   ch = getch();                /* read keyboard                        */
   initialize();                /* initialization of interface & scope */
   get_waveform();              /* tell scope to get waveform           */
   measure();                   /* make measurement and print result    */
}
```

```
error_handler(int error, char *routine)
{
char    ch;
if (error != NOERR)
   {
      printf("Error in call to %s \n", routine);
      printf("%d %s \n", error, errstr(error));
      printf("Press Enter to continue: ");
      scanf ("%c",&ch);
   }
}

initialize()
{
   char ch;
   error_handler(IORESET(ISC), "ioreset");          /* reset interface */
   error_handler(IOTIMEOUT(ISC, (double)20), "iotimeout");  /* timeout */
   error_handler(IOCLEAR(ISC), "ioclear");          /* clear interface */
   send ("*RST");                                /* reset scope to default */
   send ("*CLS");                                /* clear status registers */
   send (":autoscale");                          /* autoscale input signal */
   send (":chan1:probe 10");                     /* set probe to 10:1      */
   send (":system:header off");                  /* turn headers off       */
   _clearscreen(_GCLEARSCREEN);                  /* clear screen of PC     */
}

get_waveform()
{
  send (":digitize channel1");             /* capture data            */
}

measure()
{
  float        freq,vpp;
  send (":measure:frequency?");          /* query frequency of input signal
*/
  error_handler(IOENTER(SCOPE, &freq), "ioenter"); /* get freq from scope
*/
                                  /* noted freq must be passed by ref*/
  printf ("Frequency = %e Hz \n", freq); /* print it out
*/
  send (":measure:vpp?");                /* query Vpp of input signal
*/
```

```
  error_handler(IOENTER(SCOPE,&vpp), "ioenter");   /* get vpp from scope    */
  printf ("Vpp = %e V \n",vpp);          /* print Vpp                        */
}

send(char *cmd)
{  /* this subroutine send the char string pointed by cmd to scope          */
   error_handler(IOOUTPUTS(SCOPE, cmd, strlen(cmd)), cmd);
}
```

Microsoft QuickC Digitize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize and acquire the input data (in this case channel 1), then perform a Vpp and frequency measurement. The waveform is also stored on disk. The stored waveform is then retrieved and displayed on the computer screen. Then the integral is calculated and displayed on the computer screen. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the HP 82335A HP-IB interface module is being used.

```
#include "c:\hpib\chpib.h"
#include "c:\hpib\cfunc.h"
#include <stdio.h>
#include <graph.h>

initialize();
get_waveform();
save_waveform();
retrieve_waveform();
graph_waveform();
integrate_waveform();
send (char *cmd);
/* Filename : \QC-HPIB\DIGI.C                                          */
#define ISC     7L              /* select code of HP-IB interface      */
#define SCOPE   707L            /* isc*100 + scope address             */
#define TRUE    1

float           preamble[10];       /* array to hold the preamble data */
unsigned char   waveform[8000];     /* array to hold the waveform data */
float           math[8001];         /*array for integral calculation   */
void main()
{
   char ch;
   _clearscreen (_GCLEARSCREEN);     /* clear PC's screen */

   printf ("This sample program will perform the following tasks:\n");
   printf ("    a.  initialize the HP-IB interface and scope\n");
   printf ("    b.  digitize the signal\n");
   printf ("    c.  do a Vpp and Frequency measurement on the signal\n");
   printf ("    d.  transfer waveform data to PC\n");
   printf ("    e.  store waveform data on PC file\n");
   printf ("    f.  retrieve waveform data from PC file\n");
```

```
    printf ("    g.  draw signal of PC display\n");
    printf ("    h.  calculate and display integral\n\n");
    printf ("The sample program assumes the following address settings:\n");
    printf ("    HP-IB interface select code = 7\n");
    printf ("    scope address = 7\n");
    printf ("Set these constants as needed for your system.\n\n");
    printf ("\nConnect the 10:1 probe from channel 1 to the probe comp\n");
    printf ("signal on the front panel.\n");
    printf ("\nPress Enter to continue or Ctrl C to terminate\n\n");
    ch = getch();

    initialize();                   /* initialize scope and interface */
    capture_waveform();             /* digitize signal                */
    measure();                      /* measure Vpp and Freq           */
    get_waveform();                 /* get the waveform from scope    */
    save_waveform();                /* save the waveform to a file    */
    retrieve_waveform();            /* retrieve the waveform from file*/
    graph_waveform();               /* display the data               */
    integrate_waveform();           /* integrate the waveform and display it */
    _settextcolor (15);
}

error_handler(int error, char *routine)
{
    char    ch;
    if (error != NOERR) {                          /* check for errors   */
       printf("Error in call to %s \n", routine); /* print error info    */
       printf("%d %s \n", error, errstr(error));
       printf("Press Enter to continue: ");
       printf ("%c", ch = getchar());              /* wait for enter key */
    }
}

initialize()
{
    char ch;
    error_handler(IORESET(ISC),"ioreset");      /* reset HP-IB interface   */
    error_handler(IOTIMEOUT(ISC,(double)20),"iotimeout"); /* 20 s timeout  */
    error_handler(IOCLEAR(ISC),"ioclear");      /* clear interface         */
    send ("*rst");                              /* reset scope             */
    send ("*cls");                              /* clear status registers  */
    send (":autoscale");                        /* autoscale input signal  */
    send (":chan1:probe 10");                   /* set probe to 10:1       */
    send (":system:header off");                /* headers off             */
```

```
   /* The following four commands are added for clarity.  They are the
      default values after the *RST command, except for points which is 500
      instead of 8000.                                                    */
   send (":acquire:points 512");          /* get 512 points from scope      */
   send (":acquire:type normal");         /* set acquisition type to normal */
   send (":acquire:complete 100");        /* get data until 100% completion */
   send (":waveform:source channel1");    /* get data from channel 1        */
   _clearscreen (_GCLEARSCREEN);          /* clear PC's screen              */
}

send(char *cmd)
{  /* this routine is added to make sending the command to scope easier to
      read */
   error_handler(IOOUTPUTS(SCOPE, cmd, strlen(cmd)), cmd);
}

capture_waveform()
{
  send (":digitize channel1");            /* tell scope to gather data      */
}

measure()
{
   float        freq,vpp;
   char         ch;
   send (":measure:frequency?");       /* query frequency of input signal  */
   error_handler(IOENTER(SCOPE, &freq), "ioenter"); /* get the freq        */
                          /* note: that freq must be passed by ref */
   printf ("Frequency = %e Hz \n", freq);           /* print it out        */
   send (":measure:vpp?");                /* query Vpp of input signal      */
   error_handler(IOENTER(SCOPE,&vpp), "ioenter");   /* get vpp from scope  */
   printf ("Vpp = %e V \n",vpp);                    /* print Vpp           */
   printf ("\nPRESS Enter TO CONTINUE\n");
   ch = getch();
}

get_waveform()
{
   int  length;
   char msg[20];
   send (":waveform:format comp");        /* use the 8-bit byte format      */
   send (":waveform:preamble?");          /* ask for info about data        */
   length = 10;                           /* totally 10 parameters          */
```

```
    error_handler(IOENTERA(SCOPE,preamble,&length), "ioentera");
    send (":waveform:data?");              /* ask for the waveform data      */
    length = 8000;                         /* max. is 8000 bytes             */
    error_handler(IOENTERAB(SCOPE,waveform,&length,1), "ioenterab");
    length = 1;                            /* get the last newline char      */
    error_handler(IOENTERS(SCOPE,msg,&length), "ioenters");
}

save_waveform()
{
    FILE *fp;
    fp = fopen("wave.dat", "wb");          /* open a file for storage        */
    fwrite(preamble,sizeof(preamble[0]),10,fp); /*write the preamble to file*/
    /* write the data, preamble[2] contains the number of data               */
    fwrite (waveform, sizeof(waveform[0]), (unsigned)preamble[2], fp);
    fclose (fp);                                /* close the file            */
}

retrieve_waveform()
{
    FILE *fp;
    char ch;
    _clearscreen (_GCLEARSCREEN);      /* clear PC's screen */
printf("Waveform data has been saved to a file.  The program will retrieve\n");
printf("the data, graph it and the integral.  All subsequence operations\n");
printf ("are based on the saved information.\n\nPress Enter to continue.\n");
printf ("%c", ch = getchar());
    fp = fopen("wave.dat", "rb");                /* open the file           */
    fread(preamble,sizeof(preamble[0]),10,fp);   /* get the preambles       */
    /* retrieve the waveform data, preamble[2] contains number of points    */
    fread (waveform, sizeof(waveform[0]), (unsigned)preamble[2], fp);
    fclose(fp);                                  /* close the file          */
}

graph_waveform()
{
    int   i;
    char  ch;
    _setvideomode(_ERESCOLOR);                 /* set screen mode to EGA    */
    _rectangle (_GBORDER, 0,0, 501,257);       /* draw border               */
    _setviewport(1,1,500,256);                 /* set viewport              */
    _setwindow (TRUE, 0,0,10,8);               /* set mapping co-ordinates  */
    _setcolor (8);                             /* set color to dark gray    */
    for (i=1;i<10;i++) {                       /* draw the grid             */
```

```
      _moveto_w(i,0);
      _lineto_w(i,8);
   }
   for (i=1;i<8;i++) {
      _moveto_w(0,i);
      _lineto_w(10,i);
   }
   _setwindow(TRUE, 1,0,preamble[2],255);      /* set mapping co-ordinates  */
   _setcolor(10);                              /* set color to bright green */
   for (i=0;i<preamble[2];i++) {               /* draw the pixels           */
      _setpixel_w (i,waveform[i]);
   }
   _settextwindow(20,1,24,80);                 /* set text window           */
   _settextcolor (15);                         /* set color to bright white */
   _settextposition(1,1);                      /* position cursor           */
   printf ("vertical = %11.2e V/div \n", 32*preamble[7]);
   printf ("Offset= %11.3e V \n",
               (128 - preamble[9])*preamble[7]+preamble[8]);
   _settextposition(1,41);
   printf ("horizontal = %11.3e S/div \n", preamble[2]*preamble[4]/10);
   _settextposition(2,41);
   printf ("\nPRESS Enter to plot integral\n");
   ch = getch();
}

integrate_waveform ()
{
   float        min, max;            /* define array to hold math data     */
   int          i;

   math[0] = 0;                      /* starting from zero                 */
   for (i=0;i<preamble[2];i++) {     /* each elt. = last elt + curr wf data */
      math[i+1] = math[i] + (waveform[i]-preamble[9])*preamble[7] +
preamble[8];
   }
   max = math[1];                    /* find out the max & min of integrate */
   min = math[1];                    /* in order to have the proper scale   */
   for (i=1; i<preamble[2]; i++) {
      if (math[i] > max) {max = math[i];}
      if (math[i] < min) {min = math[i];}
   }
   _setwindow(TRUE, 1, min, preamble[2], max);   /* set up proper mapping  */
   _setcolor(12);                                /* use red color          */
   for (i=1; i<preamble[2]; i++)                /* draw the integrate     */
```

```
        _setpixel_w(i, math[i]);
    }
```

Microsoft QuickC Initialize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize the input signal (in this case channel 1), then measure frequency and Vpp and print the results. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the National Instruments PCII or PCIIA IEEE 488.2 interface module is being used.

```
#include <stdio.h>
#include <stdlib.h>
#include <graph.h>
#include "C:\gpib-pc\decl.h"

/* Filename : \QC-NI\INIT.C */

unsigned int  scope;       /*scope address set in initialize     */

void main()
{   char ch;
   _clearscreen (_GCLEARSCREEN);            /* clear PC's screen */
   printf ("This sample program will perform the following tasks:\n");
   printf ("    a.  initialize GPIB interface and scope at address 7\n");
   printf ("    b.  digitize the signal\n");
   printf ("    c.  do a Vpp and Frequency measurement on the signal\n");
   printf ("\nConnect a 10:1 probe from channel 1 to the probe comp \n");
   printf ("signal on the front panel.\n");
   printf ("\nPress Enter to continue or Ctrl C, Enter to terminate\n");
   ch = getch();
   initialize();              /* initialize scope and interface */
   capture_waveform();        /* digitize signal                */
   measure();                 /* measure Vpp and Freq           */
   ibloc(scope);              /* return scope to local          */
   exit(0);
}

send (char *cmd)
{
/*this routine is added to made sending commands easier to read */
ibwrt(scope,cmd,(long)strlen(cmd));
if (ibsta & ERR)
  {
    gpiberr("ibwrt error");
```

```
    exit (1);
  }
}

initialize()
{
int i;
char ch;
/*ibdev or  ibfind to set address of scope */
if ((scope = ibdev(0,7,0,13,1,0)) < 0)
  {
    gpiberr("ibfind Error");
    exit(1);
  }
/*clear the device*/
if (ibclr(scope) & ERR)
  {
   gpiberr("ibclr Error");
   exit(1);
  }
send ("*cls");                  /* clear scope status reg     */
send ("*rst");                  /* reset set to default config*/
send ("aut");                   /* autoscale                  */
send ("chan1:probe 10");         /* set probe to 10:1          */
send (":system:header off");    /* headers off                */

send (":acquire:points 512");   /* get 512 pts on digitize    */
send (":acquire:type normal");  /* normal acq mode            */
send (":acquire:complete 100"); /* set completion criteria    */
_clearscreen(_GCLEARSCREEN);    /* clear pc screen            */
}

capture_waveform()
{
  send(":digitize channel1");   /* acquire data on channel 1  */
}

measure()
{
  char buffer[15];
  float result;
  char ch;
  send(":measure:frequency?");        /* query freq          */
  ibrd(scope,buffer,15);
```

```
  if (ibsta & ERR)
     {
      gpiberr("ibrd error");
      exit(1);
     }
  result = (float)atof(buffer);
  printf(" Frequency = %e Hz \n",result);  /*print frequency*/
  send(dev,"measure:vpp?");              /* query vpp            */
  ibrd(dev,buffer,15);
  if (ibsta & ERR)
     {
      gpiberr("Ibrd error");
      exit(1);
     }
   result = (float)atof(buffer);
   printf("Vpp = %e V \n",result);
   printf(" Press Enter to continue \n");
   ch = getch();
}

gpiberr(char *msg) {
char c;
/* This routine would notify you that an IB call failed.     */
   printf("%s\n", msg);
   printf( "ibsta=&H%x  ", ibsta);
   if (ibsta & ERR ) printf (" ERR");
   if (ibsta & TIMO) printf (" TIMO");
   if (ibsta & END ) printf (" END");
   if (ibsta & SRQI) printf (" SRQI");
   if (ibsta & RQS ) printf (" RQS");
   if (ibsta & CMPL) printf (" CMPL");
   if (ibsta & LOK ) printf (" LOK");
   if (ibsta & REM ) printf (" REM");
   if (ibsta & CIC ) printf (" CIC");
   if (ibsta & ATN ) printf (" ATN");
   if (ibsta & TACS) printf (" TACS");
   if (ibsta & LACS) printf (" LACS");
   if (ibsta & DTAS) printf (" DTAS");
   if (ibsta & DCAS) printf (" DCAS");
   printf (" >\n");
   printf ("iberr= %d", iberr);
   if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
   if (iberr == ECIC) printf (" ECIC <Not CIC>\n");
   if (iberr == ENOL) printf (" ENOL <No Listener>\n");
```

```
    if (iberr == EADR) printf (" EADR <Address error>\n");
    if (iberr == EARG) printf (" EARG <Invalid argument>\n");
    if (iberr == ESAC) printf (" ESAC <Not Sys Ctrlr>\n");
    if (iberr == EABO) printf (" EABO <Op. aborted>\n");
    if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
    if (iberr == EOIP) printf (" EOIP <Async I/O in prg>\n");
    if (iberr == ECAP) printf (" ECAP <No capability>\n");
    if (iberr == EFSO) printf (" EFSO <File sys. error>\n");
    if (iberr == EBUS) printf (" EBUS <Command error>\n");
    if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
    if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
    if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");
    printf ("ibcnt= %d\n", ibcntl);
    printf ("\n");
/* Call the ibonl function to disable the hardware
    and software.                                                    */
    ibonl (scope,0);
    printf("Press Enter to continue: ");
    printf ("%c", ch = getchar());      /* wait for enter key */
}
```

Instrument BASIC for Windows Initialize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize the input signal (in this case channel 1), then measure frequency and Vpp and print the results. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the HP 82335A HP-IB interface module is being used.

```
10     !INIT                    HP Basic program
20     !
30     !MAIN PROGRAM
40     !
50      CLEAR SCREEN
60      GESCAPE CRT,40                    !maximize the alpha window
70      GESCAPE CRT,45                    !place alpha window on top
80      PRINT "This example program will perform the following tasks:"
90      PRINT "          a.  initialize the interface and scope"
100     PRINT "          b.  digitize signal                    "
110     PRINT "          c.  measure and print the frequency    "
130     PRINT "The program assumes the system is configured as:"
140     PRINT "          HP-IB address = 7"
150     PRINT "          scope address = 7"
160     PRINT "          10:1 probe attached to channel 1 from probe"
161     PRINT "               compensation connection on front panel"
180     PRINT "If the addresses are not correct for your configuration,
change"
190     PRINT "the ASSIGN statements in the Initialize function."
191     PRINT
210    !dialog list to continue
220     Msg$="Want to continue?"
230     DIALOG "QUESTION",Msg$,Button;SET("TITLE":"Initial Program")
240     IF Button THEN GOSUB Done
250     CLEAR SCREEN
260     GOSUB Initialize                  !initialize interface and scope
270     GOSUB Get_waveform                !digitize signal
280     GOSUB Measure                     !measure and print frequency
290     GOSUB Done                        !stop and restore
300    !
310    !INITIALIZE INTERFACE AND SCOPE
320    !
330 Initialize:                          !
340     ASSIGN @Scope TO 707             !scope address
```

```
350    ASSIGN @Isc TO 7                    !HP-IB address
360    CLEAR @Isc                          !clear HPIB interface
370    OUTPUT @Scope;"*RST"                !set scope to default config
380    OUTPUT @Scope;":AUTOSCALE"          !AUTOSCALE
381    OUTPUT @Scope;":CHAN1:PROBE 10"     !set probe attenuation to 10:1
390    OUTPUT @Scope;":SYST:HEADER OFF"    !turn headers off
400    CLEAR SCREEN                        !clear screen
410    RETURN
420     !
430     !DIGITIZE waveform to acquire data and stop scope for further
440     !measurement.  Measurement is NOT displayed on front panel.
450     !
460 Get_waveform:                                !
480    OUTPUT @Scope;":DIGITIZE CHAN1"          !macro to acquire data & stop
490    RETURN
500     !
510     !have scope to a frequency measurement and read results into
520     !controller.
530     !
540 Measure:                                     !
550    OUTPUT @Scope;":MEASURE:FREQUENCY?"      !FREQUENCY query
560    ENTER @Scope;Value                       !read from scope
570    PRINT "FREQUENCY = ";Value;"Hz"
580    OUTPUT @Scope;":MEAS:VPP?"               !Vpp query
590    ENTER @Scope;Value
600    PRINT "Vpp = ";Value;"V"
610    RETURN
620   !end of program
630 Done:                                         !end of program
640    DIALOG "INFORMATION","Program complete"
650    GESCAPE CRT,42                           !restore alpha window
660    STOP
670    END
```

Instrument BASIC for Windows Digitize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize and acquire the input data (in this case channel 1), then perform a Vpp and frequency measurement. The waveform is also stored on disk. The stored waveform is then retrieved and displayed on the computer screen. The program assumes that the PROBE COMP signal from the front panel of the instrument is connected to channel 1 through a 10:1 probe, and that the HP 82335A HP-IB interface module is being used.

```
10    !DIG            I-Basic program
20    !
30    !MAIN PROGRAM
40    !
50     REAL Preamble(1:10)              !array for preamble information
60    !
70     CLEAR SCREEN
80     GESCAPE CRT,40            !maximize the alpha window
90     GESCAPE CRT,45            !place the alpha window on top
100    PRINT "This example program will perform the following tasks:"
110    PRINT "              a.  initialize interface and scope"
120    PRINT "              b.  digitize and acquire data"
130    PRINT "              c.  save and retrieve data to PC file"
140    PRINT "              d.  draw signal on computer display"
150    PRINT
160    PRINT "This program assumes the system is configured as:"
170    PRINT "    HP-IB address = 7"
180    PRINT "    scope address = 7"
190    PRINT "    connect 10:1 probe from channel 1 to probe comp"
200    PRINT "                 connection on the front panel"
210    PRINT
220    PRINT "If the addresses are not correct, change the ASSIGN "
230    PRINT "statements in the Initialize function."
260    Msg$="Want to continue?"
270    DIALOG "QUESTION",Msg$,Button;SET("TITLE":"Digitize
Program","X":500,"Y":25)
280    IF Button THEN GOTO Done
290    GOSUB Initialize                  !initialize interface and scope
291    PRINT "Waveform data is being read from scope and stored to PC file"
300    GOSUB Get_waveform                !dig & acquire signal
310    GOSUB Save_waveform               !save waveform data to PC file
320    GOSUB Retrieve_wave               !get waveform data from PC file
```

```
321     CLEAR SCREEN
330     PRINT "waveform data and preamble saved and retrieved from PC file"
331     PRINT
332     PRINT "NOTE:  512 data points were read from scope and will be displayed"
333     PRINT "       but the scope display is only of 500 points.  Use the pan"
334     PRINT "       and zoom feature to see all 512 on scope display!"
341     Msg$="Want to continue?"
342     DIALOG "QUESTION",Msg$,Button;SET("TITLE":"Digitize Program")
343     IF Button THEN GOTO Done
345     GOSUB Graph                          !draw waveform
350 Done:                                    !end of program
360     DIALOG "INFORMATION","Program Complete";SET("X":0,"Y":400)
370     GESCAPE CRT,42              !restore alpha window to small size
380     GESCAPE CRT,31              !hide graphics window
390     STOP
400    !
410    !INITIALIZE INTERFACE AND SCOPE
420 Initialize:                             !
430     PRINT "INITIALIZE"
440     ASSIGN @Isc TO 7                    !Interface Select Code = 7
450     ASSIGN @Scope TO 707                !scope address
460     CLEAR @Isc                          !clear HP-IB interface
470     OUTPUT @Scope;"*RST;*CLS"           !set to default config;clear status reg
480     OUTPUT @Scope;":AUTOSCALE"          !AUTOSCALE
490     OUTPUT @Scope;":CHAN1:PROBE 10"     !set probe attenuation to 10
500     OUTPUT @Scope;":WAVEFORM:FORMAT COMP"  !get 8 bit data format
510     CLEAR SCREEN
520     RETURN
530    !
540    !complete scope configuration; DIGITIZE and acquire waveform data
550    !and preamble (voltage/timing) information into computer.
560    !
570 Get_waveform:                           !
580     OUTPUT @Scope;":DIGITIZE CHAN1"         !macro to acquire data & stop
590     OUTPUT @Scope;":WAVEFORM:DATA?"         !query scope for data
600     ENTER @Scope USING "#,1A,";One_char$    !strip off header & size
610     IF One_char$="#" THEN Found_pound
620     PRINT "BAD DATA"
630     STOP
640 Found_pound:     !read record length from header
650     ENTER @Scope USING "#,1D";Digits        !get length of record
660     ENTER @Scope USING "#,"&VAL$(Digits)&"D";Length
670     PRINT "reading ";Length;" bytes from scope"
680     ALLOCATE INTEGER Waveform(1:Length)
```

```
690     ENTER @Scope USING "#,B";Waveform(*)    !read waveform information
700     ENTER @Scope USING "-K,B";End$          !get last byte (line feed)
710     OUTPUT @Scope;":WAVEFORM:PREAMBLE?"     !query for preamble
720     ENTER @Scope;Preamble(*)                !read preamble information
770     RETURN
780    !
790    !plot waveform data to display
800    !
810 Graph:                                      !
820     GCLEAR                                  !initialize graphics
830     CLEAR SCREEN
840     GINIT
850     GRAPHICS ON
860     WINDOW 1,Preamble(3),0,255
870     FRAME
880     PEN 4
890     MOVE 0,0
900     FOR I=1 TO Preamble(3)                  !plot data points
910       MOVE I,Waveform(I)
920       DRAW I,Waveform(I)
930     NEXT I
940     RETURN
950 !
960 Save_waveform:                          !
970     ON ERROR GOTO 990
980     PURGE "wavesam"
990     OFF ERROR
1000    CREATE BDAT "WAVESAM",1,Length+1000 !length = bytes of waveform data
1010    ASSIGN @Path TO "WAVESAM"
1020    OUTPUT @Path;Waveform(*),Preamble(*)
1030    PRINT "waveform saved to file WAVESAM"
1040    RETURN
1050   !
1060   !retrieve waveform data and preamble information from disk
1070   !
1080 Retrieve_wave:                         !
1090    ASSIGN @Path TO "WAVESAM"
1100    ENTER @Path;Waveform(*),Preamble(*)
1110    PRINT "waveform retrieved"
1120    RETURN
1130    END
```

5

Common Commands

Introduction

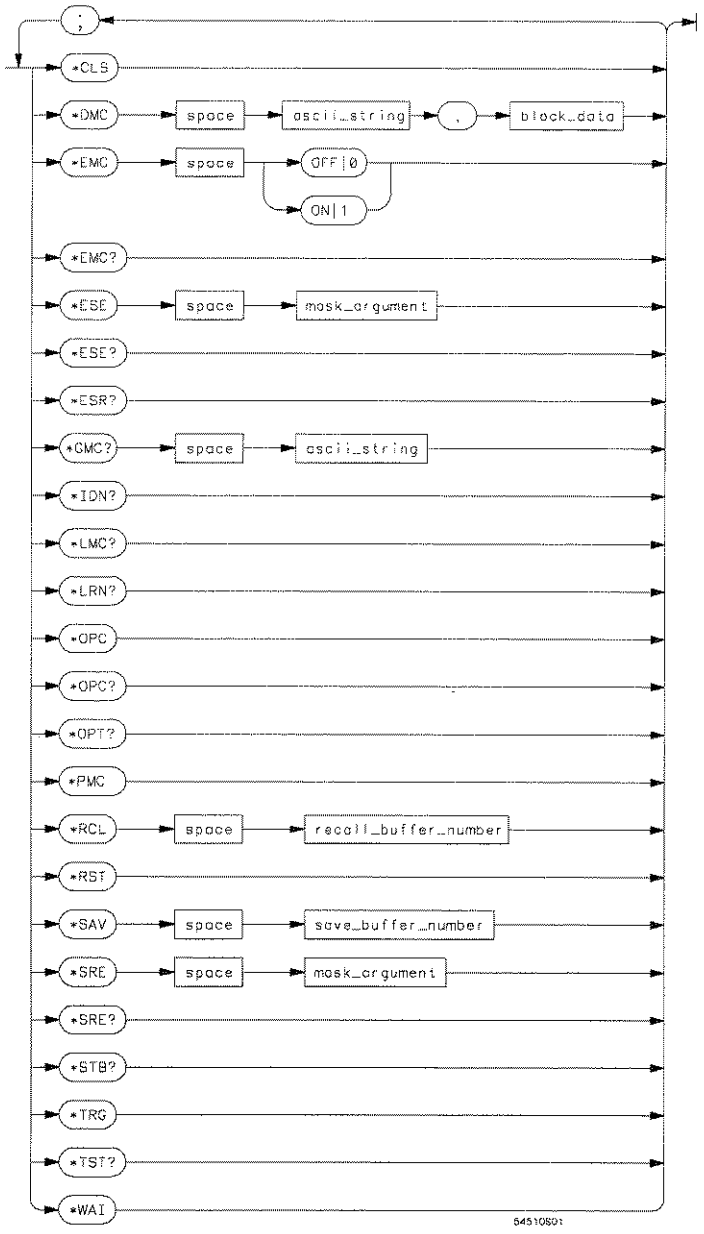
The common commands are defined by the IEEE 488.2 standard. These commands are common to all instruments that comply with the IEEE 488.2 standard. They control some of the basic instrument functions, such as instrument identification and reset, reading the learn (instrument setup) string, how status is read and cleared, and how commands and queries are received and processed by the instrument.

The following common commands are implemented in the oscilloscope:

*CLS (Clear Status)
*DMC (Define Macro)
*EMC (Enable Macro)
*ESE (Event Status Enable)
*ESR (Event Status Register)
*GMC (Get Macro Contents)
*IDN (Identification Number)
*LMC (Learn Macro)
*LRN (Learn)
*OPC (Operation Complete)
*OPT (Option)
*PMC (Purge Macro)
*RCL (Recall)
*RST (Reset)
*SAV (Save)
*SRE (Service Request Enable)
*STB (Status Byte)
*TRG (Trigger)
*TST (Test)
*WAI (Wait)

Figure 5-1 lists the syntax diagrams for the common commands.

Figure 5–1



Common Commands Syntax Diagram

Figure 5-1

mask_argument =	An integer, 0 through 255.
recall_buffer_number =	An integer, 0 through 9.
save_buffer_number =	An integer, 1 through 9.
ascii_string =	A quoted string.
block_data =	Definite block data in IEEE 488.2 # format.

Common Commands Syntax Diagram (continued)

Common Commands

Common commands can be received and processed by the oscilloscope whether they are sent over the remote interface as separate program messages or within other program messages. If an instrument subsystem has been selected and a common command is received by the instrument, the instrument remains in the selected subsystem. For example, if the program message ":ACQUIRE:TYPE AVERAGE; *CLS; COUNT 1024" is received by the instrument, the instrument sets the acquire type and count, then clears the status information without leaving the selected subsystem.

If some other type of command is received within the program message, you must reenter the original subsystem after the command. For example, the program message ":ACQUIRE:TYPE AVERAGE; :AUTOSCALE; :ACQUIRE:COUNT 1024" sets the acquire type, completes the autoscale, then sets the acquire count. In this example, :ACQuire must be sent again after the AUTOscale command in order to reenter the acquire subsystem and set the count.

Each of the status registers mentioned in this chapter has an enable (mask) register. By setting the bits in the enable register, you can select the status information you wish to use. For a complete discussion of how to read the status registers and how to use the status information available from this instrument refer to the "Status Reporting" chapter.

*CLS (Clear Status)

Command

```
*CLS
```

The *CLS (clear status) common command clears the status data structures, including the device-defined error queue.

If the *CLS command immediately follows a program message terminator, the output queue and the MAV (message available) bit are cleared.

Example

```
OUTPUT 707;"*CLS"
```

Refer to the "Status Reporting" chapter for a complete discussion of status.

*DMC (Define Macro)

Command

```
*DMC <ascii_string>,<block_data>
```

The *DMC (define macro) common command assigns a sequence of zero or more program message unit elements to a macro label.

`<ascii_string>` ::= a quoted ascii string

`<block_data>` ::= definite block data in IEEE 488.2 # format

Example

```
OUTPUT 707;"*DMC ""SETUPTB"",#222TIM:RANG 10NS;DELAY 0"
```

*EMC (Enable Macro)

Command

```
*EMC {{OFF | 0} | {ON | 1}}
```

The *EMC (enable macro) common command enables and disables expansion macros. Macro definitions are not affected by this command. *EMC can be used to turn off macro expansion in order to execute a device specific command with the same name as a macro.

Example

```
OUTPUT 707;"*EMC 1"
```

Query

```
*EMC?
```

Returned Format

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

Example

```
OUTPUT 707;"*DMC 'STORE',#228*EMC 0;STORE $1;WMEM4;*EMC 1"
OUTPUT 707;"STORE CHAN1"
OUTPUT 707;"*EMC?"
ENTER 707;Macro
PRINT Macro
```

*ESE (Event Status Enable)

Command

```
*ESE <mask_argument>
```

The *ESE command sets the bits in the Standard Event Status Enable Register. The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register. A one in the Standard Event Status Enable Register enables the corresponding bit in the Standard Event Status Register. A zero disables the bit. Refer to table 5-1 for information about the Standard Event Status Enable Register bits, bit weights, and what each bit masks.

`<mask_argument>` ::= 0 to 255 (integer - NR1 format)

Example

```
OUTPUT 707;"*ESE 64"
```

In this example, the *ESE 64 command enables URQ (user request) bit 6 of the Standard Event Status Enable Register. Therefore, when a front panel key is pressed, the ESB (event summary bit) in the Status Byte Register is also set.

Query `*ESE?`

The *ESE query returns the current contents of the register.

Returned Format `<mask_argument><NL>`

`<mask_argument>` ::= 0 to 255 (integer - NR1 format)

Example

```
OUTPUT 707;"*ESE?"
ENTER 707;Event
PRINT Event
```

Table 5-1 Standard Event Status Enable Register

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

Low - disables the ERS bit while High - enables the ERS bit

Refer to the "Status Reporting" chapter for a complete discussion of status.

*ESR? (Event Status Register)

Query

```
*ESR?
```

The *ESR query returns the contents of the Standard Event Status Register. When you read the Event Status Register, the value returned is the total bit weights of all of the bits that are high at the time you read the byte. Table 5-2 shows each bit in the Event Status Register and its bit weight.

> Reading the register clears the Standard Event Status Register.

Returned Format

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

`<status>` ::= 0 to 255 (integer - NR1 format)

Example

```
OUTPUT 707;"*ESR?"
ENTER 707;Event
PRINT Event
```

Table 5-2 **Standard Event Status Register**

Bit	Bit Weight	Bit Name	Condition
7	128	PON	1 = OFF to ON transition has occurred.
6	64	URQ	0 = no front-panel key has been pressed. 1 = a front-panel key has been pressed.
5	32	CME	0 = no command errors. 1 = a command error has been detected.
4	16	EXE	0 = no execution error. 1 = an execution error has been detected.
3	8	DDE	0 = no device dependent errors. 1 = a device dependent error has been detected.
2	4	QYE	0 = no query errors. 1 = a query error has been detected.
1	2	RQC	0 = Not used - always 0.
0	1	OPC	0 = operation is not complete. 1 = operation is complete.

0 = False = Low, 1 = True = High

*GMC? (Get Macro Contents)

Query

```
*GMC? <ascii_string>
```

The *GMC query retrieves the current definition of a macro from the oscilloscope.

`<ascii_string>` ::= a quoted string

Returned Format

```
<block_data><NL>
```

`<block_data>` ::= definite block data in # format

Example

```
DIM Def$[50]
OUTPUT 707;"*GMC? ""STORE"""
ENTER 707;Def$
PRINT Def$
```

*IDN? (Identification Number)

Query

```
*IDN?
```

The *IDN query identifies the instrument type, serial number, and software versions.

An *IDN query must be the last query in a message. Any queries after the *IDN query in a program message are ignored.

Returned Format

```
HEWLETT-PACKARD,545XXA,YYYYYYYYYY,ZZ.ZZ,ZZ.ZZ,ZZ.ZZ,ZZ.ZZ<NL>
```

`<XXA>` ::= model number is 20A (HP 54520A), 22A (HP 54522A), 40A (HP 54540A), or 42A (HP 54542A)

`<YYYYYYYYYY>` ::= the serial number of the instrument

`<ZZ.ZZ>` ::= the software revision of the software modules (Boot ROM, Flash ROM version of Boot ROM, System, Keyboard ROM). 00.00 = not installed

Example

```
DIM Id$[50]
OUTPUT 707;"*IDN?"
ENTER 707;Id$
PRINT Id$
```

*LMC? (Learn Macro)

Query

```
*LMC?
```

The *LMC query returns the currently defined macro labels.

Returned Format

```
<ascii_string><NL>
```

`<ascii_string>` ::= string list separated by commas

Example

```
DIM Label$[50]
OUTPUT 707;"*LMC?"
ENTER 707;Label$
PRINT Label$
```

*LRN? (Learn)

Query

```
*LRN?
```

The *LRN query returns a program message that contains the current state of the instrument.

This query performs the same function as the :SYSTem:SETup? query. It allows you to store an instrument setup in the controller. The stored setup can then be returned to the instrument at a later time using the :SYSTem:SETup command.

> The returned header for the *LRN query is :SYSTem:SETup. The :SYSTem:HEADer command does not affect this returned header.

Returned Format

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

<setup> ::= #800002048<learn_string><NL>

<learn_string> ::= 2048 data bytes in length

Example

```
DIM Lrn$[3000]
OUTPUT 707;"*LRN?"
ENTER 707 USING "-K";Lrn$
```

*OPC (Operation Complete)

Command

```
*OPC
```

The *OPC (operation complete) command sets the operation complete bit in the Standard Event Status Register when all pending device operations have finished.

Example

```
OUTPUT 707;"*OPC"
```

Query

```
*OPC?
```

The *OPC query places an ASCII "1" in the output queue when all pending device operations have finished.

Returned Format

```
1<NL>
```

Example

```
OUTPUT 707;":AUTOSCALE;*OPC?"
ENTER 707;Op$
```

*OPT? (Option)

Query

```
*OPT?
```

The *OPT query reports the options installed in the instrument. This query always returns a zero because the oscilloscope does not have any possible options to report.

Returned Format

```
0<NL>
```

Example

```
OUTPUT 707;"*OPT?"
ENTER 707;Value
PRINT Value
```

*PMC (Purge Macro)

Command

```
*PMC
```

The *PMC (purge macros) common command deletes all macros that have been previously defined using the *DMC common command, and removes all stored macro command sequences and labels from the oscilloscope's memory.

Example

```
OUTPUT 707;"*PMC"
```

This command is similiar to the SYSTem:PIMacro command.

*RCL (Recall)

Command

```
*RCL {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}
```

The *RCL command restores the state of the instrument from the specified save/recall register. An instrument setup must have been stored previously in the specified register. Registers 1 through 9 are general purpose and can be used with the *SAV command. Register 0 is special because it recalls the state that existed before the last AUToscale, RECall, ECL, or TTL operation.

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

Example

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

*RST (Reset)

Command

```
*RST
```

The *RST command places the instrument in a known state. Refer to table 5-3 for the reset conditions.

Example

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

Table 5-3 Reset Conditions

Horizontal Menu	
time/division	100 µs
delay	0.00 s
record length	512
reference	cntr
sample clock	auto
sample rate	real-time
sequential	off
Vertical Menu	
Channel 1	on
Channel 2	off
Channel 3 (HP 54540A/54542A only)	off
Channel 4 (HP 54540A/54542A only)	off
volts/division	500 mV
offset	0.00
coupling	dc
input resistance	1 MΩ
probe attenuation	1.000:1
Trigger Menu	
triggering	auto
mode	edge
source	Channel 1
level	0.0 V
slope	positive
noise reject	off
coupling	dc
holdoff	40 ns

Table 5-3 **Reset Conditions (continued)**

Display Menu	
mode	normal
persistence	single
# of screens	1
off/frame/axes/grid	axes
connect dots	off
Δt/ΔV Menu	
Markers	off
Waveform Math Menu	
f1	off
f2	off
f3	off
f4	off
display	off
chan/mem	Channel 1
operator	+
chan/mem	Channel 1
function sensitivity	1.00 V/div
function offset	0.0 V
Waveform Save Menu	
waveform/pixel/multiple/mask	waveform
nonvolatile	m1
display	off
source	Channel 1
protect	off
pixel display	off
multiple memory	off
Define Meas Menu	
measure/define/limit/compare	measure
continuous	on
statistics	off
static output	minimum/maximum/average
rms	ac
standard/user defined	standard
limit test	off
compare	off
Utility Menu (System Submenu)	
clicker	on
AC BNC	probe comp
probe comp freq	496 Hz
interpolation	on

*SAV (Save)

Command

```
*SAV {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}
```

The *SAV command stores the current state of the device in a save register. The data parameter is the number of the save register where the data will be saved. Registers 1 through 9 are valid for this command.

Example

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

*SRE (Service Request Enable)

Command

```
*SRE <mask_argument>
```

The *SRE command sets the bits in the Service Request Enable Register. The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A one in the Service Request Enable Register enables the corresponding bit in the Status Byte Register. A zero disables the bit. Table 5-4 lists the bits in the Service Request Enable Register and what they mask.

`<mask_argument>` ::= 0 to 255 (integer - NR1 format)

Example

```
OUTPUT 707;"*SRE 16"
```

This example enables a service request to be generated when a message is available in the output queue. When a message is available the MAV bit is high.

Query

*SRE?

The *SRE query returns the current value.

Returned Format

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

`<mask>` ::= sum of all bits that are set - 0 through 255 (integer - NR1 format)

Example

```
OUTPUT 707;"*SRE?"
ENTER 707;Value
PRINT Value
```

Table 5-4 **Service Request Enable Register**

Bit	Bit Weight	Enables
7	128	ARM - Armed
6	64	RQS - Request Service
5	32	ESB - Event Status Bit
4	16	MAV - Message Available
3	8	LTF - Limit Test Failure
2	4	MSG - Message
1	2	LCL - Local
0	1	TRG - Trigger

Low - disables the SRE bit High - enables the SRE bit

*STB? (Status Byte)

Query

```
*STB?
```

The *STB query returns the current value of the instrument's status byte. The MSS (Master Summary Status) bit is reported on bit 6 instead of the RQS (request service) bit. The MSS indicates whether or not the device has at least one reason for requesting service. Refer to table 5-5 for the meaning of the bits in the status byte.

> To read the instrument's status byte with RQS reported on bit 6, use the remote interface Serial Poll.

Returned Format

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

<value> ::= 0 through 255 (integer - NR1)

Example

```
OUTPUT 707;"*STB?"
ENTER 707;Value
PRINT Value
```

Table 5-5

Status Byte Register

Bit	Bit Weight	Bit Name	Condition
7	128	---	0 = not used.
6	64	RQS/MSS	0 = instrument has no reason for service. 1 = instrument is requesting service.
5	32	ESB	0 = no event status conditions have occurred. 1 = an enabled event status condition has occurred.
4	16	MAV	0 = no output messages are ready. 1 = an output message is ready.
3	8	LTF	0 = no limit test has failed. 1 = limit test has failed.
2	4	MSG	0 = no message has been displayed. 1 = message has been displayed.
1	2	LCL	0 = a remote-to-local transition has not occurred. 1 = a remote-to-local transition has occurred.
0	1	TRG	0 = no trigger has occurred. 1 = a trigger has occurred.

0 = False = Low, 1 = True = High

*TRG? (Trigger)

Command

```
*TRG
```

The *TRG command has the same effect as the Group Execute Trigger (GET). That effect is as if the RUN command had been sent.

Example

```
OUTPUT 707;"*TRG"
```

*TST? (Test)

Query

```
*TST?
```

The *TST query performs a self-test on the instrument. The result of the test is placed in the output queue.

> Disconnect all front-panel inputs before sending this command.

A zero indicates the test passed and a non-zero value indicates the test failed. If a test fails, refer to the troubleshooting section of the *HP 54520A, HP 54522A, HP 54540A, and HP 54542A Service Manual.*

Returned Format

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

<result> ::= 0 or non-zero value. 0 indicates the test passed.
non-zero indicates the test failed

Example

```
OUTPUT 707;"*TST?"
ENTER 707;Result
PRINT Result
```

*WAI (Wait)

Command

```
*WAI
```

The *WAI command has no function in the oscilloscope, but is parsed for compatibility with other instruments.

Example

```
OUTPUT 707;"*WAI"
```

6

Root Level Commands

Introduction

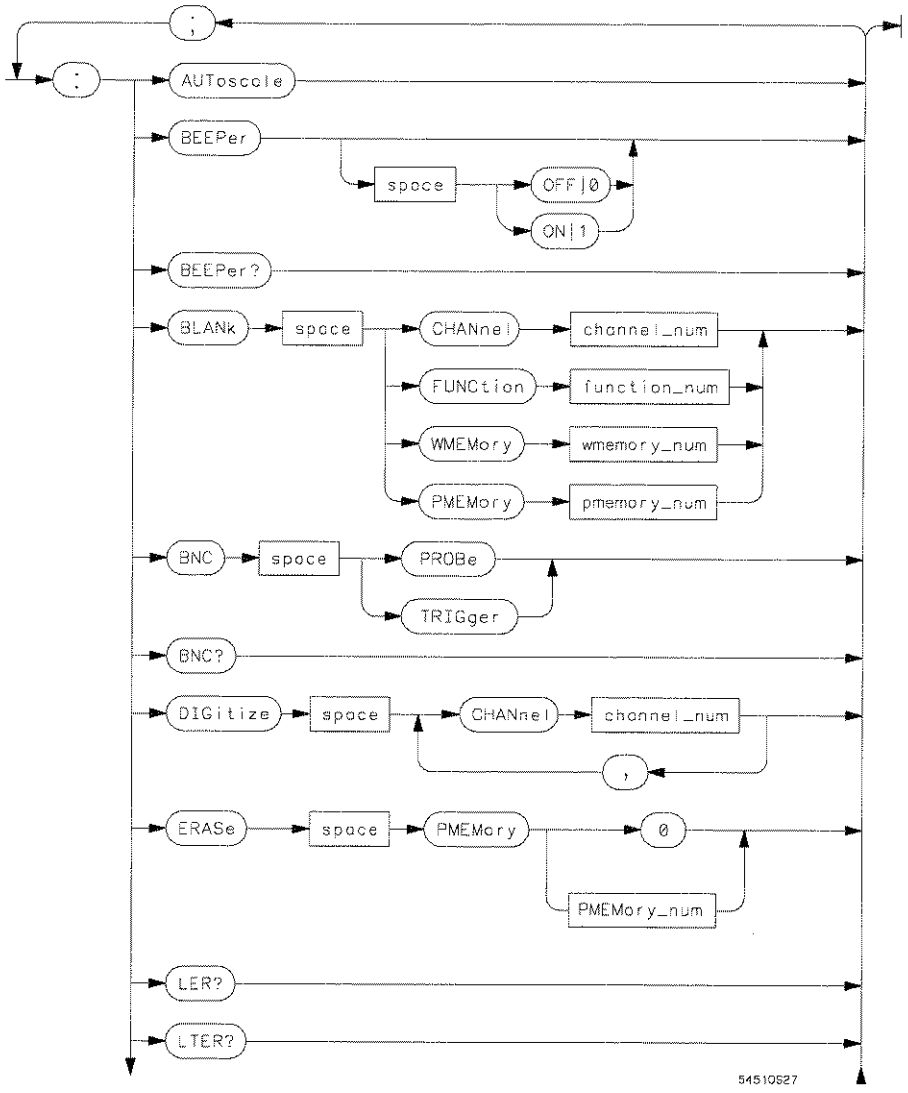
Root Level commands control many of the basic operations of the oscilloscope. These commands are always recognized by the parser if they are prefixed with a colon, regardless of current command tree position. After executing a root level command, the parser is positioned at the root of the command tree.

The following Root Level commands are implemented in the oscilloscope:

AUToscale	PLOT
BEEPer	PRINt
BLANk	POWerup
BNC	POWerup?
DIGitize	RUN
ERASe	SERial
LER	STATus
LTER	STOP
MENU	STORe
MERGe	TER
PCFRequency	VIEW

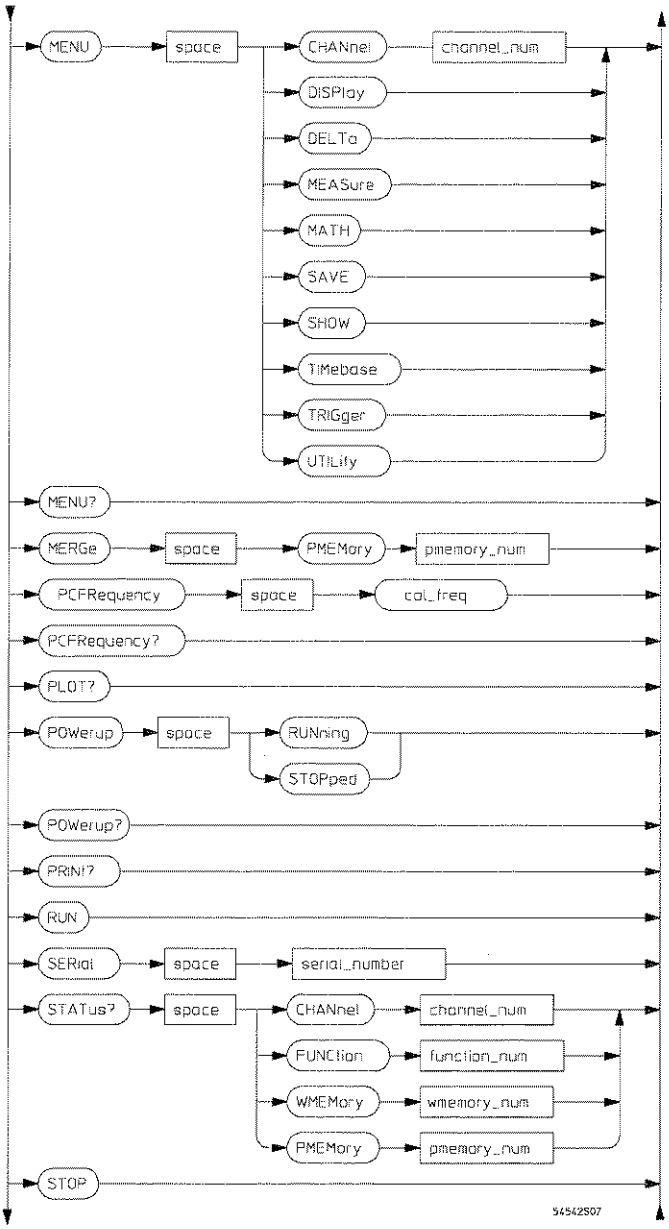
Figure 6-1 lists the syntax diagrams for the Root Level commands.

Figure 6–1



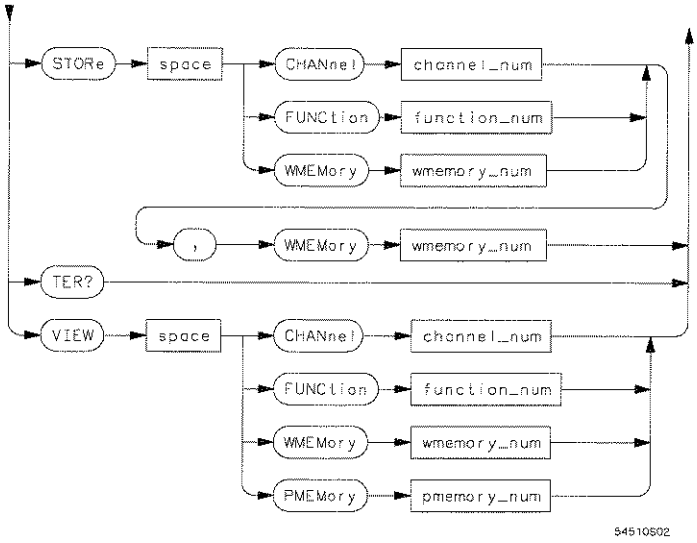
Root Level Commands Syntax Diagram

Figure 6–1



Root Level Commands Syntax Diagram (continued)

Figure 6–1



cal_freq =	frequency in Hz from 250 mHz to 32 kHz.
channel_num =	an integer, 1 or 2 (HP 54520A/54522A) or 1 through 4 (HP 54540A/54542A).
function_num =	an integer, 1 through 4.
wmemory_num =	an integer, 1 through 4.
pmemory_num =	an integer, 1 or 2.
serial_number =	a 10 character quoted ascii string.

Root Level Commands Syntax Diagram (continued)

AUToscale

Command

```
:AUToscale
```

The AUToscale command evaluates all input signals and sets the correct conditions to display the signals. When the AUToscale command is sent the following conditions are set:

- The vertical sensitivity.
- The vertical offset.
- The trigger to edge mode.
- The trigger level, holdoff, and slope.
- The sweep speed of the displayed channel.
- The time base reference to center.
- The time base delay to 0.0 s.
- Real-time acquisition mode to normal (peak detect to off).

In addition, the AUToscale command turns off the following items:

- Markers.
- All measurements.
- Functions.
- Memories.
- Connect the dots.
- Limit test.
- Waveform compare test.

If input signals are present on more than one vertical input, the sweep is triggered on the lowest channel that a signal is present on. For example, if a signal is not present on channel 1, then the oscilloscope look for a signal on channel 2. If no signals are found on any vertical input, the oscilloscope is returned to its former state.

Example

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

BEEPer

Command

```
:BEEPer [{{OFF | 0} | {ON | 1}}]
```

The BEEPer command sets the beeper mode, which controls the sound function of the instrument. The beeper can be set to on or off. If the BEEPer command is sent without an argument the beeper will be sounded without effecting the current mode of the instrument.

Example

```
OUTPUT 707;":BEEPER 1"
```

Query

```
:BEEPer?
```

The BEEPer query returns the current state of the beeper mode.

Returned Format

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

Example

```
OUTPUT 707;":BEEP?"
ENTER 707;Click
PRINT Click
```

BLANk

Command

```
:BLANk <display>
```

The BLANk command turns off (stops displaying) the specified channel, function, pixel memory, or waveform memory.

- To turn off a channel display, use the command :BLANk CHANnel<n>, where <n> is 1 or 2 for the HP 54520A and 54522A, or 1 through 4 for the HP 54540A and 54542A.
- To turn off a waveform memory display, use the command :BLANk WMEMory{1 | 2 | 3 | 4}.
- To turn off a pixel memory display, use the command :BLANk PMEMory{1 | 2}.
- To turn off a function, use the command :BLANk FUNCtion {1 | 2 | 3 | 4}.

Use the VIEW command to turn on (start displaying) an active channel, function, pixel memory, or waveform memory.

`<display>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4} | PMEMory{1 | 2 }}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

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

This command is the same as the CHANnel:DISPlay OFF, FUNCtion:DISPlay OFF, and WMEMory:DISPlay OFF commands.

BNC

Command

```
:BNC {PROBe | TRIGger}
```

The BNC command sets the output mode of the rear-panel BNC to PROBe or TRIGger. The PROBe mode outputs a square wave signal with variable frequency (frequency set using the PCFRequency command). The TRIGger mode outputs a negative going pulse when an internal trigger occurs.

Example

```
OUTPUT 707;":BNC PROBE"
```

Query

```
:BNC?
```

The BNC query returns the current mode for the rear-panel BNC.

Returned Format

```
[:BNC] {PROBe | TRIGger}<NL>
```

Example

```
DIM Mode$[50]
OUTPUT 707;":BNC?"
ENTER 707;Mode$
PRINT Mode$
```

DIGitize

Command

```
:DIGitize CHANnel<n>[,CHANnel<n>[,CHANnel<n>
[,CHANnel<n>]]]
```

The DIGitize command remotely initiates a data acquisition. Sending the command causes an acquisition to take place on the specified channels with the resulting data being placed in the channel buffer. Sending the DIGitize command also turns off any unused channels.

The sources for the :DIGitize command are channels 1 and 2 (HP 54520A/54522A), or channels 1 through 4 (HP 54540A/54542A).

When the digitize operation is complete the instrument is placed in the stopped mode. When the instrument is restarted, with a RUN command or the front panel RUN key, the digitized data stored in the channel buffers is overwritten. Therefore, ensure all operations that require the digitized data are completed before restarting the oscilloscope.

The ACQuire subsystem commands are used to set up conditions such as TYPE, number of POINts, and COUNt for the next DIGitize command. See the ACQuire subsystem for a description of these commands. To determine the actual number of points that are acquired and how the data is transferred, refer to the WAVeform subsystem commands.

You can improve the speed by turning the screen off with the command :DISPlay:SCReen OFF prior to sending the DIGitize command. In this case, nothing is plotted on the screen. Further optimization can be achieved by turning off connect the dots.

For more information on the DIGitize command refer to the section on the DIGitize command in the "Programming an Instrument" chapter.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":DIGITIZE CHANNEL1,CHANNEL2"
```

ERASe

Command

```
:ERASe {PMEMory{0 | <pmemory_num>}}
```

The ERASe command erases a specified pixel memory.

Erasing pixel memory 0 is a special case, which is the same as pressing the CLEAR DISPLAY key on the front panel. If the scope is running and being triggered and ERASe PMEMory0 is executed, the instrument momentarily stops acquiring data, clears the screen, then restarts the data acquisition.

Erasing pixel memory 1 or 2 clears the specified pixel memory and anything on the display from that pixel memory.

<pmemory_num> = 1 or 2 (integer - NR1 format)

> Once you erase pixel memory 1 or 2, there is no way to retrieve the original information.

Example

```
OUTPUT 707;":ERASE PMEMORY1"
```

LER? (Local Event Register)

Query

```
:LER?
```

The LER query reads the LCL (Local) Event Register. After the LCL Event Register is read, it is cleared. A one indicates a remote to local transition has taken place due to the front-panel LOCAL key being pressed. A zero indicates a remote to local transition has not taken place.

Once this bit is set it can only be cleared by reading the Event Register or sending a *CLS command.

A Service Request (SRQ) can only be generated when the bit transitions from 0 to 1. Therefore, the bit must be cleared each time you want a new Service Request to be generated.

Returned Format

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

Example

```
OUTPUT 707;":LER?"
ENTER 707;Event
PRINT Event
```

LTER? (Limit Test Event Register)

Query

```
:LTER?
```

The LTER query reads the Limit Test Event Register. The Limit Test Event Register contains the Limit Test Fail bit. This bit is set when either the limit test or waveform compare test is active, and the test has failed. After the Limit Test Event Register is read, it is cleared.

A Service Request (SRQ) can only be generated when the bit transitions from 0 to 1; therefore, the bit must be cleared each time you would like a new Service Request to be generated.

Returned Format

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

Example

```
OUTPUT 707;":LTER?"
ENTER 707;Lmt
PRINT Lmt
```

MENU

Command

```
:MENU {CHANnel<n> | TIMebase | TRIGger | DISK |
DISPlay | DELTa | MATH | SAVE | MEASure | UTILity
| SHOW}
```

The MENU command selects one of the menus on the front panel.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MENU DISPLAY"
```

Query

```
:MENU?
```

The MENU query returns the name of the currently displayed menu.

Returned Format

```
[:MENU] {CHANnel<n> | TIMebase | TRIGger | DISK | DISPlay |
DELTa | MATH | SAVE | MEASure | UTILity | SHOW}
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Name$[50]
OUTPUT 707;":MENU?"
ENTER 707;Name$
PRINT Name$
```

All menu commands are identical to corresponding front panel functions, except those listed in the following table:

Command	Front Panel Function
CHANnel	Vertical
TIMebase	Horizontal
DELTa	Markers
SAVE	WX Save
MEASure	Define Meas
MATH	Math/FFT

MERGe

Command

```
:MERGe PMEMory<pmemory_num>
```

The MERGe command stores the contents of the active display into the specified pixel memory. The pixel memories are PMEMory1 or PMEMory2. The function of this command is similar to the function of the "add to memory" key in the pixel menu of the Waveform Save menu.

`<pmemory_num>` ::= 1 or 2 (integer - NR1 format)

Example

```
OUTPUT 707;":MERGE PMEMORY2"
```

PCFRequency

Command

```
:PCFRequency <cal_freq>
```

The PCFRequency command is used to set the output frequency of the front panel probe compensation jack from 250 mHz to 32 kHz. Value entered is rounded to the next higher acceptable setting.

`<cal_freq>` ::= probe compensation signal frequency in Hz (exponential - NR3 format)

Example

```
OUTPUT 707;":PCFR 1024"
```

Query

```
:PCFRequency?
```

The PCFRequency query returns the frequency of the signal at the front panel probe compensation jack.

Returned Format

```
[:PCFRequency] <cal_freq>
```

`<cal_freq>` ::= probe compensation signal frequency in Hz (exponential - NR3 format)

Example

```
OUTPUT 707;":PCFR?"
ENTER 707;FREQ
PRINT FREQ
```

PLOT

Query

:PLOT?

The PLOT query outputs a copy of the display as soon as the oscilloscope is addressed to talk. The portion of the waveform to be copied must be placed on the display before sending the PLOT query. The plotter output includes the displayed waveforms, the graticule, time and voltage markers, trigger setup, and measurement results.

Example

```
10 CLEAR 707    ! Initialize instrument interface
15 HARDcopy:mode plot
20 ON INTR 7,5 GOTO 160      ! Exit printing routine after SRQ
30 ENABLE INTR 7;2     ! Enable SRQ on bus #7
40 OUTPUT 707;"*CLS"   ! Clear status data structures
50 OUTPUT 707;"*ESE 1"! Enable OPC
60 OUTPUT 707;"*SRE 32"      ! Enable Event Status Register
Interrupt
70 OUTPUT 707;":HARDCOPY:PAGE AUTOMATIC"
80 OUTPUT 707;":HARDCOPY:LENGTH 12"
90 OUTPUT 707;":PLOT?;*OPC"  ! Set OPC when plot is complete
100 SEND 7;UNT UNL     ! Clear bus
110 SEND 7;TALK 7      ! Scope to talk mode
120 SEND 7;LISTEN 1    ! Printer to listen mode
130 SEND 7;DATA ! Lower ATN line @ controller
140 GOTO 140    ! Loop until printing is
150 ! complete, then generate interrupt
160 A=SPOLL(707)       ! Clear service request
170 OUTPUT 707;":SYST:DSP ""PRINT IS COMPLETE"""
180 WAIT 6
190 OUTPUT 707;":SYST:DSP """"
200 END
```

POWerup

Command

```
:POWerup {STOPped | RUNning}
```

The :POWerup STOPped command sets up the oscilloscope so that when it is powered up a flag is set and the oscilloscope is in a stopped state with the compare/limiters turned off. This protects any data in memories WMem1-4. If the RUNning parameter is specified, or no parameter is specified, the oscilloscope starts up with default parameters with the oscilloscope running.

Example

```
OUTPUT 707;":POWerup STOPped"
```

Query

```
:POWerup?
```

The :POWerup query returns the current powerup mode.

Returned Format

```
[:POWerup] {STOPped | RUNning}
```

Example

```
DIM Mode$[50]
OUTPUT 707;":POWerup?"
ENTER 707;Mode$
PRINT Mode?
```

PRINt?

Query

```
:PRINt?
```

The PRINt query outputs a copy of the display as soon as the oscilloscope is addressed to talk. The portion of the waveform to be copied must be placed on the display before sending the PRINt query. The printer output includes the displayed waveforms, the graticule, time and voltage markers, trigger setup, and measurement results.

Example

```
10 CLEAR 707    ! Initialize instrument interface
15 HARDcopy:mode print
20 ON INTR 7,5 GOTO 160      ! Exit printing routine after SRQ
30 ENABLE INTR 7;2     ! Enable SRQ on bus #7
40 OUTPUT 707;"*CLS"   ! Clear status data structures
50 OUTPUT 707;"*ESE 1"! Enable OPC
60 OUTPUT 707;"*SRE 32"      ! Enable Event Status Register
Interrupt
70 OUTPUT 707;":HARDCOPY:PAGE AUTOMATIC"
80 OUTPUT 707;":HARDCOPY:LENGTH 12"
90 OUTPUT 707;":PRINT?;*OPC" ! Set OPC when print is complete
100 SEND 7;UNT UNL     ! Clear bus
110 SEND 7;TALK 7      ! Scope to talk mode
120 SEND 7;LISTEN 1    ! Printer to listen mode
130 SEND 7;DATA ! Lower ATN line @ controller
140 GOTO 140    ! Loop until printing is
150 ! complete, then generate interrupt
160 A=SPOLL(707)       ! Clear service request
170 OUTPUT 707;":SYST:DSP ""PRINT IS COMPLETE"""
180 WAIT 6
190 OUTPUT 707;":SYST:DSP """"
200 END
```

RUN

Command

```
:RUN
```

The RUN command acquires data for the active waveform display. The data is acquired as defined by the time base mode.

If the time base mode is in SINGLE, the RUN command enables the trigger once and displays the acquired data on the screen. This also occurs when the front panel SINGLE key is pressed while the instrument is STOPPED.

If the time base mode is set to AUTO or TRIGgered, the RUN command enables the trigger repeatedly and displays the data it acquires continuously on the screen. This is the same thing that happens when the front panel RUN key is pressed. For a description of the various modes, see the :TIMebase:MODE command in the "Timebase Subsystem" chapter.

Example

```
OUTPUT 707;":RUN"
```

SERial (Serial Number)

Command

```
:SERial <serial_number>
```

The SERial command allows you to enter a serial number in the instrument. The serial number is placed in protected non-volatile RAM, so the protection switch on the rear panel must be in the unprotected position to write a new serial number to the instrument.

A serial number corresponding to the serial number of the board in the oscilloscope is loaded at the factory. Do not use this command unless you need to serialize the instrument for a different application.

The serial number is part of the string returned for the *IDN query.

`<serial_number>` ::= 10 character serial number within quotes

Example

```
OUTPUT 707;":SERIAL ""1234A56789"""
```

STATus?

Query

```
:STATus? <display>
```

The STATus query indicates whether a channel, function, wmemory, or pmemory is ON or OFF. A one indicates ON and a zero indicates OFF.

`<display>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4} | PMEMory{1 | 2}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Returned Format

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

Example

```
OUTPUT 707;":STATUS? CHANNEL1"
ENTER 707;Status
PRINT Status
```

This command is identical to the CHANnel:DISPlay command.

STOP

Command

```
:STOP
```

The STOP command stops the data acquisition.
The RUN command must be executed to restart the acquisition.

Example

```
OUTPUT 707;":STOP"
```

STORe

Command

```
:STORe <source>,<destination>
```

The STORe command moves a stored waveform, channel, or function to a waveform memory. This command has two parameters:

The first parameter is the source of the waveform, which can be specified as any channel, function, or waveform memory.

The second parameter is the destination of the waveform, which can only be waveform memory 1 through 4.

An error is generated if the destination waveform memory write protect is set to ON.

`<source>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<destination>` ::= {WMEMory {1 | 2 | 3 | 4}}

Example

```
OUTPUT 707;":STORE CHANNEL2,WMEMORY4"
```

TER? (Trigger Event Register)

Query

```
:TER?
```

The TER query reads the Trigger Event Register. After the Trigger Event Register is read, it is cleared. A one indicates a trigger has occurred. A zero indicates a trigger has not occurred.

If a trigger event is not found and the sweep is auto-triggering, the Trigger Event Register bit is not set.

A Service Request (SRQ) can only be generated when the Trigger Event Register bit transitions from 0 to 1; therefore, the bit must be cleared each time you would like a new Service Request to be generated.

Returned Format

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

Example

```
OUTPUT 707;":TER?"
ENTER 707;Trg_event
PRINT Trg_event
```

VIEW

Command

```
:VIEW <display>
```

The VIEW command turns on (starts displaying) an active channel, function, pixel memory, or waveform memory.

To display a channel use the command :VIEW CHANnel<n>. To display a waveform memory, use the command :VIEW WMEMory{1 | 2 | 3 | 4}. To display a pixel memory, use the command :VIEW PMEMory{1 | 2}. To display a function use the command :VIEW FUNCtion{1 | 2 | 3 | 4}.

Use the BLANk command to turn off (stop displaying) a specified channel, function, pixel memory, or waveform memory.

<display> ::= CHANnel<n>| FUNCtion{1 | 2 | 3 | 4} | PMEMory{1 | 2} | WMEMory{1 | 2 | 3 | 4}}

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":VIEW CHANNEL1"
```

This command is the same as the CHANnel:DISPlay ON, FUNCtion:DISPlay ON, and WMEMory:DISPlay ON commands.

7

SYSTem Subsystem

Introduction

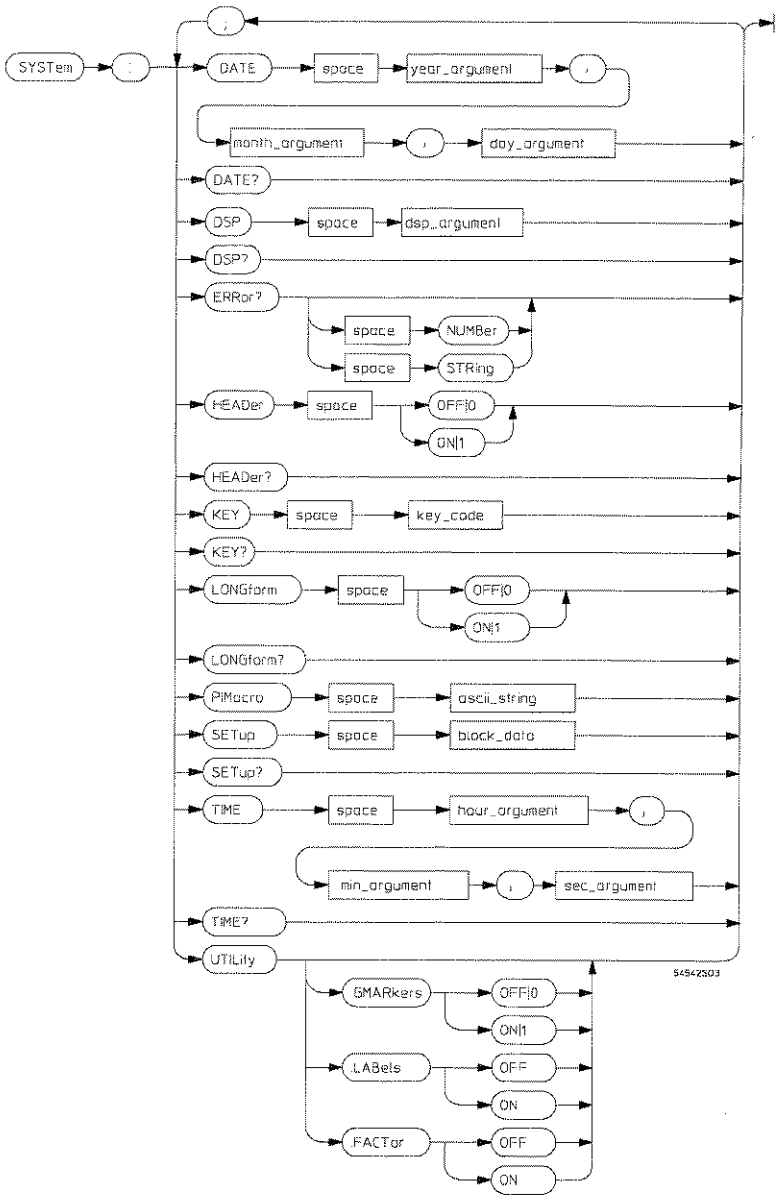
System commands control the way in which query responses are formatted, simulate front panel key presses, allows setting of the real-time clock, and enable reading and writing to the advisory line of the instrument.

The System subsystem contains the following commands:

DATE
DSP
ERRor
HEADer
KEY
LONGform
PIMacro
SETup
TIME
UTILity

Figure 7-1 lists the syntax diagrams for the System subsystem commands.

Figure 7–1



System Subsystem Commands Syntax Diagram

Figure 7-1

dsp_argument =	any quoted string.
key_code =	an integer, 1 through 63.
ascii_string =	any quoted string.
block_data =	block data in IEEE 488.2 # format.
year_argument =	an integer, 1990 through 2059.
month_argument =	an integer, 1 through 12.
day_argument =	an integer, 1 through 31.
hour_argument =	an integer, 0 through 23.
minute_argument =	an integer, 0 through 59.
second_argument =	an integer, 0 through 59.

System Subsystem Commands Syntax Diagram (continued)

DATE

Command

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

The :SYSTem:DATE command sets the oscilloscope's real-time clock. This is the date used when data is stored in memory, and on hard copy outputs. The real-time clock is maintained even when the oscilloscope is off or removed from line power.

`<year>` ::= 1990 to 2059 (integer - NR1 format)

`<month>` ::= 1 to 12 (integer - NR1 format)

`<day>` ::= 1 to 31 (integer - NR1 format)

Example

```
OUTPUT 707;":SYSTEM:DATE 1992,2,29"
```

Query

```
:SYSTem:DATE?
```

The SYSTem:DATE query returns the current date.

Returned Format

```
[:SYSTem:DATE] "DDMMMYYYY"<NL>
```

`<YYYY>` ::= 1990 through 2059 (integer - NR1 format)

`<MMM>` ::= three digit alphabetic month

`<DD>` ::= 1 through 31 (integer - NR1 format)

Example

```
DIM Date$[20]
OUTPUT 707;":SYST:DATE?"
ENTER 707;Date$
PRINT Date$
```

DSP

Command

```
:SYSTem:DSP <ascii_string>
```

The :SYSTem:DSP command writes a quoted string, excluding quotes, to the advisory line (line 1) of the screen.

`<ascii_string>` ::= any quoted string

Example

```
OUTPUT 707;":SYSTEM:DSP ""This is a message"""
```

Query

```
:SYSTem:DSP?
```

The DSP query returns the last string written to the advisory line. This may be a string written with a DSP command or an internally generated advisory.

The string is actually read from the message queue. The message queue is cleared when it is read. Therefore, the displayed message can only be read once over the bus.

Returned Format

```
[:SYSTem:DSP] <ascii_string><NL>
```

`<ascii_string>` ::= string response data containing the last information written on the advisory line.

Example

```
DIM Display$[100]
OUTPUT 707;":SYST:DSP?"
ENTER 707;Display$
PRINT Display$
```

ERRor?

Query

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

The :SYSTem:ERRor query outputs the next error number in the error queue over the remote interface. This instrument has an error queue that is 30 errors deep and operates on a first-in, first-out basis. Repeatedly sending the query :SYSTem:ERRor? returns the error numbers in the order that they occurred until the queue is empty. Any further queries then return zeros until another error occurs.

When the NUMBer parameter is used in the query, only the numeric error code is output. When the STRing parameter is used, the error number is output followed by a comma and a quoted string. If no parameter is specified, then only the numeric error code is output. Not specifying a parameter is the same as specifying NUMBer.

See table 7-1 for the error numbers and descriptions.

Returned Format

```
[:SYSTem:ERRor] <error>[,<ascii_string>]<NL>
```

`<error>` ::= an integer error code.

`<ascii_string>` ::= an alpha string specifying the error condition.

Example

```
DIM Emsg$[50]
OUTPUT 707;":SYSTEM:ERROR? STRING"
ENTER 707;Emsg$
PRINT Emsg$
```

Table 7-1 **Error Messages**

Error Number	Description
11	Questionable horizontal scaling
12	Edges required not found
13	Not a 545XXA command
70	RAM write protected
-100	Command error (unknown command)
-101	Invalid character
-102	Syntax error
-103	Invalid separator
-104	Data type error
-105	GET not allowed
-108	Parameter not allowed
-109	Missing parameter
-112	Program mnemonic too long
-113	Undefined header
-121	Invalid character in number
-123	Numeric overflow
-124	Too many digits
-128	Numeric data not allowed
-130	Suffix error
-131	Invalid suffix
-138	Suffix not allowed
-140	Character data error
-141	Invalid character data
-144	Character data too long
-148	Character data not allowed
-150	String data error
-151	Invalid string data
-158	String data not allowed
-160	Block data error
-161	Invalid block data
-168	Block data not allowed
-170	Expression error
-171	Invalid expression
-178	Expression data not allowed
-180	Macro error
-181	Invalid outside macro define
-183	Invalid inside macro define
-200	Execution error
-211	Trigger ignored
-221	Settings conflict
-222	Data out of range
-223	Too much data
-250	Mass storage error

Table 7-1. Error Messages (continued)

Error Number	Description
-251	Missing mass storage
-252	Missing media
-253	Corrupt media
-254	Media full
-255	Directory full
-256	File name not found
-257	File name error
-258	Media protected
-270	Macro error
-272	Macro execution error
-273	Illegal macro label
-276	Macro recursion error
-277	Macro redefined not allowed
-310	System error
-350	Too many errors
-400	Query error
-410	Query INTERRUPTED
-420	Query UNTERMINATED
-430	Query DEADLOCKED
-440	Query UNTERMINATED after indefinite response

HEADer

Command

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

The :SYSTem:HEADer command turns the command headers for query responses on and off. When the HEADer is set to ON, query responses include the command header.

Example

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

Query

```
:SYSTem:HEADer?
```

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

Returned Format

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

Example

```
OUTPUT 707;":SYST:HEAD?"
ENTER 707;Hdr
PRINT Hdr
```

The following example shows the response to the query :CHANnel1:RANGe? with the headers on and off.

With headers set to ON; long form ON:
:CHANNEL1:RANGE 6.40000E-01

With headers set to ON; long form OFF:
:CHAN1:RANG 6.40000E-01

With headers set to OFF:
6.40000E-01

Headers should be turned off when returning values to numeric variables. Otherwise, the controller may misinterpret the headers as part of returned data.

KEY

Command

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

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

Refer to table 7-2 for key codes.

<key_code> ::= 1 to 63 (integer - NR1 format)

Example

```
OUTPUT 707;":SYSTEM:KEY 2"
```

Query

```
:SYSTem:KEY?
```

The KEY query returns the key code for the last key pressed from the front panel or returns the last simulated key press over the remote interface. Key codes range from 1 to 63. Zero represents no key and is returned after power up.

Returned Format:

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

<key_code> ::= 0 through 63 (integer - NR1 format)

Example

```
OUTPUT 707;":SYST:KEY?
"ENTER 707;Input
PRINT Input
```

Table 7-2 **Oscilloscope Front-Panel Key Codes**

Key	Key Code	Key	Key Code
Menu - HORIZ SETUP	1	2	28
Menu - VERTICAL 1	2	3	29
Menu -TRIGGER SETUP	3	4	30
Menu - DISPLAY	4	5	31
Menu - MARKER	5	6	32
Menu - MATH/FFT	6	7	33
Menu - WFORM SAVE	7	8	34
Menu - DEFINE MEAS	8	9	35
Menu - UTILITY	9	RUN	36
Softkey 1 (top key)	10	STOP/SINGLE	37
Softkey 2	11	CLEAR DISPLAY	38
Softkey 3	12	UNUSED	39
Softkey 4	13	PRINT	40
Softkey 5	14	AUTO-SCALE	41
Softkey 6	15	RECALL SETUP	42
Softkey 7 (bottom key)	16	SAVE SETUP	43
FINE	17	SHOW	44
ENTER	18	EX (exponential)	45
m (milli)	19	p (pico)	46
μ (micro)	20	UNUSED	47 - 49
n (nano)	21	Menu - VERTICAL 2	50
CLEAR	22	UNUSED	51 - 52
SHIFT (blue key)	23	Menu - VERTICAL 3	53
"+/-" (plus/minus)	24	UNUSED	54 - 55
"." (decimal point)	25	Menu - VERTICAL 4	56
0	26	UNUSED	57 - 64
1	27		

LONGform

Command

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

The :SYSTem:LONGform command sets the long form variable for formatting query responses. If the LONGform command is set to OFF, command headers and alpha arguments are sent from the oscilloscope in the short form. If the LONGform command is set to ON, the whole word is output. This command does not affect the input data messages to the oscilloscope. Headers and arguments may be sent to the oscilloscope in either the long form or short form, regardless of how the LONGform command is set. For more information, refer to the HEADer command in this chapter.

Example

```
OUTPUT 707;":SYSTEM:LONGFORM ON"
```

Query

```
:SYSTem:LONGform?
```

The LONGform query returns the state of the LONGFORM command.

Returned Format

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

Example

```
OUTPUT 707;":SYST:LONG?"
ENTER 707;Long
PRINT Long
```

If SYSTem:HEADer are turned OFF, :LONGform has no effect.

PIMacro

Command

```
:SYSTem:PIMacro <ascii_string>
```

The :SYSTem:PIMacro command purges the macro described by the string name.

`<ascii_string>` ::= name of the defined macro

Example

```
OUTPUT 707;":SYSTEM:PIMacro ""STORE"""
```

Macro related commands are *DMC, *GMC?, *LMC?, *PMC, and *EMC.

SETup

Command `:SYSTem:SETup <block_data>`

The :SYSTem:SETup command sets the oscilloscope as defined by the data in the setup (learn) string sent from the controller. The setup string contains 2048 bytes of setup data (not including header or "#80002048.")

The setup string does not change the remote interface mode, remote interface address, fine key setting, waveform format, or waveform source.

`<block_data>` ::= #800002048<setup_string>

`<setup_string>` ::= block of binary data bytes

Query `:SYSTem:SETup?`

The SETup query operates the same as the *LRN? query. It outputs the current oscilloscope setup in the form of a learn string to the controller.

The setup (learn) string is sent and received as a binary block of data. The format for the data transmission is the # format defined in the IEEE 488.2 specification.

Returned Format `[:SYSTem:SETup] <block_data><NL>`

`<block_data>` ::= #800002048<setup_string>

`<setup_string>` ::= block of binary data bytes

Example:0

```
10 DIM Set$[3000]
20 !Setup the instrument as desired
30 OUTPUT 707;":SYST:HEAD OFF"
40 OUTPUT 707;":SYSTEM:SETUP?"
50 !Transfer the instrument setup to controller
60 ENTER 707 USING "-K";Set$  !Store the setup
70 PAUSE
80 OUTPUT 707 USING "#,K";":SYST:SETUP ";Set$
90 !Returns the instrument to the first setup
100 END
```

The query causes the learn string to be sent to the controller and the command causes the learn string to be returned to the oscilloscope.

TIME

Command

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

The :SYSTem:TIME command sets the oscilloscope's real-time clock in 24-hour format. This is the time used when data is stored in memory, and on hard copy outputs. The real-time clock is maintained even when the oscilloscope is off or removed from line power.

`<hour>` ::= 0 to 23 (integer - NR1 format)

`<minute>` ::= 0 to 59 (integer - NR1 format)

`<second>` ::= 0 to 59 (integer - NR1 format)

Example

```
OUTPUT 707;":SYSTEM:TIME 12,0,0"
```

Query

```
:SYSTem:TIME?
```

The SYSTem:TIME query returns the current time.

Returned Format

```
[:SYSTem:TIME] "HH:MM:SS"<NL>
```

`<HH>` ::=0 through 23 (integer - NR1 format)

`<MM>` ::=0 through 59 (integer - NR1 format)

`<SS>` ::=0 through 59 (integer - NR1 format)

Example

```
DIM Time$[20]
OUTPUT 707;":SYST:TIME?"
ENTER 707;Time$
PRINT Time$
```

UTILity

Command

```
:SYSTem:UTILity:GMARkers {ON | OFF}
:SYSTem:UTILity:LABels {ON | OFF}
:SYSTem:UTILity:FACTor {ON | OFF}
```

The :SYSTem:UTILity:GMARkers command turns the ground markers on or off. The :SYSTem:UTILity:LABels command turns the channel labels on or off. The :SYSTem:UTILity:FACTor command turns the channel factors on or off.

Example

```
OUTPUT 707;":SYSTEM:UTILity:GMARkers:ON"
```

Query

```
:SYSTem:UTILity:GMARkers?
:SYSTem:UTILity:LABels?
:SYSTem:UTILity:FACTor?
```

The SYSTem:UTILity queries return the setting for the specified function.

Returned Format

```
[:SYSTem:UTILity:GMARkers] {ON | OFF}<NL>
```

Example

```
OUTPUT 707;":SYST:UTIL:GMAR?"
ENTER 707;GMA$
PRINT GMA$
```

8

ACQuire Subsystem

Introduction

The ACQuire subsystem commands set up conditions for executing a DIGitize root level command to acquire waveform data.

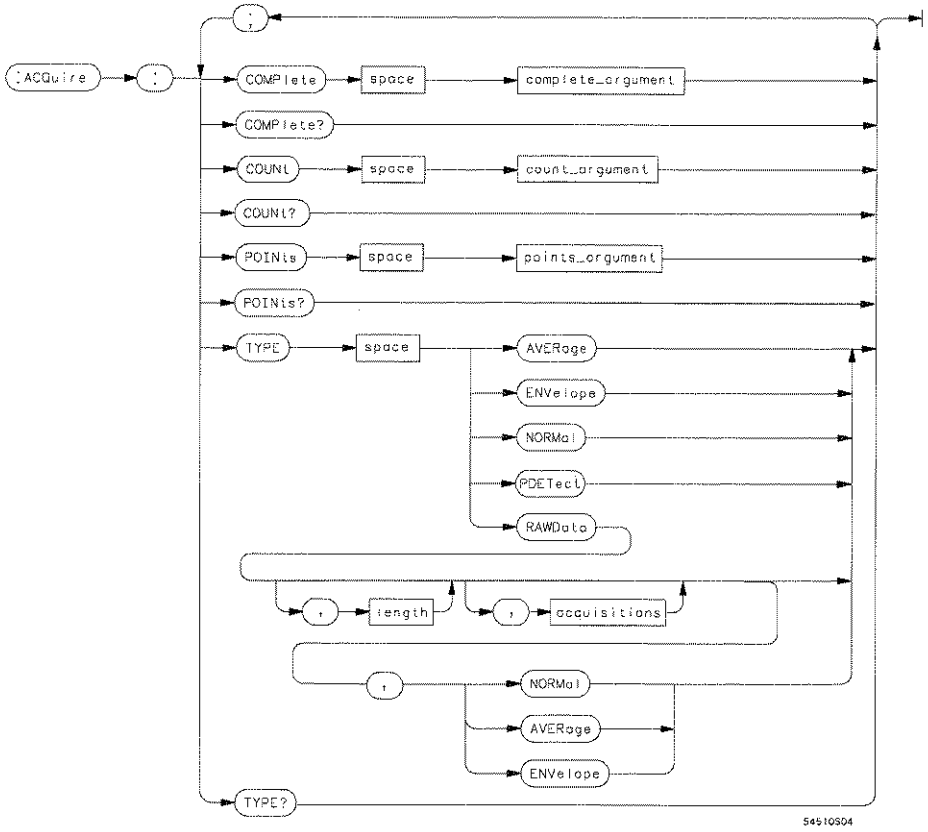
The Acquire subsystem selects the type of data, the number of averages, and the number of data points.

This subsystem contains the following commands:

COMPlete
COUNt
POINts
TYPE

Figure 8-1 lists the syntax diagrams for the Acquire subsystem commands.

Figure 8–1



acquisitions =	dependent on the length of the acquisitions and the buffer size.
complete_argument =	an integer, 0 through 100.
count_argument =	an integer, 1 to 2048.
length =	an integer, 4 to 32768.
points_argument =	500 in the repetitive mode. 512, 1024, 2048, 4096, 8192, 16384, 32768 in real-time mode.

Acquire Subsystem Commands Syntax Diagram

Acquire Type and Count

The ACQuire subsystem is the only remote interface control for two display parameters: Display Mode and Number of Averages. There is a coupling between the front panel Display menu and the ACQuire subsystem parameters. This means that when the remote interface parameters for ACQuire TYPE or COUNt are changed, the front panel changes. Also, when the front-panel parameters are changed, the remote interface parameters change.

(Normal) Persistence Mode

The :ACQuire:TYPE NORMal command sets the oscilloscope to the variable persistence mode when the instrument is operating in the repetitive time base mode. You can activate the same mode from the front panel by selecting the Display menu, then setting the display mode to Normal. The persistence time is set over the remote interface in the DISPlay subsystem using the :DISPlay:PERSistence command.

The :ACQuire:COUNt can be set in the variable persistence mode, but has no impact on the current display mode or Remote Interface acquisition. The :ACQuire:COUNt query always returns a 1 when the acquisition type is set to NORMal.

Averaging Mode

The :ACQuire:TYPE AVERage command sets the oscilloscope to the Averaging mode. The averaging mode is available only when the :TIMebase:SAMPle is set to REPetitive.

COUNt can be set in the AVERage mode by sending the :ACQuire:COUNt command followed by the number of averages. In this mode, the value is rounded to the nearest power of two. The COUNt value determines the number of averages that must be acquired.

Envelope Mode

The :ACQuire:TYPE ENVelope command sets the oscilloscope to the Envelope mode. The envelope mode is only available when the :TIMebase:SAMPle is set to REPetitive. In envelope mode, the ACQuire:COUNt command specifies how many times each time bucket is hit.

Rawdata Mode

The :ACQuire:TYPE RAWData command is a special command that allows you to get unfiltered, 16-bit binary data over the bus. The RAWData command has three optional parameters: Length, Acquisitions, and Mode.

- Length specifies the number of points of each acquisition. Acquisitions specify the number of acquisitions to be taken in a single digitize operation.
- The maximum number of acquisitions is a function of the length of the acquisitions and number of channels being acquired. Buffer size is the limiting factor. The total rawdata buffer size is 800,000 bytes. The maximum number of acquisitions must satisfy the following condition:

$$\max\ acquisition = \frac{800{,}000}{(((\#\ of\ points) * 2) + 74)\ (\#\ of\ active\ \ channels)}$$

- Mode choices are NORMal, AVERage, or ENVelope. Mode is used to select how the acquired data is processed. When NORMal is selected, each individual data acquisition is available. When AVERage is selected, all acquisitions are averaged, and the resulting data is available. In envelope, the maximum and minimum points of all acquisition are determined, and the resulting data is available.

The parameters for RAWData are optional, but are also dependent on one another. For example, if you specify the number of acquisitions, then you must specify the length. The length parameter can be specified without specifying the number of acquisitions.

The command :WAVeform:FORMat has no effect in the rawdata mode. Data is always transferred in the WORD format.

The Rawdata mode is exited using SEQuential:DISPlay OFF, or by specifying another acquisition type before performing a digitize.

It is highly recommended that the SEQuential Subsystem be used when using sequential-shot mode (Rawdata mode). Rawdata commands are included only for compatibility between oscilloscopes.

Peak Detect

The :ACQire type peak detect command sets the oscilloscope to the Peak Detect mode. This mode is only available when the TIMebase:SAMPle is set to real-time mode, and sequential single-shot turned off. Peak Detect is an acquisition mode (effects how the oscilloscope acquires data) rather than a display mode.

When selected, peak detect stores the minimum and maximum values for each time bucket. Peak detect can detect excursions as small as 1 nanosecond. When peak detect is on, the sample rate that appears in the timebase menu is the sample rate at which the minimum and maximum pairs are stored (up to 250MSa/s). Because peak detect stores two data points per sample clock, the memory size is divided in half. For example, if the record length is set to 32768 and peak detect is enabled, the record length changes to 16384.

COMPlete

Command

```
:ACQuire:COMPlete <complete_argument>
```

The :ACQuire:COMPlete command specifies the minimum completion criteria for an acquisition performed by the DIGitize command. The parameter determines what percentage of the time buckets need to be "full" before an acquisition is considered complete. If you are in the NORMal mode, the instrument only needs one data point per time bucket for that time bucket to be considered full. In order for the time bucket to be considered full in the AVERage or ENVelope mode, a specified number of data points (COUNt) must be acquired.

The range for the COMPlete command is 0 to 100 and indicates the minimum percentage of time buckets that must be "full" before the acquisition is considered complete. If the complete value is set to 100%, all time buckets must contain data for the acquisition to be considered complete. If the complete value is set to zero, then one acquisition cycle will take place.

> In the real-time mode, and in the repetitive mode at sweep speeds slower than 50 ns per division, 100% of the time buckets are always filled in a single acquisition. This is due to the fast sample rate of the oscilloscope and does not vary with the value specified by the :ACQuire:COMPlete command.

`<complete_argument>` ::= 0 to 100 percent (integer - NR1 format)

Example

```
OUTPUT 707;":ACQUIRE:COMPLETE 85"
```

Query

```
:ACQuire:COMPlete?
```

The COMPlete query returns the completion criteria for the currently selected mode.

Returned Format

```
[:ACQuire:COMPlete] <complete_argument><NL>
```

<complete_argument> ::= 0 to 100 (integer - NR1 format)

Example

```
OUTPUT 707;":ACQ:COMP?"
ENTER 707;Cmp
PRINT Cmp
```

COUNt

Command

```
:ACQuire:COUNt <count_argument>
```

In the average mode, the :ACQuire:COUNt command specifies the number of values to be averaged for each time bucket before the acquisition is considered complete for that time bucket. The :ACQuire:COUNt command is only valid when the oscilloscope is set to TIMebase:SAMPle REPetitive.

When the acquisition type is set to ENVelope, the count can be any value between 1 and 2048 and represents the number of hits per bucket.

When the acquisition type is set to AVERage, the count can range from 1 to 2048. Any value can be sent in this mode; however, the value will be rounded to the nearest power of two.

When the acquisition type is set to NORMal, the count is 1.

`<count_argument>` ::= 1 to 2048 (depending on the acquisition type) (integer - NR1 format)

Example

```
OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE" !select sampling mode
OUTPUT 707;":ACQUIRE:TYPE AVERAGE" !select acquisition type
OUTPUT 707;":ACQUIRE:COUNT 1024"
```

Query

```
:ACQuire:COUNt?
```

The COUNt query returns the currently selected count value.

Returned Format

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

`<count_argument>` ::= 1 through 2048 (integer - NR1 format)

Example

```
OUTPUT 707;":ACQ:COUN?"
ENTER 707;Cnt
PRINT Cnt
```

POINts

Command

```
:ACQuire:POINts <points_argument>
```

The :ACQuire:POINts command specifies the number of time buckets for each acquisition record.

- When operating in the repetitive mode, the legal setting is 500. If a value is sent that is not a legal value, it is set to 500.
- When operating in the real-time mode (sequential mode OFF), the legal settings are 512, 1024, 2048, 4196, 8192, 16384, and 32768. If a value is sent that is not a legal value, it is set to the next higher allowable setting (e.g., 580 is set to 1024).
- When operating in the real-time mode (sequential mode ON), the legal settings are from 4 to 32768, dependent on the number of segments selected.

> Always query the Waveform Subsystem Points value to determine the actual number of acquired time buckets.

`<points_argument>`

::= 500 in the repetitive mode
::= 512, 1024, 2048, 4196, 8192, 16384, or 32768 in the real-time mode (sequential mode off)
::= 4 to 32768 in the real-time mode (sequential mode on), depending on the current :SEQuential:NSEGments value (integer - NR1 format)

Example

```
OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE" !select sample mode
OUTPUT 707;":ACQUIRE:POINTS 500"
```

Query

```
:ACQuire:POINts?
```

The POINts query returns the number of time buckets to be acquired.

Returned Format

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

`<points_argument>`

::= 500 in the repetitive mode

::= 512, 1024, 2048, 4196, 8192, 16384, or 32768 in the real-time mode (sequential mode off)

::= 4 to 32768 in the real-time mode (sequential mode on), depending on the current :SEQuential:NSEGments value (integer - NR1 format)

Example

```
OUTPUT 707;":ACQUIRE:POINTS?"
ENTER 707;Pnts
PRINT Pnts
```

The POINts command/query is identical to the TIMebase:RLENgth command/query.

TYPE

Command

```
:ACQuire:TYPE {NORMal | AVERage | ENVelope |
PDETect|RAWData[,<length>[,<acquisitions>]]
[,{NORMal | AVERage | Envelope}]
```

The :ACQuire:TYPE command selects the type of acquisition that is to take place when a :DIGitize root level command is executed. There are five acquisition types: NORMal, AVERage, ENVelope, PDETect, and RAWData.

> Valid settings for ACQuire:TYPE when TIMebase:REALtime is selected are NORMal, PDETect, and RAWData. Valid settings for ACQuire:TYPE when TIMebase:REPetitive is selected are NORMal, AVERage, or ENVelope.

Refer to the discussion on Acquire Type and Count for information on the different ACQuire:TYPE choices.

> To get a correct PREAMBLE with the corresponding 8000 point real-time acquisition, you must send the DIGitize command or STOP the acquisition before sending the WAVeform:DATA query.

`<length>` ::= 4 to 32768 (integer - NR1 format)

`<acquisitions>` ::= dependent on length of acquisitions and buffer size as discussed previously

Example

```
OUTPUT 707;":ACQUIRE:TYPE ENVELOPE"
```

Query

```
:ACQuire:TYPE?
```

The :ACQuire:TYPE query returns the current acquisition type except in the RAWData mode. In the RAWData mode it returns the acquisition type, length, and number of acquisitions.

Returned Format

```
[:ACQuire:TYPE] {NORMal | AVERage | ENVelope | PDETect
RAWData,<length>,<acquisitions><NL>
```

`<length>` ::= 4 to 32768 (integer - NR1 format)

`<acquisitions>` ::= dependent on length of acquisitions and buffer size as discussed previously

Example

```
OUTPUT 707;":ACQ:TYPE?"
ENTER 707;Tpe
PRINT Tpe
```

9

CALibrate Subsystem

Introduction

The CALibrate subsystem contains only two commands/queries: :DATA:ASCii, SETup, and TNULl. This subsystem calibrates the instrument for different probes, cables, or setups.

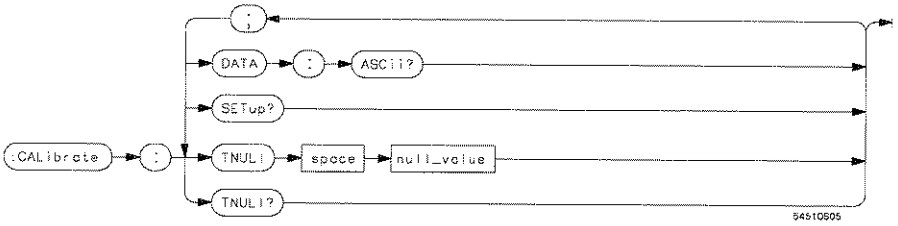
This subsystem contains the following commands:

DATA
SETup
TNULl

Figure 9-1 lists the syntax diagrams for the Calibrate subsystem commands.

> The time null is set in the Probe Cal menu of the Utility menus. For more information refer to the *HP 54520A, HP 54522A, HP 54540A, and HP 54542A Front-Panel Reference.*

Figure 9-1



null_value = channel1 to channel<n> skew, where n=2 (HP 54520A/54522A) or 2 through 4 (HP 54540A/54542A)

Calibrate Subsystem Commands Syntax Diagram

DATA:ASCii?

Query

```
:CALibrate:DATA:ASCii?
```

The :CALibrate:DATA:ASCii query returns the instrument's calibration data.

Returned Format

```
[:CALibrate:DATA:ASCii] <data>,<data>,...<NL>
```

<data> = calibration data

Example

```
DIM Data$[20000]
OUTPUT 707;":CALIBRATE:DATA:ASCII?"
ENTER 707;Data$
PRINT Data$
```

SETup?

Query

```
:CALibrate:SETup?
```

The :CALibrate:SETup query returns the current settings for the Calibrate Subsystem commands.

Returned Format

```
:CALibrate:TNUL
<null_value_n>,<null_value_n>,<null_value_n><NL>
```

`<null_value_n>` ::= channel 1 to channel<n> skew,
where n = 2 for the HP 54520A/54522A
or n = 1 through 4 for the HP 54540A/54542A (exponential - NR3 format)

Example

```
DIM Stp$[100]
OUTPUT 707;":CALIBRATE:SETUP?"
ENTER 707;Stp$
PRINT Stp$
```

TNULl

Command

```
:CALibrate:TNULl <null_value_n>
```

The :CALibrate:TNULl command sends the time null (channel-to-channel skew) values to the oscilloscope. The time null values should have been obtained from the instrument during a previous setup.

`<null_value_n>` ::= channel 1 to channel<n> skew, where n = 2 (HP 54520A/54522A) or 2 through 4 (HP 54540A/54542A) (exponential - NR3 format)

Example

```
On the two-channel models
  OUTPUT 707;":CALIBRATE:TNULL 5NS"

On the four-channel models
  OUTPUT 707;":CALIBRATE:TNULL 5NS, 6 ns, 7 ns"
```

Query

```
:CALibrate:TNULl?
```

The TNULl query tells the instrument to output the time null values to the controller.

Returned Format

```
[:CALibrate:TNULl] <null_value_n><NL>
```

`<null_value_n>` ::= channel 1 to channel<n> skew, where n =
::= 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (exponential - NR3 format)

Example

```
DIM N11$[50]
OUTPUT 707;":CAL:TNUL?"
ENTER 707;N11$
PRINT N11$
```

10

CHANnel Subsystem

Introduction

The CHANnel subsystem commands control the vertical (Y-axis) of the oscilloscope. Each channel is independently programmable for all offset (position), probe, coupling, and range functions. The channel number specified in the command selects the channel that is affected by the command. The channels available depend on the oscilloscope model. Channels 1 and 2 are available on the HP 54520A/54522A, and channels 1 through 4 are available on the HP 54540A/54542A.

The EXTernal subsystem commands control the vertical of the external trigger for the HP 54520A and 54522A oscilloscopes.

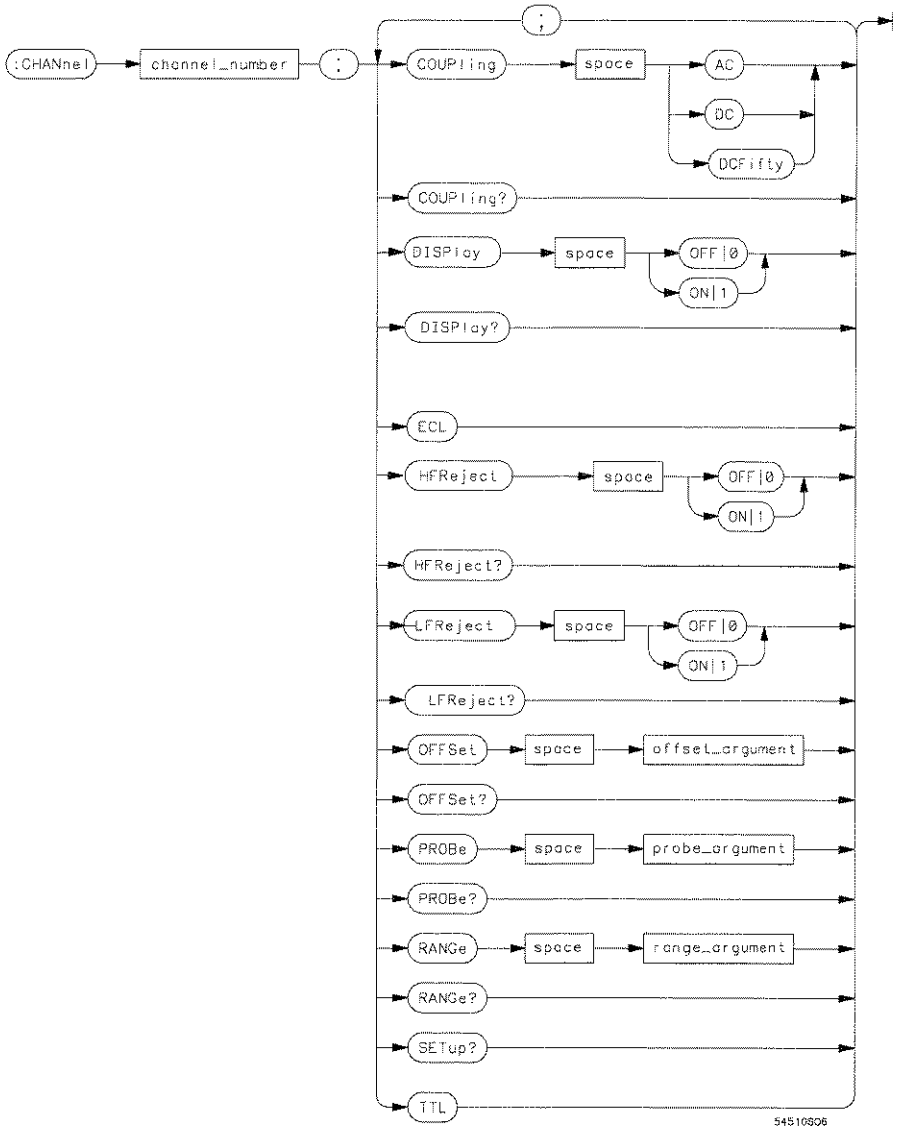
The channel displays are toggled on and off with the root level commands VIEW and BLANK, or using the CHANnel:DISPlay command.

The Channel subsystem contains the following commands:

COUPling
DISPlay
ECL
HFReject
LFReject
OFFSet
PROBe
RANGe
SETup
TTL

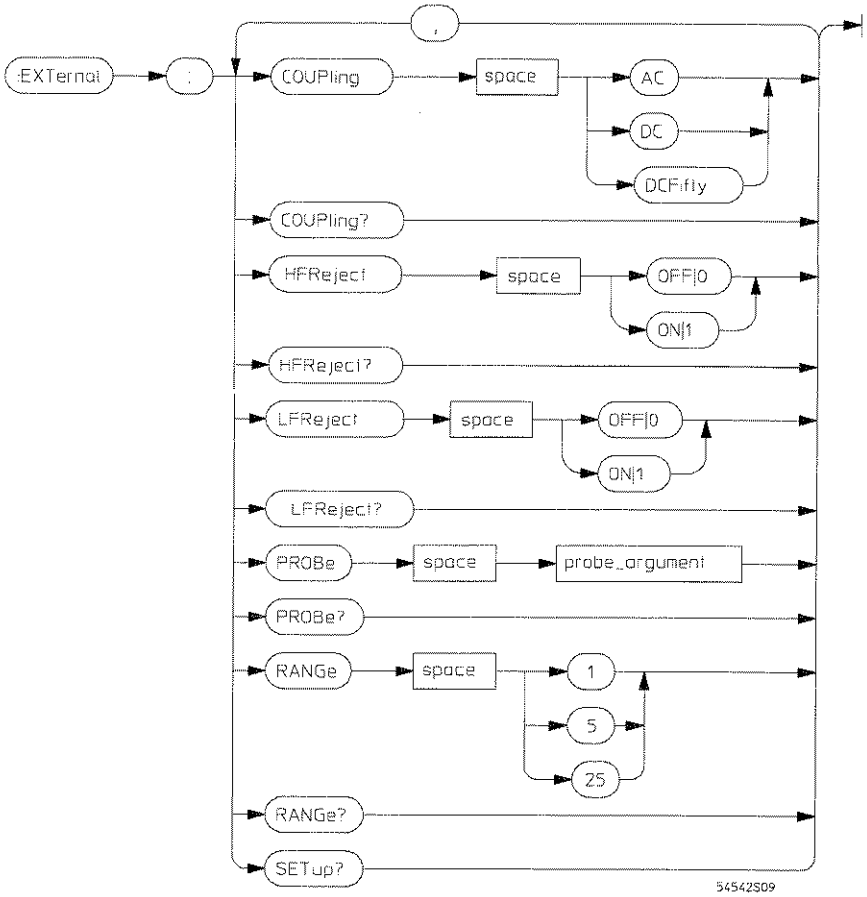
Figure 10-1 lists the syntax diagrams for the Channel subsystem commands.

Figure 10–1



Channel Subsystem Commands Syntax Diagram

Figure 10–1



External Channel Subsystem Commands Syntax Diagramon for the 54520A and 54522A oscilloscopes

channel_number = an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).

offset_argument = a real number defining the voltage at the center of the display range.

probe_argument = a real number from 0.9 to 1000.0 specifying the probe attenuation with respect to 1.

range_argument = a real number specifying the size of the acquisition window in volts.

COUPling

Command

```
:CHANnel<n>:COUPling {AC | DC | DCFifty}

:EXTernal:COUPling {AC | DC | DCFifty}
(on 54520A and 54522A only)
```

The :CHANnel<n>:COUPling command selects the input coupling for the specified channel. The coupling for each channel can be set to AC, DC, or DCFifty. DCFifty places an internal 50 Ω load on the input. AC and DC automatically set the input impedance to 1M ohm.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHANNEL2:COUPLING DC"
```

Query

```
:CHANnel<n>:COUPling?
```

The COUPling query returns the current coupling for the specified channel.

Returned Format

```
[:CHANnel<n>:COUPling] {AC | DC | DCFifty}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Ch$[50]
OUTPUT 707;":CHAN2:COUP?"
ENTER 707;Ch$
PRINT Ch$
```

DISPlay

Command

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

The :CHANnel<n>:DISPlay command controls the individual channel displays. On starts displaying, and off stops displaying the channel selected.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHANNEL2:DISPLAY ON"
```

Query

```
:CHANnel<n>:DISPlay?
```

The DISPlay query returns the current setting of the command.

Returned Format

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

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHAN1:DISP?"
ENTER 707;Disp
PRINT Disp
```

ECL

Command

```
:CHANnel<n>:ECL
```

The :CHANnel<n>:ECL command sets the vertical range, offset (position), channel coupling, and trigger level of the selected channel for optimum viewing of ECL signals. The channel offset (position) and trigger level are set to –1.3 volts and the range is set to 1.6 volts full scale. Channel coupling is set to DC.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHANNEL1:ECL"
```

HFReject

Command

```
:CHANnel<n>:HFReject {{OFF | 0} | {ON | 1}}

:EXTernal:HFReject {{OFF | 0} | {ON | 1}}
(on 54520A and 54522A only)
```

The :CHANnel<n>:HFReject command controls an internal low-pass filter. When the filter is ON, the bandwidth of the specified channel is limited to approximately 20 MHz. The bandwidth limit filter may be used when either AC, DC, or DCFifty coupling is used.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHANNEL2:HFREJECT ON"
```

Query

```
:CHANnel<n>:HFReject?
```

The HFReject query returns the current setting of the command.

Returned Format

```
[:CHANnel<n>:HFReject] {0 | 1}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHAN1:HFR?"
ENTER 707;Hf
PRINT Hf
```

LFReject

Command

```
:CHANnel<n>:LFReject {{OFF | 0} | {ON | 1}}

:EXTernal:LFReject {{OFF | 0} | {ON | 1}}
(on 54520A and 54522A only)
```

The :CHANnel<n>:LFReject command controls an internal high-pass filter. When the filter is ON, the bandwidth of the specified channel is limited to approximately 400 Hz. The bandwidth limit filter may be used only when AC coupling is used. Otherwise, the bandwidth limit filter is automatically turned off.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHANNEL2:COUPLING AC"     !select AC coupling
OUTPUT 707;":CHANNEL2:LFREJECT ON"
```

Query

```
:CHANnel<n>:LFReject?
```

The LFReject query returns the current setting of the command.

Returned Format

```
[:CHANnel<n>:LFReject] {0 | 1}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHAN1:LFR?"
ENTER 707;Lf
PRINT Lf
```

OFFSet

Command

```
:CHANnel<n>:OFFSet <offset_argument>
```

The :CHANnel<n>:OFFSet command sets the voltage that is represented at center screen for the selected channel. This is the same as the vertical position control on the front panel. The range of legal values vary with the value set with the RANGe command. If you set the offset to a value outside of the legal range, the offset value is automatically set to the nearest legal value. Because offset is dependent on the vertical range currently selected, CHANnel:PROBe and CHANnel:RANGe must be specified prior to changing CHANnel:OFFSet.

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<offset_argument>` ::= offset value (exponential - NR3 format)

Example

```
OUTPUT 707;":CHANNEL1:OFFSET 200M"
OUTPUT 707;":CHANNEL2:OFFSET 20E-3"
```

Query

```
:CHANnel<n>:OFFSet?
```

The OFFSet query returns the current offset value for the selected channel.

Returned Format

```
[:CHANnel<n>:OFFSet] <offset_argument><NL>
```

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<offset_argument>` ::= offset value in volts (exponential - NR3 format)

Example

```
OUTPUT 707;":SYSTEM:HEADERS OFF"     !turn headers off
OUTPUT 707;":CHAN2:OFFS?"
ENTER 707;Offset
PRINT Offset
```

PROBe

Command

```
:CHANnel<n>:PROBe <probe_argument>

:EXTernal:PROBe <probe_argument>
(on 54520A and 54522A only)
```

The :CHANnel<n>:PROBe command specifies the probe attenuation factor for the selected channel. The range of the probe attenuation factor is from 0.9 to 1000.0. This command does not change the actual input sensitivity of the oscilloscope. It changes the reference constants for scaling the display factors, for making automatic measurements, for setting trigger levels, etc. Because of the coupling that occurs between the vertical and trigger level controls, CHANnel:PROBe must be specified prior to changing CHANnel:OFFSet, CHANnel:RANGe, or TRIGger:LEVel.

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<probe_argument>` ::= 0.9 to 1000.0 (exponential - NR3 format)

Example

```
OUTPUT 707;":CHANNEL2:PROBE 10"
```

Query

```
:CHANnel<n>:PROBe?
```

The PROBe query returns the current probe attenuation factor for the selected channel.

Returned Format

```
[:CHANnel<n>:PROBe] <probe_argument><NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

<probe_argument> ::= 0.9 to 1000.0 (exponential - NR3 format)

Example

```
OUTPUT 707;":CHAN1:PROB?"
ENTER 707;Prb
PRINT Prb
```

RANGe

Command

```
:CHANnel<n>:RANGe <range_argument>

:EXTernal:RANGe {1 | 5 | 25}
(on 54520A and 54522A only)
```

The :CHANnel<n>:RANGe command defines the full-scale vertical axis of the selected channel. The RANGe for any channel can be set to any value from 8 mV to 40 V when using 1:1 probe attenuation. If the probe attenuation is changed, the range value is multiplied by the probe attenuation factor.

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<range_argument>` ::= full-scale range value (exponential - NR3 format)

Examples

```
OUTPUT 707;":CHANNEL1:RANGE 0.64"
OUTPUT 707;":CHANNEL2:RANGE 1.2 V"
```

Query

```
:CHANnel<n>:RANGe?
```

The RANGe query returns the current full-scale range setting for the specified channel.

Returned Format

```
[:CHANnel<n>:RANGe] <range_argument><NL>
```

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<range_argument>` ::= full-scale range value (exponential - NR3 format)

Example

```
OUTPUT 707;":CHAN2:RANG?"
ENTER 707;Rng
PRINT Rng
```

SETup?

Query

```
:CHANnel<n>:SETup?

:EXTernal:SETup? (on 54520A and 54522A only)
```

The :CHANnel<n>:SETup query returns the current settings for the CHANnel Subsystem commands.

Returned Format

```
:CHAN<n>:COUP {AC|DC|DCF};
  DISP {0 | 1};
  HFR {0 | 1};
  LFR {0 | 1};
  OFFS <offset_ argument>;
  PROB <probe_argument>;
  RANG <range_argument><NL>
```

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<offset_argument>` ::= offset value in volts (exponential - NR3 format)

`<probe_argument>` ::= 0.9 to 1000.0 (exponential - NR3 format)

`<range_argument>` ::= full-scale range value (exponential - NR3 format)

Example

```
DIM Stp$[300]
OUTPUT 707;":CHANNEL1:SETUP?"
ENTER 707;Stp$
PRINT Stp$
```

TTL

Command

```
:CHANnel<n>:TTL
```

The :CHANnel<n>:TTL command sets the vertical range, offset (position), channel coupling, and trigger level of the selected channel for optimum viewing of TTL signals. The channel offset (position) is set to 2.5 volts. The trigger level is set to 1.4 volts and the range is set to 8.0 volts full scale. Channel coupling is set to DC.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":CHANNEL1:TTL"
```

11

DISK Subsystem

Introduction

The DISK subsystem commands perform the disk operations as defined under the disk menu. This allows storage and retrieval of waveforms, setups, pixel memory, and masks. It also allows formatting the disk and making directories.

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

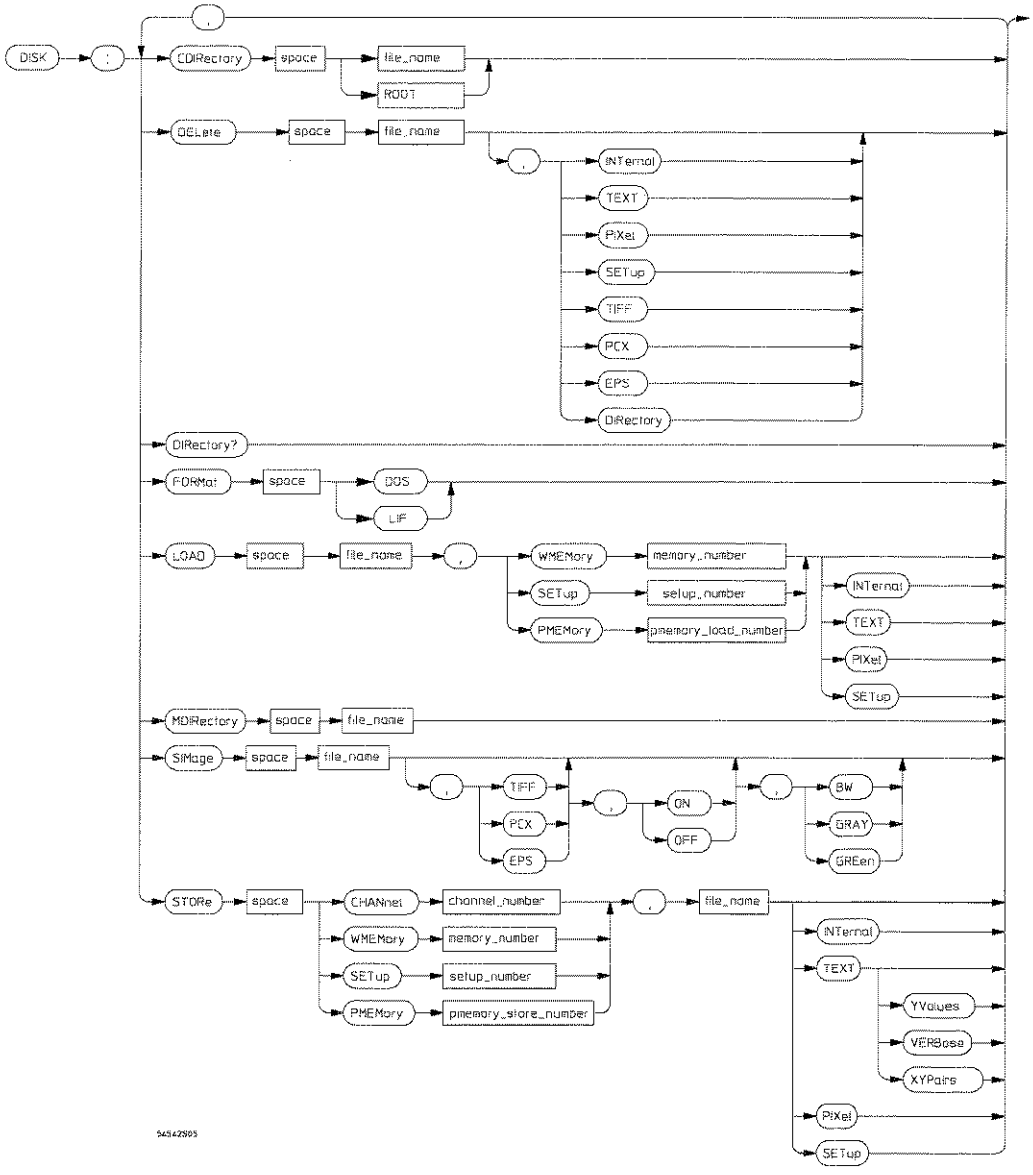
> The filename must be enclosed in quotes.

The Disk subsystem contains the following commands and queries:

CDIRectory
DELete
DIRectory?
FORMat
LOAD
MDIRectory
PWD?
SIMage
STORe

Figure 11-1 is the syntax diagram of the Disk Subsystem.

Figure 11–1



Disk Subsystem Commands Syntax Diagram

Figure 11-1

channel_num =	an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).
destination =	WMEMory{1 | 2 | 3 | 4}, SETup{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}, or PMEMory{1 | 2}.
directory_name=	a quoted ASCII string with a maximum of 65 characters
file_name =	one to 8 character quoted ASCII string. With DOS, the filename can have an optional 0 to 3 character extension.
memory_number =	an integer, 1 through 4.
setup_number =	integer from 0 through 9.
pmemory_store_number =	integer from 0 through 2.
pmemory_load_number=	an integer, 1 or 2

Disk Subsystem Commands Syntax Diagram (Cont)

CDIRectory

Command

:DISK:CDIRectory <directory_name | ROOT>

The DISK:CDIRectory command changes the present working directory to the designated directory name. An error is displayed on the screen if the requested directory does not exist. If the ROOT command is used, the present working directory is changed to the root directory of the disk.

`<directoryname>` A 1 to 65 character quoted ASCII string, including subdirectory designation. The directory name and any subdirectories must be separated by \.

Example

```
OUTPUT 707;":DISK:CDIRECTORY ""DIRECTORYNAME"""
OUTPUT 707;":DISK:CDIRECTORY ROOT"
```

DELete

Command

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

The DISK:DELete command deletes a file from the disk. An error is displayed on the screen if the requested file does not exist.

`<filename>` A 1 to 8 character quoted ASCII string. With DOS, the filename can have an optional 0 to 3 character extension.

Example

```
OUTPUT 707;":DISK:DELETE ""FILE1.SET"""
```

DIRectory?

Query

```
:DISK:DIRectory?
```

This query returns the directory listing of the currently installed disk. Each entry is 70 bytes long including a carriage return and line feed.

Returned Format

```
[:DISK:DIRectory] <number_of_lines><cr><lf><directory><lf><disk_info>
```

`<number_of_lines>` ::= number of lines (that follow) in the root directory (integer-NR1 format)

`<directory>` ::= {<filename>, <date>,<time>,<size>,<description><cr><lf>...}

`<filename>` ::= one to 8 character ASCII string, if DOS, may contain an optional 0 to 3 character extension

`<date>` ::= DDMMMYY (month is alphabetic)

`<time>` ::= HH:MM:SS

`<size>` ::= an integer

`<description>` ::= DOS format: <model> DOS FILE - NO DESC

::= LIF format: <model> <source> <type>

`<model>` ::= HP 545XX

`<source>` ::= {CHAN<n> | WMEM {0 | 1 | 2 | 3} |
SET {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9} | PMEM {0 | 1 | 2}}

`<type>` ::= {SET | WAV | PIX | TXT}

`<disk_info>` ::= DOS disk space (bytes) - Total: 1474560 Free: 1350144
::= LIF disk space (blocks) - Total: 6122 Free: 6024 Largest: 6024

FORMat

Command

```
:DISK:FORMat <format_type>
```

The DISK:FORMat command formats a disk in the drive. It is assumed that the disk that is to be formatted is in the drive when the command is issued.

<format_type> ::= {DOS | LIF}

Example

```
OUTPUT 707;":DISK:FORMAT DOS"
```

All data on the disk is lost once the DISK:FORMat command is executed.

LOAD

Command

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

The DISK:LOAD command restores a setup from the disk, a waveform or a pixel memory. The type of file is determined by the filename suffix if one is present or by the destination field if one is not present.

`<filename>` Quoted ASCII string. Up to 8 characters, if DOS may include an optional 0 to 3 character extension. Either .wav or .txt may be used as an extension after the filename. If no file extension is specified, the default is .wav. See table below for the file that is loaded depending on the destination and extension.

`<destination>` ::= {WMEMory {1 | 2 | 3 | 4 } | SETup{0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9} | PMEMory{ 1 | 2}}

`<format>` ::= {TEXT | INTernal}

Example

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

Extensions	Destinations		
	PIXel	SETup	WMEMory
DOS with extension	filename.pix	filename.set	filename.wav filename.txt
DOS without extension or LIF	filename	filename	if <format> is TEXT, then filename.txt, else filename.wav

See also

the **File Formats** section for a description of the different file formats.

MDIRectory

Command

```
:DISK:MDIRectory <directory_name>
```

The DISK:MDIRectory command creates a directory in the present working directory with the designated directory name. If a subdirectory that does not exist is designated in the directory name, an error message appears on the screen.

`<directoryname>` A 1 to 65 character quoted ASCII string, including subdirectory designation. The directory name and any designated subdirectories must be separated by \, such as PXX\H123\INN.

Example

```
OUTPUT 707;":DISK:MDIRECTORY ""DIRECTORYNAME"""
```

PWD?

Command

```
:DISK:PWD?
```

The DISK:PWD? query returns the name of the present working directory. The entry is an ASCII string, such as \xyz\234.

Returned Format

```
[:DISK:PWD] <present_working_directory>
```

Example

```
OUTPUT 707;":DISK:PWD?"
```

SIMage

Command

```
:DISK:SIMage <file_name> [|,<format>|
[|,<compression>| [,<rendering>]]]
```

The DISK:SIMage command stores a screen image in TIFF, PCX, or EPS format. You can add a file extension, but the scope does not require that you do so. If you omit the extension and the disk is DOS formatted, an extension is supplied by the scope depending on the selected file format. If you add an extension and the disk is LIF formatted, the extension is ignored.

`<filename>` ::= a descriptive name of the file up to 8 characters long. With DOS, the filename can have an optional 0 to 3 character extension. You can use .TIF, .PCX or .EPS as an extension after the filename.

`<format>` ::= {TIFF | PCX | EPS} (The default value is TIFF.)

`<compression>` ::= {ON | OFF} (The default value is ON.)

`<rendering>` ::= {BW | GRAY [GREen} (The default value is BW.)

Examples

```
OUTPUT 707;":DISK:SIM ""FILE1.PCX"", PCX, ON, GRAY"
or
OUTPUT 707;":DISK:SIM ""FILE1.TIF"", TIFF, ON"
or
OUTPUT 707;":DISK:SIM ""FILE1"""
```

STORe

Command

```
:DISK:STORe <source>,<file_name>[,<format>]
```

The DISK:STORe command stores a setup, a waveform, or a pixel memory to the disk. The filename does not include a suffix. The suffix is supplied by the instrument depending on the source and file format specified.

`<source>` ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4} |
SETup {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9} | PMEMory {1 | 2}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<filename>` ::= a descriptive name of the file up to 8 characters long. With DOS, the filename can have an optional 0 to 3 character extension.

`<format>` ::= {INTernal | TEXT [,{<YVALues> | <VERBose> | <XYPairs>}]}

The format field is for waveforms, the default is INTernal. In TEXT mode, y values may be specified so that only the y values are stored. VERBose is the default in which y values and the waveform preamble is stored.

Example

```
OUTPUT 707;":DISK:STORE CHANNEL1 ""FILE1.WAV","TEXT"
```

See also

the **File Formats** section for a description of the different file formats.

File Formats

The following is a description of the verbose header format.

Type Type describes how the waveform was acquired: normal, raw, interpolate, average, or versus. When this field is read back into the oscilloscopes, all the modes, except versus, are converted to raw. The default value is normal.

Points Points indicates the number of data points contained in the waveform record. The number of points is set by the Record length softkey in the Acquisition menu. The default value is 500.

> The number of points stored when the oscilloscope is in the envelope display mode is twice the number of points shown in the verbose header. In this mode, two waveforms are stored: the minimum value points waveform and the maximum value points waveform.

Count Count represents the minimum number of hits at each time bucket in the waveform record when the waveform was created using an ensemble acquisition mode, like averaging. For example, when averaging, a count of four would mean every waveform data point in the waveform record has been averaged at least four times. Count is ignored when it is read back into the oscilloscope. The default value is 0.

XInc X increment is the time duration between data points on the X-axis. X increment is equal to the YData range value for versus waveforms. The default value is $2\ E^{-6}$.

XOrg X origin is the x-value of the first data point in the data record. X origin is equal to the YData center value for versus waveforms. The default value is $-1.3107\ E^{-6}$

XRef X reference is always set to zero when it is read back into the oscilloscope. The default value is $3.125\ E^{-2}$.

YData range Y data range is the full voltage range covered by the A/D converter. If this field is omitted, it is calculated when read back into the oscilloscope. The default value is 0.

YData center YData center is voltage level at the center of the YData range. If this field is omitted, it is calculated when read back into the oscilloscope. The default value is 0.

Coupling Coupling is ignored when it is read back into the oscilloscope. The default value is dc 50 Ω.

XRange XRange is the time duration across 10 horizontal divisions of the display. The default value is 100 E $^{-6}$.

XOffset XOffset is the time at the left edge of the display. The default value is 0.

YRange YRange is the voltage across eight vertical divisions of the display. The default value is 4.0 E $^{-0}$.

YOffset YOffset is the voltage at the center of the display. The default value is 0.

Date Date is the date when the waveform was acquired. The default value is 10 AUG 1992.

Time Time is the time when the waveform was acquired. The default value is 01:00:00.

Frame Frame is the model and serial number of the oscilloscope that acquired the waveform. The default value is 54542A:00000A00000.

Acq mode Acquisition mode is the sampling mode used to acquire the waveform, either real time or equivalent time. The default value is real time.

Completion Completion represents the percent of the time buckets in the waveform record that contain data. The number of time buckets is equal to the number of points in the waveform record. Completion is ignored when it is read back into the oscilloscope. The default value is 100.

X Units X units is the horizontal scaling units set in the channel menu: unknown, volt, second, constant, or ampere. The default value is unknown.

Y Units Y units is the vertical scaling units set in the channel menu: unknown, Volt, second, constant, or Ampere. The default value is unknown.

Max bandwidth Maximum bandwidth is an estimation of the maximum bandwidth limit of the source signal. The default value is 500 MHz.

Min bandwidth Minimum bandwidth is an estimation of the minimum bandwidth limit of the source signal. The default value is 0.

Data The data is in exponential format with a value of 9.9999E+37 representing a data record which has no valid data.

The following figures show all of the different file formats that can be stored on disk for text waveforms.

Figure 11-2

```
Type:              Normal
Points:            500
Count:             1
XInc:              9.99999971718E-10
XOrg:              0.00000000000000E+00
XRef:              0
YData range:       1.60000E-01
YData center:      0.000000E+00
Coupling:          DC
XRange:            5.00000E-07
XOffset:           -2.50000000000000E-07
YRange:            1.60000E-01
YOffset:           0.00000E+00
Date:              1 NOV 1995
Time:              1:06:21
Frame:             54542A:3207A00101
Acq mode:          equivalent time
Completion:        100
X Units:           second
Y Units:           Volt
Max bandwidth:     500000000
Min bandwidth:     0
Data:
-3.14173E-02
-3.14173E-02
     .
     .
     .
4.04331E-02
4.02756E-02
```

Equivalent time acuisition mode type normal verbose format for a text waveform

Figure 11-3

```
Type:              Average
Points:            500
Count:             1
XInc:              9.99999971718E-10
XOrg:              0.00000000000000E+00
XRef:              0
YData range:       1.60000E-01
YData center:      0.000000E+00
Coupling:          DC
XRange:            5.00000E-07
XOffset:           -2.50000000000000E-07
YRange:            1.60000E-01
YOffset:           0.00000E+00
Date:              1 NOV 1995
Time:              1:06:21
Frame:             54542A:3207A00101
Acq mode:          equivalent time
Completion:        100
X Units:           second
Y Units:           Volt
Max bandwidth:     500000000
Min bandwidth:     0
Data:
-3.14173E-02
-3.14173E-02
    .
    .
    .
4.04331E-02
4.02756E-02
```

Equivalent time acuisition mode type average verbose format for a text waveform

Figure 11-4

```
Type:              Envelope
Points:            500
Count:             1
XInc:              9.99999971718E-10
XOrg:              0.000000000000000E+00
XRef:              0
YData range:       1.60000E-01
YData center:      0.000000E+00
Coupling:          DC
XRange:            5.00000E-07
XOffset:           -2.500000000000000E-07
YRange:            1.60000E-01
YOffset:           0.00000E+00
Date:              1 NOV 1995
Time:              1:06:21
Frame:             54542A:3207A00101
Acq mode:          equivalent time
Completion:        100
X Units:           second
Y Units:           Volt
Max bandwidth:     500000000
Min bandwidth:     0
Data:
-3.14173E-02                                  Start of lower waveform
-3.14173E-02
    .
    .
    .
4.04331E-02
4.02756E-02
-3.14753E-02                                  Start of upper waveform
-3.147573E-02
    .
    .
    .
4.0E-02
4.02756E-02
```

Equivalent time acuisition mode type envelope verbose format for a text waveform

Figure 11-5

```
Type:              Normal
Points:            512
Count:             1
XInc:              9.99999971718E-10
XOrg:              0.00000000000000E+00
XRef:              0
YData range:       1.60000E-01
YData center:      0.000000E+00
Coupling:          DC
XRange:            5.00000E-07
XOffset:           -2.50000000000000E-07
YRange:            1.60000E-01
YOffset:           0.00000E+00
Date:              1 NOV 1995
Time:              1:06:21
Frame:             54542A:3207A00101
Acq mode:          real time
Completion:        100
X Units:           second
Y Units:           Volt
Max bandwidth:     500000000
Min bandwidth:     0
Data:
-3.14173E-02
-3.14173E-02
     .
     .
     .
4.04331E-02
4.02756E-02
```

Real time acuisition mode (greater than 500 points) type normal verbose format for a text waveform

Figure 11-6

```
Type:              Sequential Normal
Acquisitions:      10 of 10 stored to disk
Points:            512
Count:             1
Xinc:              9.99999971718E-10
XOrg:              0.00000000000000E+00
XRef:              0
YData range:       1.60000E-01
YData center:      0.000000E+00
Coupling:          DC
XRange:            5.00000E-07
XOffset:           -2.50000000000000E-07
YRange:            1.60000E-01
YOffset:           0.00000E+00
Date:              1 NOV 1995
Time:              1:06:21
Frame:             54542A:3207A00101
Acq mode:          real time
Completion:        100
X Units:           second
Y Units:           Volt
Max bandwidth:     500000000
Min bandwidth:     0
Data:
Timetag:           #1, 0.00000000000000E+00
-3.14173E-02
    .
    .
    .
4.04331E-02
    .
    .                                          Timetag 2 through 9
    .
Timetag:           #10, 1.11721189047721E-03
-3.14173E-02
    .
    .
    .
4.04331E-02
```

Real time acuisition mode type sequential normal verbose format for a text waveform

This file format cannot be loaded into the oscilloscope waveform memories.

Figure 11-7

```
Type:              Sequential Average
Acquisitions:      10 averaged
Points:            500
Count:             1
XInc:              9.99999971718E-10
XOrg:              0.00000000000000E+00
XRef:              0
YData range:       1.60000E-01
YData center:      0.000000E+00
Coupling:          DC
XRange:            5.00000E-07
XOffset:           -2.50000000000000E-07
YRange:            1.60000E-01
YOffset:           0.00000E+00
Date:              1 NOV 1995
Time:              1:06:21
Frame:             54542A:3207A00101
Acq mode:          real time
Completion:        100
X Units:           second
Y Units:           Volt
Max bandwidth:     500000000
Min bandwidth:     0
Data:
-3.14173E-02
-3.14173E-02
    .
    .
    .
4.04331E-02
4.02756E-02
```

Real time acuisition mode type sequential average verbose format for a text waveform

Figure 11-8

```
Type:              Sequential Envelope
Acquisitions:      10 enveloped
Points:            500
Count:             1
XInc:              9.99999971718E-10
XOrg:              0.00000000000000E+00
XRef:              0
YData range:       1.60000E-01
YData center:      0.000000E+00
Coupling:          DC
XRange:            5.00000E-07
XOffset:           -2.50000000000000E-07
YRange:            1.60000E-01
YOffset:           0.00000E+00
Date:              1 NOV 1995
Time:              1:06:21
Frame:             54542A:3207A00101
Acq mode:          real time
Completion:        100
X Units:           second
Y Units:           Volt
Max bandwidth:     500000000
Min bandwidth:     0
Data:
-3.14173E-02                                  Start of lower waveform
-3.14173E-02
    .
    .
    .
4.04331E-02
4.02756E-02
-3.14753E-02                                  Start of upper waveform
-3.147573E-02
    .
    .
    .
4.0E-02
4.02756E-02
```

Real time acuisition mode type sequential envelope verbose format for a text waveform

Text Y values Text Y value files are identical to the text verbose files, except the header information is deleted from the front of the file. Figure 12-9 shows an example of the text Y value format for all modes except for the sequential normal type which is shown in figure 11-10. Text Y value files also have a .TXT file extension.

Text Y value files are intended for importing into other applications. They do not contain any header information. If they are loaded into the oscilloscope, the current horizontal setup of the oscilloscope is used but the vertical setup is adjusted to fit the waveform on screen.

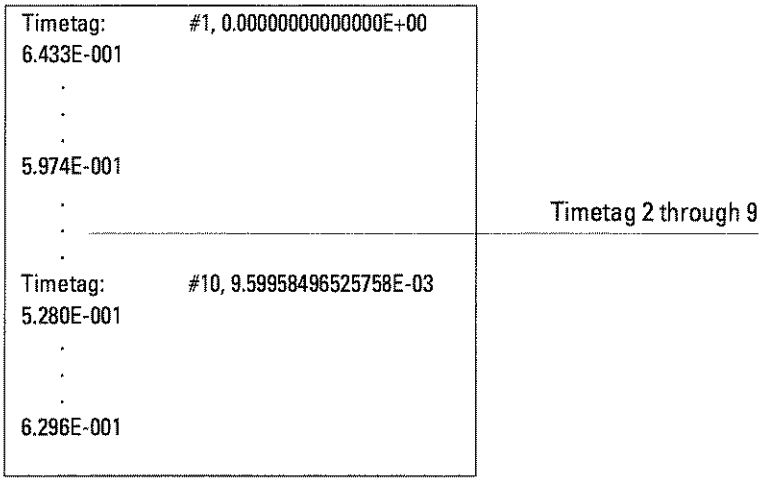
Figure 11-9

```
5.657E-001
6.433E-001
5.974E-001
1.864E-001
8.05E-002
-3.147E-001
-3.057E-001
-4.77E-002
5.280E-001
6.296E-001
6.296E-001
2.297E-001
1.259E-001
-3.147E-001
-3.147E-001
-9.40E-002
```

Text Yvalue format

If the file is of type envelope which contains two waveforms then the loaded waveform will be the top waveform appended by the bottom waveform. The reason for this is there is no way to tell from the file that it is an envelope type without the header information.

Figure 11-10



Text Yvalues for sequential normal type format

This file format cannot be loaded into the oscilloscope waveform memories.

XY pairs XY pairs files are identical to the text Y value files, except they also contain the relative time of the measurement. Like text Y value files, header information is deleted from the front of the file. XY pairs files also have a .TXT file extension.

Figure 11-11 through figure 11-17 show examples of the different XY pairs formats. The first column is the time, the second column is the voltage. Negative time indicates events before the trigger.

> XY pairs files are intended for importing into other applications and cannot be loaded into the oscilloscope.

Figure 11-11

```
XYPairs: Normal
Acquistions: 1
Points: 500
-2.5763e-03,-1.04E+00
-2.5663e-03,-1.04E+00
    .
    .
    .
+2.3737e-03,-2.47E-02
+2.3837e-03,-2.47E-02
```

Equivalent time acuisition mode type normal format for XYpairs text waveform

Figure 11-12

```
XYPairs: Average
Acquistions: 1
Points: 500
-2.5763e-03,-1.04E+00
-2.5663e-03,-1.04E+00
    .
    .
    .
+2.3737e-03,-2.47E-02
+2.3837e-03,-2.47E-02
```

Equivalent time acuisition mode type average format for XYpairs text waveform

Figure 11-13

```
XYPairs: Envelope
Acquistions: 1
Points: 500
-2.5763e-03,-1.04E+00   <-- Start of lower waveform
-2.5663e-03,-1.04E+00
    .
    .
    .
+2.3737e-03,-2.47E-02
+2.3837e-03,-2.47E-02
-3.1475e-02,-2.00E-02   <-- Start of upper waveform
-3.1475e-02,-2.00E-02
    .
    .
    .
4.0315e-02,-3.14E-02
4.0275e-02,-3.15E-02
```

Equivalent time acuisition mode type envelope format for XYpairs text waveform

Figure 11-14

```
XYPairs:  Normal
Acquistions: 1
Points: 512
-2.5763e-03,-1.04E+00
-2.5663e-03,-1.04E+00
    .
    .
    .
+2.3737e-03,-2.47E-02
+2.3837e-03,-2.47E-02
```

Real time acuisition mode (greater than 500 points) type normal format for XYpairs text waveform

Figure 11-15

```
XYPairs:  Sequential Normal
Acquistions: 10 of 10 stored to disk
Points/Acquisition: 512
Timetag: #1,  0.00000000000000E+00
-2.5535e-07,-3.11719E-02
    .
    .
    .
+2.5564e-07,3.96094E-02
    .                                  Timetag 2 through 9
    .
    .
Timetag: #10,  9.5495896525758E-03
-2.5535e-07,-3.11719E-02
    .
    .
    .
+2.5564e-07,3.96094E-02
```

Real time acuisition mode type sequential normal format for XYpairs text waveform

Figure 11-16

```
XYPairs:  Sequential Average
Acquistions: 10
Points/Acquisition: 500
-2.5535e-07,-3.11719E-02
+2.5564e-07,-3.96094E-02
     .
     .
     .
-2.4234e-07,4.04395E-02
+2.4334e-07,4.02783E-02
```

Real time acuisition mode type sequential average format for XYpairs text waveform

Figure 11-16

```
XYPairs:  Sequential Envelope
Acquistions: 10 enveloped
Points/Acquisition: 500
-2.5535e-07,-3.11719E-02   <-- Start of lower waveform
+2.5564e-07,-3.96094E-02
     .
     .
     .
-2.4234e-07,4.04395E-02
+2.4334e-07,4.02783E-02
-3.1475e-02,-2.00E-02      <-- Start of upper waveform
-3.1475e-02,-2.00E-02
     .
     .
     .
4.0315e-02,-3.14E-02
4.0275e-02,-3.15E-02
```

Real time acuisition mode type sequential envelope format for XYpairs text waveform

12

DISPlay Subsystem

Introduction

The DISPlay subsystem is used to control the display of data, markers, text, and graticules.

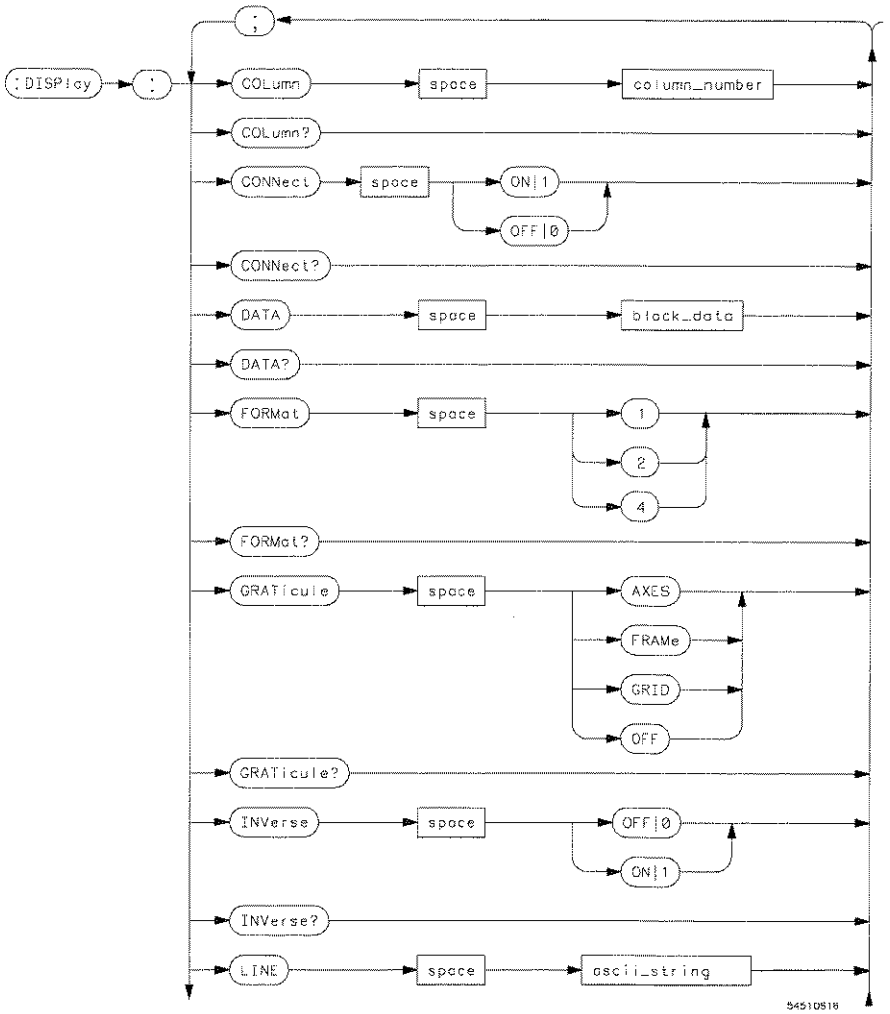
The Display subsystem contains the following commands:

COLumn	PERSistence
CONNect	ROW
DATA	SCReen
FORMat	SETup
GRATicule	SOURce
INVerse	STRing
LINE	TEXT
MASK	MARKer | TMARker | VMARker

Figure 12-1 lists the syntax diagrams for the Display subsystem commands.

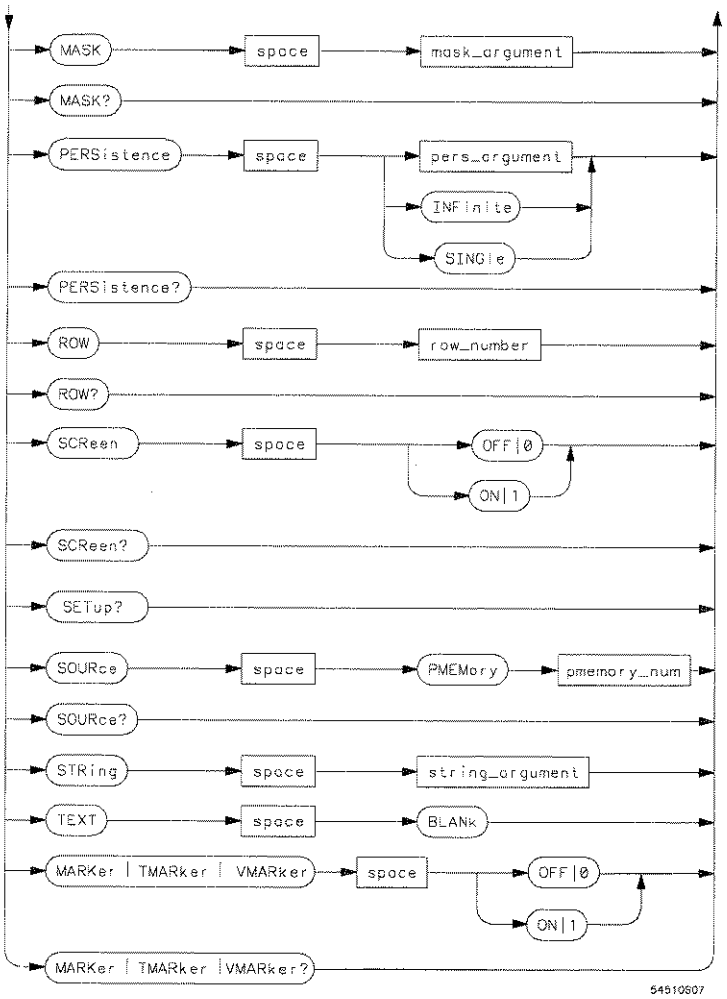
The Display mode (NORMal | PEAK | ENVelope | AVERage) is selected with the :ACQuire:TYPE command. The number of averages and envelope acquisitions is specified with the ACQuire:COUNt command.

Figure 12–1



Display Subsystem Commands Syntax Diagram

Figure 12–1



Display Subsystem Commands Syntax Diagram (continued)

Figure 12-1

column_number =	an integer, 0 through 78.
block_data =	definite block data in IEEE 488.2 # format.
mask_argument =	an integer, 0 through 255.
pers_argument =	a real number, 0.1 through 11.
pmemory_num =	an integer, 0 through 2.
row_number =	an integer, 0 through 24.
ascii_string=	any quoted ascii string.
string_argument=	text string up to 90 characters

Display Subsystem Commands Syntax Diagram (continued)

COLumn

Command

```
:DISPlay:COLumn <column_number>
```

The :DISPlay:COLumn command specifies the starting column for subsequent STRing and LINE commands.

`<column_number>` ::= 0 through 78 (integer - NR1 format)

Example

```
OUTPUT 707;":DISPLAY:COLUMN 50"
```

Query

```
:DISPlay:COLumn?
```

The COLumn query returns the column where the next LINE or STRing will start.

Returned Format

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

`<column_number>` ::= 0 through 78 (integer - NR1 format)

Example

```
OUTPUT 707;":DISP:COL?"
ENTER 707;Clmn
PRINT Clmn
```

CONNect

Command

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

The :DISPlay:CONNect command turns the connect-the-dots function on and off.

Example

```
OUTPUT 707;":DISPLAY:CONNECT ON"
```

Query

```
:DISPlay:CONNect?
```

The CONNect query returns the current setting of the connect-the-dots function. The returned status is indicated by using a 1 for on and a 0 for off.

Returned Format

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

Example

```
OUTPUT 707;":DISP:CONN?"
ENTER 707;Cnn
PRINT Cnn
```

DATA

Command

```
:DISPlay:DATA <binary_block>
```

The :DISPlay:DATA command writes waveform data to one of the pixel planes in the oscilloscope. The DATA command is followed by a block of binary data that is transferred from the controller to pixel memory 1 or 2. The data is in the IEEE 488.2 definite block form with 16576 bytes of data preceded by 10 header bytes. The block header contains the ASCII characters "#800016576" and is sent prior to the data.

`<binary_block>` ::= definite block data in IEEE 488.2 # format

PMEMory 1 or 2 is specified using the :DISPlay:SOURce command.

Query

```
:DISPlay:DATA?
```

The :DISPlay:DATA query is used to write waveform data from one of the pixel planes in the oscilloscope. The pixel planes available are planes 0 through 3. The DATA query causes the oscilloscope to output pixel data from the specified plane. If plane 0 is specified, the oscilloscope transfers the active display. If PMEMory1 or PMEMory2 is specified that memory is transferred.

> The pixel planes (PMEMory0 through PMEMory2) are specified with the :DISPlay:SOURce command.

Returned Format

```
[:DISPlay:DATA] #800016576<16576 bytes of binary data><NL>
```

Example

```
10 CLEAR 707
20 DIM Plane$[17000]
30 OUTPUT 707;":SYST:HEAD OFF"
40 OUTPUT 707;":DISPLAY:SOURCE PMEMORY0;DATA?"
50 ENTER 707 USING "-K";Plane$
60 OUTPUT 707;":DISPLAY:SOURCE PMEM1"
70 OUTPUT 707 USING "#,-K";":DISPLAY:DATA ";Plane$
80 END
```

This example transfers data from the active display memory to the controller, then transfers the data back to pixel memory 1 in the oscilloscope.

> In program line 70, the space after :DISPlay:DATA and before the quotation mark is required.

FORMat

Command

```
:DISPlay:FORMat {1 | 2 | 4}
```

The :DISPlay:FORMat command sets the number of display areas on the screen. FORMat 1 provides one display area and uses eight divisions for the full-scale range. FORMat 2 sets the number of screens to 2 and uses the full-scale range. FORMat 4 sets the number of screens to 4 and uses two divisions for the full-scale range.

Example

```
OUTPUT 707;":DISPLAY:FORMAT 1"
```

Query

```
:DISPlay:FORMat?
```

The FORMat query returns the current display format.

Returned Format

```
[:DISPlay:FORMat] {1 | 2 | 4}<NL>
```

Example

```
OUTPUT 707;":DISP:FORM?"
ENTER 707;Frmt
PRINT Frmt
```

GRATicule

Command

```
:DISPlay:GRATicule {AXES | GRID | FRAMe | OFF}
```

The :DISPlay:GRATicule command selects the type of graticule that is displayed.

Example

```
OUTPUT 707;":DISPLAY:GRATICULE AXES"
```

Query

```
:DISPlay:GRATicule?
```

Returned Format

The GRATicule query returns the type of graticule that is currently displayed.

```
[:DISPlay:GRATicule] {AXES | GRID | FRAMe | OFF}<NL>
```

Example

```
DIM Grt$[30]
OUTPUT 707;":DISP:GRAT?"
ENTER 707;Grt$
PRINT Grt$
```

INVerse

Command

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

The :DISPlay:INVerse command determines whether text sent with the LINE or STRing command in the DISPlay subsystem is to be written with the INVERSE attribute. If the inverse attribute is ON the text will be written in inverse video.

Example

```
OUTPUT 707;":DISPLAY:INVERSE OFF"
```

Query

```
:DISPlay:INVerse?
```

The INVerse query returns the current state of this command.

Returned Format

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

Example

```
OUTPUT 707;":DISP:INV?"
ENTER 707;Iv
PRINT Iv
```

LINE

Command

```
:DISPlay:LINE <ascii_string>
```

The :DISPlay:LINE command writes a text string to the screen. The text is displayed starting at the location of the current row and column. The row and column can be set by the :DISPlay:ROW and :DISPlay:COLumn commands prior to sending the :DISPlay:LINE command. Text can be written over the entire screen with the LINE command.

If the text string is longer than the available space on the current line, the text wraps around to the start of the same line. In any case, the ROW value is incremented by one and the COLumn value remains the same. The next :DISPlay:LINE command will write on the next line of the display starting at the same column as the previous text. After writing line 24, the last line in the display area, ROW is reset to 0.

<ascii_string> ::= any series of ascii characters enclosed in quotes.

Example

```
OUTPUT 707;":DISPLAY:LINE ""ENTER PROBE ATTENUATION"""
```

MASK

Command

```
:DISPlay:MASK <mask_argument>
```

The :DISPlay:MASK command inhibits the instrument from writing to selected areas of the screen. Text sent over the remote interface with the line and string commands, or the SYSTem:DSP command, is not affected by this command. The purpose of the command is to allow remote interface text to be written anywhere on the screen, and to prevent the instrument from overwriting the text through its normal operation.

The mask parameter is an 8-bit integer in which each bit controls writing to an area of the screen. A zero inhibits writing to the area represented by the bit, and a 1 enables writing to that area.

`<mask_argument>` ::= 0 to 255 (integer - NR1 format)

Example

```
OUTPUT 707;":DISPLAY:MASK 67"
```

The previous example enables writing to the Menu Area (bit 6), the Status Line (bit 1), and the Advisory Area (bit 0).

Query `:DISPlay:MASK?`

The MASK query returns the current value of the MASK command.

Returned Format `[:DISPlay:MASK] <mask_argument><NL>`

`<mask_argument>` ::= 0 to 255 (integer - NR1 format)

Example

```
OUTPUT 707;":DISP:MASK?"
ENTER 707;Msk
PRINT Msk
```

Table 12-1 Display Mask Byte.

Bit	Bit Weight	Screen Area Affected
7	128	unused
6	64	Menu Area
5	32	Timebase Information
4	16	Measurement Result Area
3	8	Graticule Area
2	4	unused
1	2	Status Line
0	1	Advisory Area

0 = masked, 1 = enabled

PERSistence

Command

```
:DISPlay:PERSistence {SINGle | INFinite |
{0 | {0.5 to 10}|11}}
```

The :DISPlay:PERSistence command sets the display persistence. The PERSistence command is only effective in the Normal display mode.

In the real-time timebase mode, the parameters for this command are the keywords INFinite or SINGle.

In the repetitive time base mode, the parameter for this command is the keyword INFinite, or real numbers from 0.5 through 10.0 representing the persistence in seconds. A value of 0 sets the PERSistence to minimum. A value of 11 seconds sets the PERSistence to INFinite.

When the key word SINGle is sent as the argument for this command, the persistence value is set to minimum.

Example

```
OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE" !select time base
mode
OUTPUT 707;":DISPLAY:PERSISTENCE 3.0"
```

In the real-time timebase mode the PERSistence query returns the actual mnemonic: SINGle or INFinite.

In the repetitive timebase mode the PERSistence query returns the current persistence value. When in minimum persistence, the value returned is 0. When in infinite persistence mode, the value returned is 11.

Query `:DISPlay:PERSistence?`

Returned Format `[:DISPlay:PERSistence] <value><NL>`

`<value>` ::= {0 (minimum) | {0.5 to 10} | 11 (infinite)} (exponential - NR3 format) in the repetitive mode
::= {SINGle | INFinite} in real-time mode

Example

```
DIM Prs$[50]
OUTPUT 707;":DISP:PERS?"
ENTER 707;Prs$
PRINT Prs$
```

ROW

Command

```
:DISPlay:ROW <row_number>
```

The :DISPlay:ROW command specifies the starting row on the screen for subsequent STRing and LINE commands. The ROW number remains constant until another ROW command is received, or until it is incremented by the LINE command. The ROW value can be set to 0 through 24.

`<row_number>` ::= 0 through 24 (integer - NR1 format)

Example

```
OUTPUT 707;":DISPLAY:ROW 10"
```

Query

```
:DISPLAY:ROW?
```

The ROW query returns the current ROW number.

Returned Format

```
[:DISPlay:ROW] <row_number><NL>
```

`<row_number>` ::= 0 through 24 (integer - NR1 format)

Example

```
OUTPUT 707;":DISP:ROW?"
ENTER 707;Rw
PRINT Rw
```

SCReen

Command

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

The :DISPlay:SCReen command turns the displayed screen on and off. The status line is the only part of the screen that remains on after the :DISPlay:SCReen OFF command is executed. The screen can be turned on again with the ON parameter.

> The command :DISPlay:TEXT BLANk removes only the text from the display.

Example

```
OUTPUT 707;":DISPLAY:SCREEN ON"
```

Query

```
:DISPlay:SCReen?
```

The SCReen query returns the current setting of this function. The returned status is indicated by using a 1 for on and a 0 for off.

Returned Format

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

Example

```
OUTPUT 707;":DISP:SCR?"
ENTER 707;Srn
PRINT Srn
```

SETup?

Query

```
:DISPlay:SETup?
```

The :DISPlay:SETup query returns the current settings for the Display Subsystem commands.

Returned Format

```
:DISP:COL <column_number>;
  CONN {0 | 1};
  FORM {1 | 2 | 4};
  GRAT {AXES | FRAM | GRID | OFF};
  INV {0 | 1};
  MASK <mask_argument>;
  PERS <pers_argument>;
  ROW <row_number>;
  SCR {0 | 1};
  SOUR PMEM {0 | 1 | 2};
  MARK {0 | 1}<NL>
```

`<column_number>` ::= 0 through 78 (integer - NR1 format)

`<mask_argument>` ::=0 to 255 (integer - NR1 format)

`<pers_argument>` ::= {0 | 0.5 to 10 | 11} (exponential - NR3 format) in the repetitive mode
::= {SINGle | INFinite} in the real-time mode

`<row_number>` ::=0 to 24 (integer - NR1 format)

Example

```
DIM Stp$[300]
OUTPUT 707;":DISP:SET?"
ENTER 707;Stp$
PRINT Stp$
```

SOURce

Command

```
:DISPlay:SOURce PMEMory{0 | 1 | 2}
```

The :DISPlay:SOURce command specifies the source or destination for the :DISPlay:DATA query and command. The SOURce command has one parameter which can be set to PMEMory0 through PMEMory2.

`PMEMory0` ::= active display

`PMEMory1` ::= pixel memory 1

`PMEMory2` ::= pixel memory 2

Example

```
10 CLEAR 707
20 DIM Plane$[17000]
30 OUTPUT 707;":SYSTEM:HEADER OFF"
40 OUTPUT 707;":DISPLAY:SOURCE PMEMORY0;DATA?"
50 ENTER 707 USING "-K";Plane$
60 OUTPUT 707;":DISPLAY:SOURCE PMEM1"
70 OUTPUT 707 USING "#,-K";":DISPLAY:DATA ";Plane$
80 END
```

This example transfers data from the active display to the controller, then transfers it back to pixel memory 1 in the oscilloscope.

Query

```
:DISPlay:SOURce?
```

The SOURce query returns the currently specified SOURce.

Returned Format

```
[:DISPlay:SOURce] PMEMory{0 | 1 | 2}<NL>
```

Example

```
DIM Src$[30]
OUTPUT 707;":DISP:SOUR?"
ENTER 707;Src$
PRINT Src$
```

STRing

Command

```
:DISPlay:STRing <string_argument>
```

The :DISPlay:STRing command writes a text string to the screen of the oscilloscope. The text is written starting at the current ROW and COLumn values. If the column limit is reached (column 78), the excess text is written over the text on the left side of that line. If 90 or more characters are sent, the error -144, "Character data too long" is produced. The STRing command does not increment the ROW value; however, the LINE command does.

`<string_argument>` ::= text string up to 90 characters

Example

```
OUTPUT 707;":DISPLAY:STRING ""INPUT SIGNAL TO CHANNEL 2"""
```

TEXT

Command

```
:DISPlay:TEXT BLANk
```

The :DISPlay:TEXT command blanks the user text area on the screen. When this command is sent, all text on the entire screen is blanked. This command has only one parameter: BLANk.

The :DISPlay:SCReen ON command restores the user text to the screen.

Example

```
OUTPUT 707;":DISPLAY:TEXT BLANK"
```

{MARKer | TMARker | VMARker}

Command

```
:DISPlay{:MARKer|TMARker|VMARker}
{{OFF | 0} | {ON | 1}}
```

The :DISPlay:{MARKer | TMARker | VMARker} command turns all markers (x1/y1 and x2/y2) on and off.

> All three commands perform the identical function, and are included only for compatibility with other oscilloscopes.

Example

```
OUTPUT 707;":DISPLAY:TMARKER OFF"
```

Query

```
:DISPlay{:MARKer|TMARker|VMARker}?
```

The {MARKer | TMARker | VMARker} query returns the current state of the specified marker.

Returned Format

```
[:DISPlay:MARKer|TMARker|VMARker] {0 | 1}<NL>
```

Example

```
DIM Mkr$[30]
OUTPUT 707;":DISP:MARK?"
ENTER 707;Mkr$
PRINT Mkr$
```

> It is a recommended practice to turn the selected marker on before attempting to set them using a :MEASure:{T | V} STARt or MEASure:{T | V} STOP command. Also see MARKer subsystem for more information on using markers.

13

FUNCtion Subsystem

Introduction

The FUNCtion subsystem defines functions using the displayed channels or waveform memories as operands. All available channels and waveform memories are available for functions.

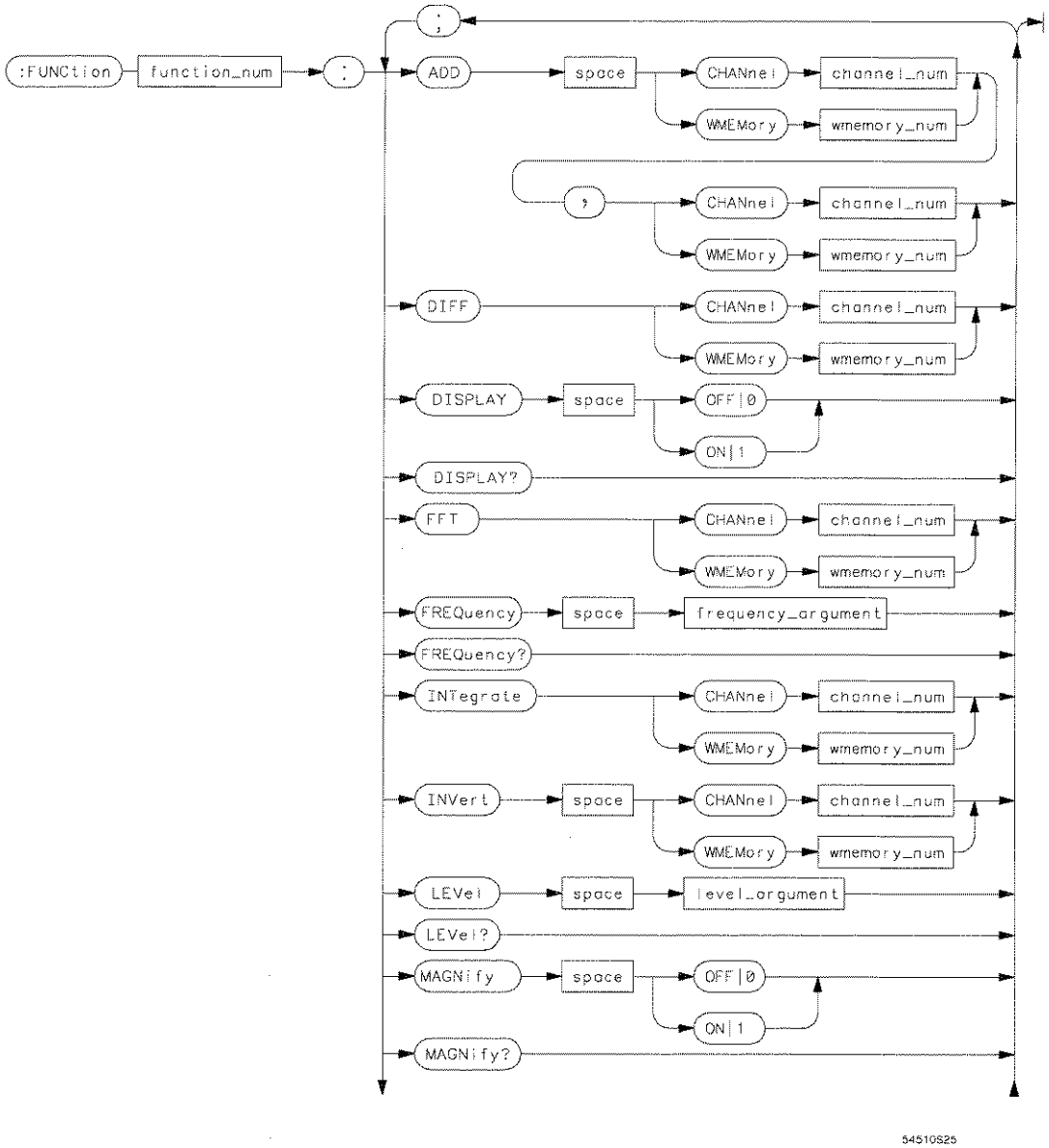
Functions are turned on/off with the VIEW/BLANk commands (see the "Root Level Commands" chapter), or by using the FUNCtion:DISPlay command (in this chapter).

Single valued functions (such as ADD, SUBTract, and FFT) may be transferred to a controller by first storing the function to a waveform memory, then transform that memory to the controller. Some of the commands apply only when the FFT function is selected. The Function subsystem contains the following commands:

ADD
DIFFerentiate
DISPlay
FFT (fast Fourier transform)
FREQuency (FFT function)
INTegrate
INVert
LEVel (FFT function)
MAGNify (FFT function)
MODE
MULTiply
OFFSet
ONLY
PEAK (FFT function)
POINts (FFT function)
RANGe
SETup (FFT function)
SPAN (FFT function)
SUBTract
VERSus
WINDow (FFT function)

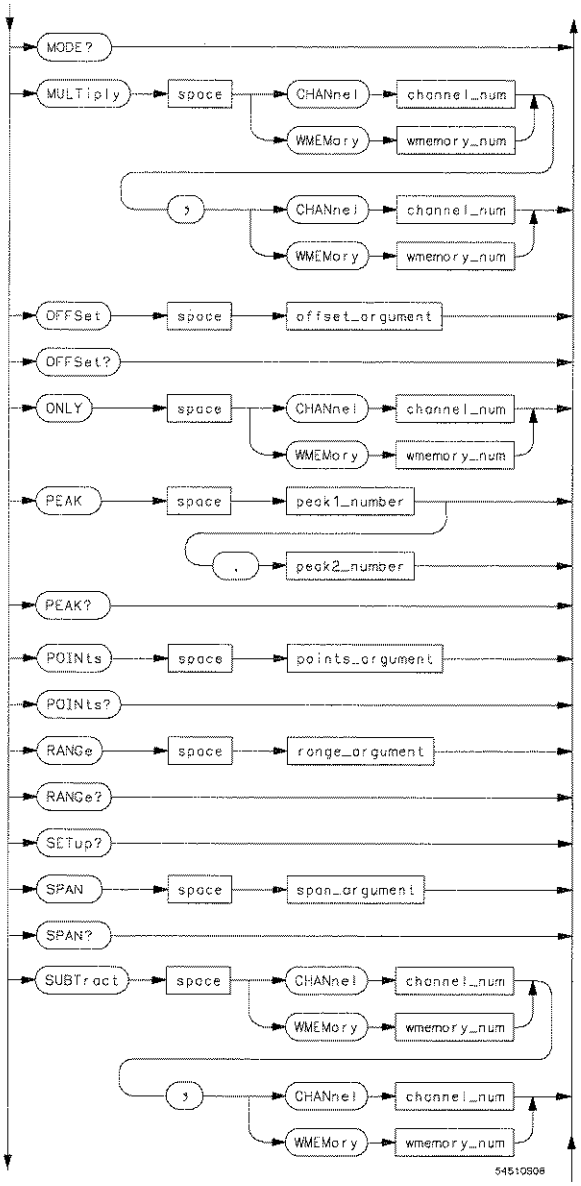
Figure 13-1 lists the syntax diagrams for the Function subsystem commands.

Figure 13–1



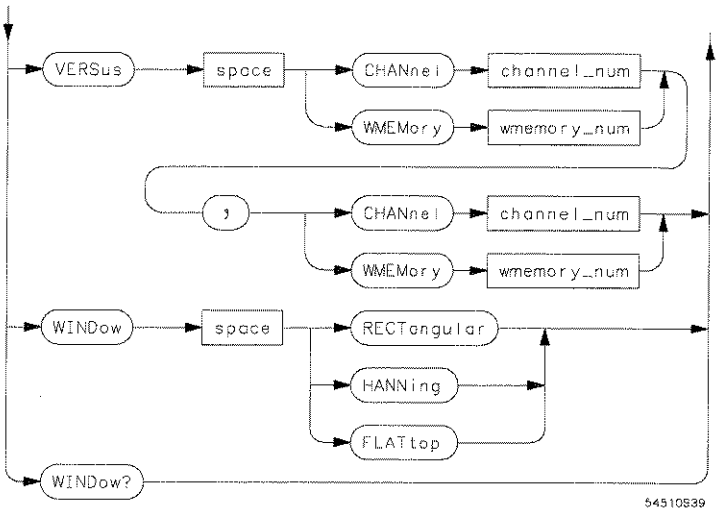
Function Subsystem Commands Syntax Diagram

Figure 13–1



Function Subsystem Commands Syntax Diagram (continued)

Figure 13–1



channel_num =	an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).
frequency_argument =	center frequency in Hz.
function_num =	an integer, 1 through 4.
level_argument =	0 to ±600 dBm.
offset_argument =	0 to ± voltage full scale, or 0 to ±200 dBm for FFT functions.
peak1_number =	1 to 99.
peak2_number =	1 to 99.
points_argument =	512, 1024, 2048, 4096, 8192, 16384, or 32768.
range_argument =	full screen voltage, or full screen dB for FFT functions.
span_argument =	frequency span in Hz.
wmemory_num =	an integer, 1 through 4.

Function Subsystem Commands Syntax Diagram (continued)

ADD

Command

```
:FUNCtion{1 | 2 | 3 | 4}:ADD <operand>,<operand>
```

The :FUNCtion{1 | 2 | 3 | 4}:ADD command algebraically adds the two defined operands.

<operand> ::= {CHANnel<n> | WMEMory {1 | 2 | 3 | 4}}

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:ADD WMEMORY3,WMEMORY4"
```

DIFF

Command

```
:FUNCtion{1 | 2 | 3 | 4}:DIFF <operand>
```

The :FUNCtion{1 | 2 | 3 | 4}:DIFF command computes the voltage differences between consecutive points in time divided by the time bucket width, Δt.

The differentiate function calculates the derivative of the designated operand with respect to time. Differentiation proceeds on a point-by-point basis. If a data point is not encountered in the operand, then differentiation uses the next valid data point.

$d_1 = 0$

$$d_n = \frac{c_{(n)} - c_{(n-1)}}{\Delta t}$$

The equation calculates the differential waveform of the operand, where d represents the differential waveform and c represents the operand voltage.

> The differential function will magnify signal noise. This noise will be less in the real-time mode.

`<operand>` ::= {CHANnel<n> | WMEMory {1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:DIFF WMEMORY4"
```

DISPlay

Command

```
:FUNCtion{1 | 2 | 3 | 4}:DISPlay
{{OFF | 0} | {ON | 1}}
```

The :FUNCtion{1 | 2 | 3 | 4}:DISPlay controls the individual function displays. On starts displaying, and off stops displaying the selected function.

Example

```
OUTPUT 707;":FUNCTION2:DISPLAY ON"
```

Query

```
:FUNCtion{1 | 2 | 3 |4}:DISPlay?
```

The DISPlay query returns the current setting of the command for the specified function.

Returned Format

```
[:FUNCtion{1 | 2 | 3 | 4}:DISPlay] {0 | 1}<NL>
```

Example

```
OUTPUT 707;":FUNC1:DISP?"
ENTER 707;Disp
PRINT Disp
```

This command is similar to the BLANk and VIEW commands.

FFT

Command

```
:FUNCtion{1 | 2 | 3 | 4}:FFT <operand>
```

The :FUNCtion:{1 | 2 | 3 | 4}:FFT computes the fast Fourier transform function of the specified channel or memory. The FFT takes the digitized time record of the specified channel and transforms it to the frequency domain.

When the function is turned on and the :FUNCtion:{1 | 2 | 3 | 4}:FFT command is sent to the oscilloscope, the FFT spectrum is plotted on the oscilloscope display as dBm versus frequency.

An example FFT program is on the demonstration disk as discussed in the "Example Programs" chapter.

`<operand>` ::= {CHANnel<n> | WMEMory {1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;"FUNCTION1:FFT CHANNEL1"
```

FREQuency

Command

```
:FUNCtion{1 | 2 | 3 | 4}:FREQuency
<frequency_argument>
```

This command is used when an FFT function is selected. See FFT command.

The :FUNCtion{1 | 2 | 3 | 4}:FREQuency command sets the center frequency when the FFT magnify mode is selected. See MAGNify command.

`<frequency_argument>` ::= center frequency from 0 Hz to 1.5X of frequency span (exponential - NR3 format)

Example

```
OUTPUT 707;"FUNCTION1:FREQUENCY 25E6"
```

Query

```
:FUNCtion{1 | 2 | 3 | 4}:FREQuency?
```

The FREQuency query returns the current center frequency setting for the specified function.

Returned Format

```
[FUNCtion{1 | 2 | 3 | 4}:FREQuency] <frequency_argument><NL>
```

`<frequency_argument>` ::= center frequency from 0 Hz to 1.5X of frequency span (exponential - NR3 format)

Example

```
OUTPUT 707;"FUNC2:FREQ?"
ENTER 707;Frq
PRINT Frq
```

INTegrate

Command

```
:FUNCtion{1 | 2 | 3 | 4}:INTegrate <operand>
```

The :FUNCtion{1 | 2 | 3 | 4}:INTegrate command computes the integral of the specified waveform. The integral is calculated by adding voltage points multiplied by the time bucket width, Δt.

The integrate function calculates the integral (with respect to time) of the vertical value of of the designated operand. Integration proceeds on a point-by-point basis. If a data point is not encountered in the operand, then differentiation uses the next valid data point.

$$I_n = \sum_{i=0}^{n-1} C_i \ \Delta t$$

The equation calculates the integral of the operand, where *I* represents the integral and *C* represents the operand data points.

`<operand>` ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:INTEGRATE WMEMORY4"
```

INVert

Command

```
:FUNCtion{1 | 2 | 3 | 4}:INVert <operand>
```

The :FUNCtion{1 | 2 | 3 | 4}:INVert command inverts the operand.

<operand> ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:INVERT WMEMORY3"
```

LEVel

Command

```
:FUNCtion{1 | 2 | 3 | 4}:LEVel <level_argument>
```

This command is used when an FFT function is selected. See FFT command. The :FUNCtion{1 | 2 | 3 | 4}:LEVel command sets the minimum search level for the peak search. The oscilloscope will search for peaks above the level specified with this command. To qualify as a peak, both positive and negative edges of the peak must be at least one vertical display division high (or one-half a display division in split screen), and be above the search level.

`<level_argument>` ::= level value 0 to ± 600 dBm (exponential - NR3 format)

Example

```
OUTPUT 707;"FUNCTION1:LEVEL -25"
```

Query

```
:FUNCtion{1 | 2 | 3 | 4}:LEVEL?
```

The LEVel query returns the current level setting for the specified function.

Returned Format

```
[FUNCtion{1 | 2 | 3 | 4}:LEVEL]<level_argument><NL>
```

`<level_argument>` ::= level value 0 to ± 600 dBm (exponential - NR3 format)

Example

```
OUTPUT 707;":FUNC1:LEV?"
ENTER 707;Lvl
PRINT Lvl
```

MAGNify

Command

```
:FUNCtion{1 |2 | 3 | 4}:MAGNIFY {{OFF|0} | {ON|1}}
```

This command is used when an FFT function is selected. See FFT command.

The :FUNCtion{1 | 2 | 3 | 4}:MAGNify command magnifies a particular frequency range of interest.

In the magnify mode, 500 points of the FFT are displayed. For POINts = 512, half of the 500 points displayed in the magnify mode are interpolated. When the magnify is off, a compression algorithm is used to compress all the FFT points into 500 points. The compression algorithm accurately displays peaks and most noise. However, low points in the noise may occasionally be missed.

Example

```
OUTPUT 707;"FUNCTION1:MAGNIFY ON"
```

Query

```
:FUNCtion{1 | 2 | 3 | 4}:MAGNIFY?
```

The MAGNify query returns the current magnification state for the specified function.

Returned Format

```
[:FUNCtion{1 | 2 | 3 | 4}:MAGNify] {0 | 1}<NL>
```

Example

```
OUTPUT 707;"FUNC1:MAGN?"
ENTER 707;Mfy
PRINT Mfy
```

MODE?

Query

```
:FUNCtion{1 | 2 | 3 | 4}:MODE?
```

The MODE query returns the currently set operation for the function specified. The first operand is returned for all operations. The second operand is returned only for the ADD, SUBTract, MULTiply, and VERSus operations.

Returned Format

```
:FUNCtion{1 | 2 | 3 | 4}:MODE]
<operation>,<operand>[,<operand>]
```

<operation> ::= {ADD | SUBTract | MULTiply | VERSus | ONLY | INVert | INTegrate | DIFF | FFT}

<operand> ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Fmode$[50]
Output 707;":FUNCTION3:MODE?"
Enter 707;Fmode$
Print Fmode$
```

MULTiply

Command

```
:FUNCtion{1 | 2 | 3 | 4}:MULTiply
<operand>,<operand>
```

The :FUNCtion{1 | 2 | 3 | 4}:MULTiply command algebraically multiplies the two defined operands.

`<operand>` ::= {CHANnel<n> | WMEMory {1 | 2 | 3 | 4}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:MULTIPLY CHANNEL1,CHANNEL2"
```

OFFSet

Command

```
FUNCtion{1 | 2 | 3 | 4}:OFFSet <offset_argument>
```

The :FUNCtion{1 | 2 | 3 | 4 }:OFFSet command sets the voltage or, if the FFT function is selected, the dBm value represented at the center of the screen for the selected function.

`<offset_argument>` ::= offset value of 0 to ± voltage full scale, or 0 to ± 200 dBm for FFT function (exponential - NR3 format)

Example

```
OUTPUT 707;":FUNCTION1:OFFSET 650E-4"
```

Query

```
:FUNCtion{1 | 2 | 3 | 4}:OFFSet?
```

The OFFSet query returns the current offset value for the selected function.

Returned Format

```
[:FUNCtion{1 | 2 | 3 | 4}:OFFSet] <offset_argument><NL>
```

`<offset_argument>` ::= offset value of 0 to ± voltage full scale, or 0 to ± 200 dBm for FFT function (exponential - NR3 format)

Example

```
OUTPUT 707;":FUNC2:OFFS?"
ENTER 707;Off
PRINT Off
```

ONLY

Command

```
:FUNCtion{1 | 2 | 3 | 4}:ONLY <operand>
```

The :FUNCtion{1 | 2 | 3 | 4 }:ONLY command produces another copy of the operand and places it in the specified function. The ONLY command is useful for scaling channels and memories with the :FUNCtion{1 | 2 | 3 | 4}:RANGe and :FUNCtion{1 | 2 | 3 | 4}:OFFSet commands.

> The :FUNCtion{1 | 2 | 3 | 4}:ONLY command functions the same as the front panel Magnify function.

`<operand>` ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:ONLY WMEMORY4"
```

PEAK

Command

```
:FUNCtion{1 | 2 | 3 | 4}:PEAK
<peak1_number>[,<peak2_number>]
```

This command is used when an FFT function of the oscilloscope is selected. See FFT command.

The :FUNCtion{1 | 2 | 3 | 4}:PEAK command causes a peak search to be performed. The peak search sets Vmarker1 (y1) and the start marker on the first peak specified and sets Vmarker2 (y2) and the stop marker on the second peak specified. Peaks from 1 to 99 can be specified.

The marker values in frequency and dBm are automatically displayed at the bottom of the oscilloscope screen. The difference in frequency and dBm between the two peaks is also displayed.

The marker values can be read over the bus using the VSTArt, VSTOp, VDELta, TSTArt, TSTOp, and TDELta queries. See the "Measure Subsystem" chapter.

If a peak is not found a message is displayed on the screen. The vertical marker is set to the right edge of the screen and the horizontal marker is set to the bottom of the screen.

For maximum frequency accuracy, the FFT horizontal magnify function should be turned on. See MAGNify command.

`<peak1_number>` ::= 1 through 99 (integer - NR1 format)

`<peak2_number>` ::= 1 through 99 (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION1:PEAK 5,7"
```

Query `:FUNCtion{1 | 2 | 3 | 4}:PEAK?`

The PEAK query returns the current peak(s) selected for the specified function.

Returned Format
```
[:FUNCtion{1 | 2 | 3 | 4}:PEAK]
<peak1_number>,<peak2_number><NL>
```

`<peak1_number>` ::= 1 through 99 (integer - NR1 format)

`<peak2_number>` ::= 1 through 99 (integer - NR1 format)

Example
```
OUTPUT 707;":FUNC1:PEAK?"
```

POINts

Command

```
:FUNCtion{1 | 2 | 3 |4}:POINts <points_argument>
```

This command is used when the FFT function is selected. See FFT command.

The :FUNCtion{1 | 2 | 3 | 4}:POINts <points_number> assigns the number of data points in the record to be used for the FFT computation. The number of points affects the computation speed, frequency resolution, and the noise floor. The frequency resolution of an FFT is $\frac{F_s}{N}$, where: F_s is the sampling frequency and N is the number points. The maximum signal-to-noise ratio for an FFT is also related to the number of points and the A/D converter bits of resolution.

An FFT performed on a small number of points can be computed faster. An FFT performed on a large number of points has better frequency resolution and a lower noise floor. The resolution affects how accurately frequencies can be measured and how well two frequencies that are close together can be resolved.

If the oscilloscope is in repetitive mode, or the operand is a memory that was stored from the repetitive mode, the number of points defaults to 512 and cannot be changed.

When the operand is a repetitive source, on-screen data is used for the FFT. When the operand is real time, the FFT is performed on data in the 8k record, but not necessarily data that is displayed on the screen. The points used from the source record for the FFT are based on the TIMebase reference setting of left, center, or right. See also :TIMebase:REFerence command.

When ":TIMebase:REFerence LEFT" is sent, data from the beginning of the record is used. When "TIMebase:REFerence CENTer" is sent, data from the middle of the record is used. When "TIMebase:REFerence RIGHt" is sent, data from the end of the record is used. When in the MATH menu, the memory bar above the graticule indicates the portion of the data record used for the FFT. The memory bar showing the FFT record is displayed whenever the FFT menu with the **# of points** key is on.

When it is necessary to window the data, the LEFT REFerence is a good choice since the FFT record then begins on the left side of the screen.

An FFT with an input record of *N* points becomes a frequency record of *N* points. Because half of the points are above the Nyquist frequency (sample frequency/2) and provide redundant information, they are not used so the number of frequency points are half the number of input points.

The FFTs are computed on records that are powers of two. If a repetitive source (500 points) or an entire record (8000 points) is selected, the record is zero padded at the ends to achieve 512 and 8192 points.

If the number of FFT points is 512, 256 additional frequency points are created by interpolating between the actual points to generate 512 frequency points. Table 13-1 lists the number of input points and the resultant number of frequency points.

Table 13-1 **FFT Points**

INPUT POINTS	FREQUENCY POINTS
512 (repetitive)	512
512 (real -time)	512
1024	512
2048	1024
4096	2048
8192	4096
16384	8192
32768	16384

`<points_argument>` ::= {512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768}

Example

```
OUTPUT 707;":FUNCTION1:POINTS 2048"
```

Query

```
:FUNCtion{1 | 2 | 3 |4}:POINts?
```

The POINts query returns the current points setting for the specified function.

Returned Format

```
[:FUNCtion{1 |2 | 3 | 4}:POINts] <points_argument><NL>
```

<points_argument> ::= {512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768}

Example

```
OUTPUT 707;"FUNC1:POIN?"
ENTER 707;Pts
PRINT Pts
```

RANGe

Command

```
:FUNCtion{1 | 2 | 3 | 4}:RANGe <range_argument>
```

The :FUNCtion{1 | 2 | 3 | 4 }:RANGe command defines the full scale vertical axis of the selected function.

`<range_argument>` ::= full scale vertical range in volts, dB, or dBm (exponential - NR3 format)

Example

```
OUTPUT 707;":FUNCTION2:RANGE 400E-3"
```

Query

```
:FUNCtion{1 | 2 | 3 | 4}:RANGe?
```

The RANGe query returns the current range setting for the specified function.

Returned Format

```
[:FUNCtion{1 | 2 | 3 | 4}:RANGe] <range_argument><NL>
```

`<range_argument>` ::= full scale vertical range in volts, dB, or dBm (exponential - NR3 format)

Example

```
OUTPUT 707;":FUNC2:RANG?"
ENTER 707;Rng
PRINT Rng
```

SETup?

Query

```
:FUNCtion{1 | 2 | 3 |4}:SETup?
```

The :FUNCtion{1 | 2 | 3 | 4}:SETup query returns the current settings for the FUNCtion Subsystem commands.

Returned Format

```
:FUNC{1 | 2 | 3 | 4}:DISP {0 | 1};
  {ADD | SUBT | MULT | VERS | ONLY | INV | INT | DIFF |
    FFT} {CHAN<n> | WMEM {1 | 2 | 3 | 4} [,{CHAN<n> |
    WMEM {1 | 2 | 3 | 4}];
  OFF <offset_argument>;
  RANG <range_argument>;
  FREQ <frequency_argument>; (FFT only)
  LEV <level_argument>; (FFT only)
  MAGN {0 | 1}; (FFT only)
  PEAK <peak1_number>,peak2_number>; (FFT only)
  POIN <points_argument>; (FFT only)
  SPAN <span_argument>; (FFT only)
  WIND {RECT | HANN | FLAT}; (FFT only)<NL>
```

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<offset_argument>` ::= offset value of 0 to ± voltage full scale, or 0 to ± 200 dBm for FFT function (exponential - NR3 format)

`<range_argument>` ::= full scale vertical range in volts, dB, or dBm (exponential - NR3 format)

`<frequency_argument>` ::= center frequency from 0 Hz to 1.5X of frequency span (exponential - NR3 format)

`<level_argument>` ::= level value 0 to ± 600 dBm (exponential - NR3 format)

`<peak1_number>` ::= 1 through 99 (integer - NR1 format)

`<peak2_number>` ::= 1 through 99 (integer - NR1 format)

`<points_argument>` ::= {512 | 1024 | 2048 | 4096 | 8192 | 16384 | 32768}

`<span_argument>` ::= number in Hertz (exponential - NR3 format)

Example

```
DIM Stp$[200]
OUTPUT 707;":FUNCTION1:SETUP?"
ENTER 707;Stp$
PRINT Stp$
```

SPAN

Command

```
:FUNCtion{1 | 2 | 3 | 4}:SPAN <span_argument>
```

This command is used when the FFT function is selected. See FFT command.

The :FUNCtion{1 | 2 | 3 | 4}:SPAN command controls the frequency span of the FFT record. Changing the span for an FFT function with a channel source causes the timebase to change. The span is the sample frequency divided by two. Since the sample frequency for memories is fixed once a record is stored, the span is also fixed and cannot be changed for FFT functions with a memory source.

<span_argument> ::= number in Hertz (exponential - NR3 format)

Example

```
OUTPUT 707;;"FUNCTION1:SPAN 25E6"
```

Query

```
:FUNCtion{1 | 2 | 3 | 4}:SPAN?
```

The SPAN query returns the current span setting for the specified function.

Returned Format

```
[:FUNCTION{1 | 2 | 3 | 4}:SPAN]<span_argument><NL>
```

<span_argument> ::= number in Hertz (exponential - NR3 format)

Example

```
OUTPUT 707:":FUNC1:SPAN?"
ENTER 707;Spn
PRINT Spn
```

SUBTract

Command

```
:FUNCtion{1 | 2 | 3 | 4}:SUBTract
<operand1>,<operand2>
```

The :FUNCtion{1 | 2 | 3 | 4}:SUBTract command algebraically subtracts operand 2 from operand 1.

`<operand1>` ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}

`<operand2>` ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:SUBTRACT WMEMORY3,WMEMORY2"
```

VERSus

Command

```
:FUNCtion{1 | 2 | 3 |4}:VERSus
<Y_operand>,<X_operand>
```

The :FUNCtion{1 | 2 | 3 |4}:VERSus command allows X vs Y displays with two operands. The first operand defines the Y-axis and the second defines the X-axis. The Y-axis range and offset is initially equal to that of the first operand, and can be adjusted with the FUNCtion{1 | 2 | 3 | 4}:RANGe and FUNCtion{1 | 2 | 3 | 4}:OFFSet commands.

The X-axis range and offset is always equal to that of the second operand. They can only be changed by changing the vertical settings of the second operand.

`<Y_operand>` ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}

`<X_operand>` ::= {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":FUNCTION2:VERSUS CHANNEL1,CHANNEL2"
```

WINDow

Command

```
:FUNCtion{1 | 2 | 3 |4}:WINDow {RECTangular |
HANNing | FLATtop}
```

This command is used when the FFT function is selected. See FFT command.

The :FUNCtion{1 | 2 | 3 |4}:WINDow command allows selection of three windows for the FFT function.

The FFT operation assumes that the time record repeats. Unless there is an integer number of cycles of the sampled waveform in the record, a discontinuity is created at the end of the record. This introduces additional frequency components into the spectrum about the actual peaks. This is referred to as leakage. In order to minimize leakage, windows that approach zero smoothly at the beginning and end of the record are employed as filters to the FFTs.

Each window is useful for certain classes of input signals.

The **rectangular** window is essentially no window, all points are multiplied by 1. The **rectangular** window is useful for transient signals and signals where there are an integral number of cycles in the time record. The **Hanning** window is useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together or for making frequency measurements. The **flattop** window is best for making accurate amplitude measurements of frequency peaks.

Example

```
OUTPUT 707;":FUNCTION<N>:WINDOW RECTANGULAR"
```

Query

```
:FUNCtion{1 | 2 | 3 |4}:WINDow?
```

The WINDow query returns the current window selected for the specified function.

Returned Format

```
[FUNCTION{1 | 2 | 3 | 4}:WINDow] {RECTangular | HANNing |
FLATtop}<NL>
```

Example

```
DIM Wnd$[50]
OUTPUT 707;":FUNC1:WIND?"
ENTER 707;Wnd$
PRINT Wnd$
```

14

HARDcopy Subsystem

Introduction

The HARDcopy subsystem commands set various parameters for printing and plotting waveforms from the oscilloscope.

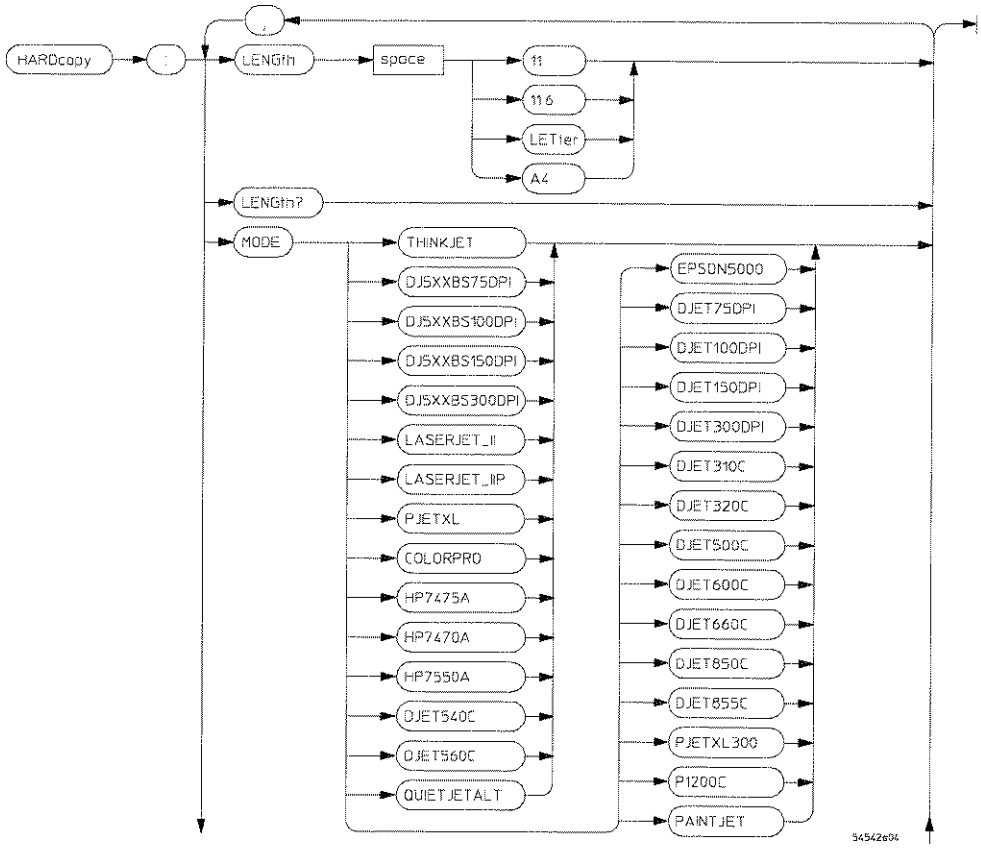
To actually make the hardcopy print or plot, refer to the root level commands :PRINt and :PLOT for the sequence of bus commands that actually get the data to the printer or plotter.

The Hardcopy subsystem contains the following commands:

LENGth
MODE
PAGE
PLOT:AREA
PLOT:INITialize
PLOT:{PEN | COLor}

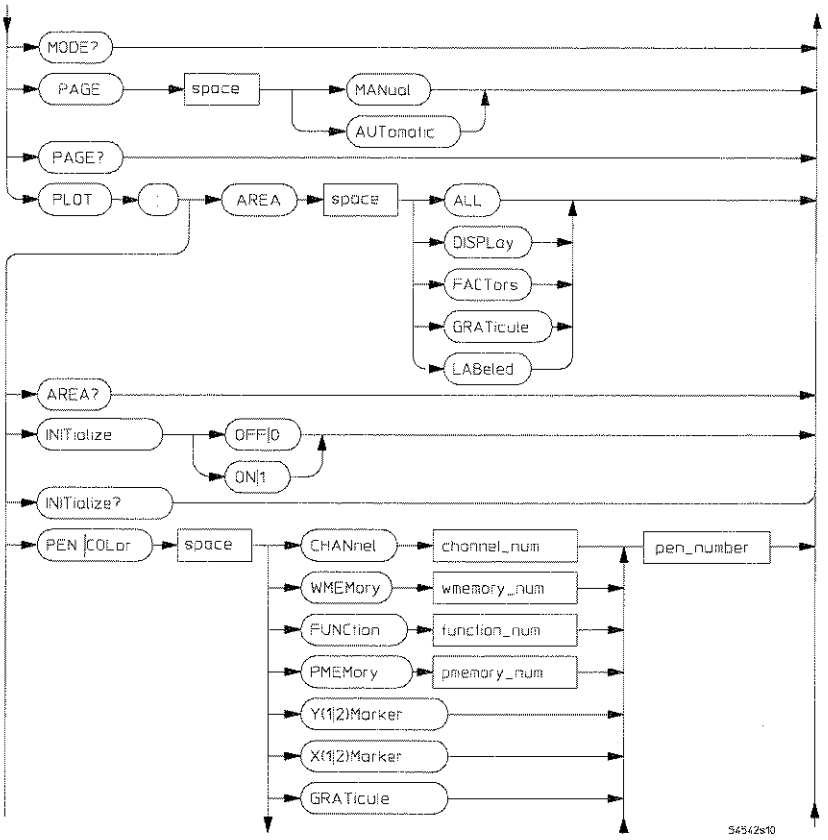
Figure 14-1 lists the syntax diagrams for the Hardcopy subsystem commands.

Figure 14–1



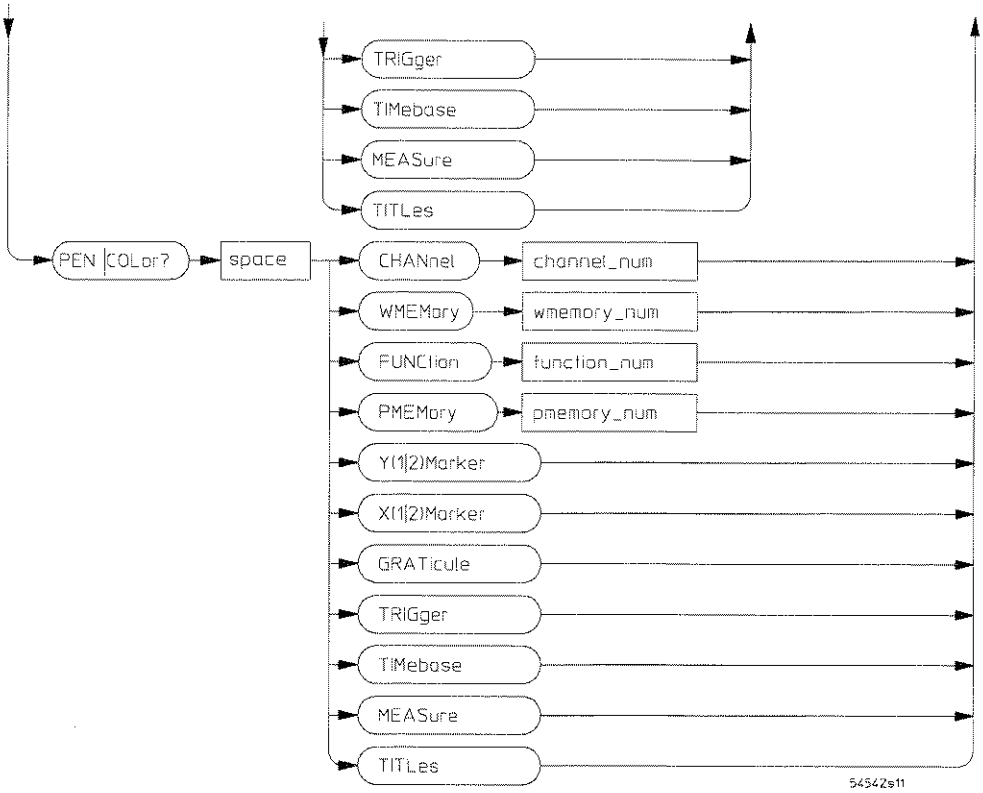
Hardcopy Subsystem Commands Syntax Diagram

Figure 14–1



Hardcopy Subsystem Commands Syntax Diagram (continued)

Figure 14–1



Hardcopy Subsystem Commands Syntax Diagram (continued)

Figure 14-1

channel_number =	an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).
function_num =	an integer, 1 to 4.
pen_number =	an integer, 0 to 8.
pmemory_num =	an integer, 1 or 2.
wmemory_num =	an integer, 1 to 4.

Hardcopy Subsystem Commands Syntax Diagram (continued)

LENGth

Command

```
:HARDcopy:LENGth {11 | 12}
```

The :HARDcopy:LENGth command sets the length of the page to either 11 inches or 12 inches.

Example

```
OUTPUT 707;":HARDCOPY:LENGTH 12"
```

Query

```
:HARDcopy:LENGth?
```

The LENGth query returns the current length setting.

Returned Format

```
[:HARDcopy:LENGth] {11 | 12}<NL>
```

Example

```
OUTPUT 707;":HARD:LENG?"
ENTER 707;Lgth
PRINT Lgth
```

MODE

Command

```
:HARDcopy:MODE {THINKJET | DJ5XBW75DPI |
DJ5XBW100DPI | DJ5XBW150DPI | DJ5XBW300DPI |
DJET75DPI | DJET100DPI | DJET150DPI |
DJET300DPI | LASERJET_II | LASERJET_IIP |
PJETXL300 | PJETXL | PAINTJET | COLORPRO |
HP7475A | HP7470A | HP7550A | DJET310C |
DJET320C | DJET500C | DJET540C | DJET560C |
DJET600C | DJET660C | DJET850C | DJET855C |
P1200C | QUIETJETALT | EPSON5000}
```

The :HARDcopy:MODE command sets the remote interface device mode for a printer or plotter output.

Example

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

Query

```
:HARDcopy:MODE?
```

The MODE query returns the current setting of the MODE command.

Returned Format

```
[:HARDcopy:MODE] {THINKJET | DJ5XBW75DPI |
DJ5XBW100DPI | DJ5XBW150DPI | DJ5XBW300DPI |
DJET75DPI | DJET100DPI | DJET150DPI |
DJET300DPI | LASERJET_II | LASERJET_IIP |
PJETXL300 | PJETXL | PAINTJET | COLORPRO |
HP7475A | HP7470A | HP7550A | DJET310C |
DJET320C | DJET500C | DJET540C | DJET560C |
DJET600C | DJET660C | DJET850C | DJET855C |
P1200C | QUIETJETALT | EPSON5000}<NL>
```

Example

```
DIM Md$[30]
OUTPUT 707;":HARD:MODE?"
ENTER 707;Md$
PRINT Md$
```

PAGE

Command

```
:HARDcopy:PAGE {MANual | AUTomatic}
```

The :HARDcopy:PAGE command configures the oscilloscope to send a formfeed to the printer after it outputs a hardcopy.

If the PAGE command is set to AUTomatic, a formfeed occurs at the end of the hardcopy; otherwise, the page scrolls up by 4 lines.

Example

```
OUTPUT 707;":HARDCOPY:PAGE AUTOMATIC"
```

Query

```
:HARDcopy:PAGE?
```

The PAGE query returns the current state of the PAGE command.

Returned Format

```
[:HARDcopy:PAGE] {MANual | AUTomatic}<NL>
```

Example

```
DIM Pg$[30]
OUTPUT 707;":HARD:PAGE?"
ENTER 707;Pg$
PRINT Pg$
```

PLOT:AREA

Command

```
:HARDcopy:PLOT:AREA {ALL | DISPlay | FACTors |
GRATicule | LABeled}
```

The :HARDcopy:PLOT:AREA command selects the area to be plotted.

Example

```
OUTPUT 707;":HARDCOPY:PLOT:AREA ALL"
```

Query

```
:HARDcopy:PLOT:AREA?
```

The PLOT:AREA query returns the current selection for the PLOT:AREA command.

Returned Format

```
[:HARDcopy:PLOT:AREA] {ALL | DISPlay | FACTors | GRATicule |
LABeled}<NL>
```

Example

```
DIM P1$[50]
OUTPUT 707;":HARD:PLOT:AREA?"
ENTER 707;P1$
PRINT P1$
```

PLOT:INITialize

Command

```
:HARDcopy:PLOT:INITialize {{OFF | 0} | {ON | 1}}
```

The :HARDcopy:PLOT:INITialize command sets the plotter to a known state. Sending the command :HARDcopy:PLOT:INITialize ON sends the "IN" command to the plotter which sets P1 and P2 to the default value. Sending the command :HARDcopy:PLOT:INITialize OFF sends the "DF" command to the plotter and does not set P1 and P2 to the default value.

Example

```
OUTPUT 707;":HARDCOPY:PLOT:INITIALIZE ON"
```

Query

```
:HARDcopy:PLOT:INITialize?
```

The PLOT:INITialize query returns the current setting of the command.

Returned Format

```
[:HARDcopy:PLOT:INITialize] {0 | 1}<NL>
```

Example

```
OUTPUT 707;":HARD:PLOT:INIT?"
ENTER 707;P1
PRINT P1
```

PLOT:{PEN|COLor}

Command

```
:HARDcopy:PLOT:{PEN|COLor} <item>,<pen_number>
```

The :HARDcopy:PLOT:PEN|COLor command selects the plotter pen to plot individual items on the screen. By selecting the appropriate plotter pen, you can plot different items on the screen in different colors to distinguish them from each other. Pen/color numbers 1 through 8 select the corresponding pens on the plotter. Pen/color number 0 returns the plotter pen to its carousel without plotting the selected item.

`<item>` ::= {CHANnel<n> | MEMory{1 | 2 | 3 | 4} | FUNCtion{1 | 2 | 3 | 4} | PMEMory{1 | 2} | Y{1|2}Marker | X{1|2}Marker | GRATicule | TRIGger | TIMebase | MEASure | TITLes}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<pen_number>` ::= 0 through 8 (integer - NR1 format)

Example

```
OUTPUT 707;":HARDCOPY:PLOT:COLOR CHANNEL1,2"
```

Query

```
:HARDcopy:PLOT:{PEN|COLor}? <item>
```

The PLOT:PEN|COLor query returns the current pen/color selection for an individual item on the screen.

Returned Format

```
[:HARDcopy:PLOT:{PEN|COLor}] <pen_number><NL>
```

<item> ::= {CHANnel<n> | MEMory{1 | 2 | 3 | 4} | FUNCtion{1 | 2| 3 | 4} | PMEMory{1 | 2} | Y{1|2}Marker | X{1|2}Marker | GRATicule | TRIGger | TIMebase | MEASure | TITLes}

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

<pen_number> ::= 0 through 8 (integer - NR1 format)

Example

```
OUTPUT 707;":HARD:PLOT:COL? CHAN1"
ENTER 707;P1
PRINT P1
```

15

MARKer Subsystem

Introduction

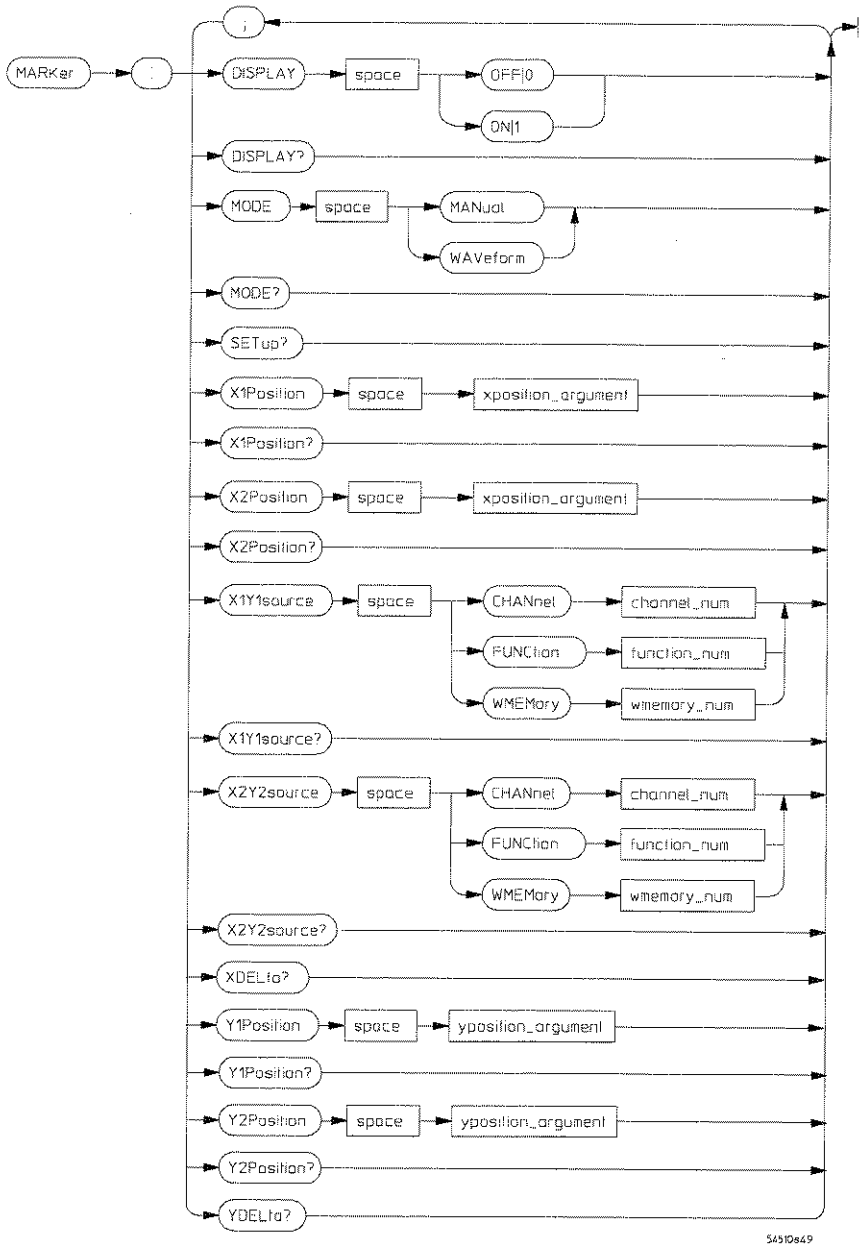
The MARKer subsystem commands control the two x-axis (vertical) and y-axis (horizontal) markers. Each set of markers are independently programmable for source and position.

The Marker subsystem contains the following commands:

DISPlay
MODE
SETup
X1Position
X2Position
X1Y1source
X2Y2source
XDELta
Y1Position
Y2Position
YDELta

Figure 15-1 lists the syntax diagrams for the Marker subsystem commands.

Figure 15–1



Marker Subsystem Commands Syntax Diagram

Figure 15-1

xposition_argument =	time in seconds from the trigger.
yposition_argument =	a real number within the voltage range.
channel_number =	an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).
function_number =	an integer, 1 through 4.
wmemory_number =	an integer, 1 through 4.

Marker Subsystem Commands Syntax Diagram (continued)

DISPlay

Command

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

The :MARKer:DISPlay command controls the marker displayed on the screen. On displays x/y markers, menus, and current position and statistic data on the screen.

Example

```
OUTPUT 707;":MARKER:DISPLAY ON"
```

Query

```
:MARKer:DISPlay?
```

The DISPlay query returns the current setting of the command.

Returned Format

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

Example

```
OUTPUT 707;":MARK:DISP?"
ENTER 707;Disp
PRINT Disp
```

MODE

Command

```
:MARKer:MODE {MANual | WAVeform}
```

The :MARKer:MODE command controls the operation of the marker displayed on the screen.

Example

```
OUTPUT 707;":MARKER:MODE WAVeform"
```

Query

```
:MARKer:MODE?
```

The MODE query returns the current setting of the command.

Returned Format

```
[:MARKer:MODE] {MANual | WAVeform}<NL>
```

Example

```
DIM Disp$[50]
OUTPUT 707;":MARK:MODE?"
ENTER 707;Disp$
PRINT Disp$
```

SETup?

Query

```
:MARKer:SETup?
```

The :MARKer:SETup query returns the current settings for the MARKer Subsystem commands.

Returned Format

```
:MARK:MODE {MAN | WAV};DISP {0 | 1};
  XDEL <xdelta>;
  X1P <xposition_argument>;
  X2P <xposition_argument>;
  X1Y1 {CHAN<n> | FUNC{1 | 2 | 3 | 4} | WMEM{1 | 2 | 3 | 4}};
  X2Y2 {CHAN<n> | FUNC{1 | 2 | 3 | 4} | WMEM{1 | 2 | 3 | 4}};
  Y1P <yposition_argument>;        (MAN mode only)
  Y2P <yposition_argument>         (MAN mode only)
  YDEL <ydelta>                    (MAN mode only)
  Y1P <yposition_argument>;        (MAN mode only)
  Y2P <yposition_argument><NL>     (MAN mode only)
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

<xdelta> ::= difference in seconds between x1 and x2 markers (exponential - NR3 format)

<xposition_argument> ::= xmarker in seconds or hertz (exponential - NR3 format)

<ydelta> ::= difference in volts between y1 and y2 markers (exponential - NR3 format)

<yposition_argument> ::= ymarker in volts or power (exponential - NR3 format)

Example

```
DIM Stp$[300]
OUTPUT 707;":MARKER:SETUP?"
ENTER 707;Stp$
PRINT Stp$
```

X1Position

Command

```
:MARKer:X1Position <xposition_argument>
```

The :MARKer:X1Position command moves the x1marker to the specified time with respect to the trigger time. Positive values set the marker to the right of the trigger point (post-trigger), while negative values set the marker to the left of the trigger point (pre-trigger). Trigger point is set using the TIMebase:REFerence command.

`<xposition_argument>` ::= xmarker time in seconds (all modes except FFT)
::= frequency in hertz (FFT mode) (exponential - NR3 format)

Example

```
OUTPUT 707;":MARKER:X1POSITION 30 NS"
```

Query

```
:MARKer:X1Position?
```

The X1Position query returns the position of the x1marker. Time is returned in seconds for all modes except FFT, where frequency is returned in hertz.

Returned Format

```
[:MARKer:X1Position] <xposition_argument><NL>
```

`<xposition_argument>` ::= xmarker time in seconds
::= frequency in hertz (exponential - NR3 format)

Example

```
OUTPUT 707;":MARK:X1P?"
ENTER 707;X1p
PRINT X1p
```

This command is identical to the MEASure:TSTArt command, and is provided for compatibility with other oscilloscopes.

X2Position

Command

```
:MARKer:X2Position <xposition_argument>
```

The :MARKer:X2Position command moves the x2marker to the specified time with respect to the trigger time. Positive values set the marker to the right of the trigger point (post-trigger), while negative values set the marker to the left of the trigger point (pre-trigger). Trigger point is set using the TIMebase:REFerence command.

`<xposition_argument>` ::= xmarker time in seconds (all modes except FFT)
::= frequency in hertz (FFT mode) (exponential - NR3 format)

Example

```
OUTPUT 707;":MARKER:X2POSITION 40 NS"
```

Query

```
:MARKer:X2Position?
```

The X2Position query returns the position of the x1marker. Time is returned in seconds for all modes except FFT, where frequency is returned in hertz.

Returned Format

```
[:MARKer:X2Position] <xposition_argument><NL>
```

`<xposition_argument>` ::= xmarker time in seconds
::= frequency in hertz (exponential - NR3 format)

Example

```
OUTPUT 707;":MARK:X2P?"
ENTER 707;X2p
PRINT X2p
```

This command is identical to the MEASure:TSTOp command, and is provided for compatibility with other oscilloscopes.

X1Y1source

Command

```
:MARKer:X1Y1source {CHANnel<n> |
FUNCtion{1|2|3|4} | WMEMory{1|2|3|4}}
```

The :MARKer:X1Y1source command selects the channel, function, or waveform memory to be used for the source of the X1 and Y1 marker commands.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MARKER:X1Y1SOURCE CHANNEL1"
```

Query

```
:MARKer:X1Y1source?
```

The X1Y1source query returns the currently selected source for the X1 and Y1 marker commands.

Returned Format

```
[:MARKer:X1Y1source] {CHANnel<n> | FUNCtion{1|2|3|4} |
WMEMory{1|2|3|4}}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[30]
OUTPUT 707;":MARK:X1Y1?"
ENTER 707;Src$
PRINT Src$
```

X2Y2source

Command

```
:MARKer:X2Y2source {CHANnel<n> |
FUNCtion{1|2|3|4} | WMEMory{1|2|3|4}}
```

The :MARKer:X2Y2source command selects the channel, function, or waveform memory to be used for the source of the X2 and Y2 marker commands.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MARKER:X2Y2SOURCE CHANNEL1"
```

Query

```
:MARKer:X2Y2source?
```

The X2Y2source query returns the currently selected source for the X2 and Y2 marker commands.

Returned Format

```
[:MARKer:X2Y2source] {CHANnel<n> | FUNCtion{1|2|3|4} |
WMEMory{1|2|3|4}}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[30]
OUTPUT 707;":MARK:X2Y2?"
ENTER 707;Src$
PRINT Src$
```

XDELta?

Query

```
:MARKer:XDELta?
```

The :MARKer:XDELta query returns the time or frequency difference between the x1 and x2 markers:

xdelta = x2marker - x1marker

Time is returned in seconds for all modes except FFT, where frequency is returned in hertz. If the returned value is negative, x1marker is located at a point previous to the x2marker.

Returned Format

```
[:MARKer:XDELta] <xdelta><NL>
```

<xdelta> ::= difference in seconds between x1 and x2 markers (exponential - NR3 format)

Example

```
OUTPUT 707;":MARKER:XDELTA?"
ENTER 707;Xdl
PRINT Xdl
```

This command is identical to the MEASure:TDELta command, and is provided for compatibility with other oscilloscopes.

Y1Position (Command ignored in waveform mode)

Command

```
:MARKer:Y1Position <yposition_argument>
```

The :MARKER:Y1Position command moves the y1marker to the specified level. The acceptable values are limited to the currently defined channel, function, or waveform memory range.

`<yposition_argument>` ::= ymarker voltage (all modes except FFT)
::= power in dBm (FFT mode) (exponential - NR3 format)

Example

```
OUTPUT 707;":MARKER:Y1POSITION -10MV"
```

Query

```
:MARKer:y1Position?
```

The Y1Position query returns the position of the y1marker. Level is returned in volts for all modes except FFT, where power is returned in dBm.

Returned Format

```
[:MARKer:Y1Position] <yposition_argument><NL>
```

`<yposition_argument>` ::= ymarker level in volts
::= power in dBm (exponential - NR3 format)

Example

```
OUTPUT 707;":MARK:Y1P?"
ENTER 707;X1p
PRINT Y1p
```

This command is identical to the MEASure:VSTArt command, and is provided for compatibility with other oscilloscopes.

Y2Position (Command ignored in waveform mode)

Command

```
:MARKer:Y2Position <yposition_argument>
```

The :MARKer:Y2Position command moves the y2marker to the specified level. The acceptable values are limited to the currently defined channel, function, or waveform memory range.

`<yposition_argument>` ::= ymarker voltage (all modes except FFT)
::= power in dBm (FFT mode) (exponential - NR3 format)

Example

```
OUTPUT 707;":MARKER:Y2POSITION -10MV"
```

Query

```
:MARKer:Y2Position?
```

The Y2Position query returns the position of the y2marker. Level is returned in volts for all modes except FFT, where power is returned in dBm.

Returned Format

```
[:MARKer:Y2Position] <yposition_argument><NL>
```

`<yposition_argument>` ::= ymarker level in volts
::= power in dBm (exponential - NR3 format)

Example

```
OUTPUT 707;":MARK:Y2P?"
ENTER 707;Y2p
PRINT Y2p
```

This command is identical to the MEASure:VSTOp command, and is provided for compatibility with other oscilloscopes.

YDELta?

Query

```
:MARKer:YDELta?
```

The :MARKer:YDELta query returns the voltage or power difference between the y1 and y2 markers:

ydelta = y2marker - y1marker

Level is returned in volts for all modes except FFT, where power is returned in dBm. If the returned value is negative, the y1marker is located at a lower level than the y2marker.

Returned Format

```
[:MARKer:YDELta] <ydelta><NL>
```

<ydelta> ::= difference in volts between y1 and y2 markers (exponential - NR3 format)

Example

```
OUTPUT 707;":MARKER:YDELTA?"
ENTER 707;Yd1
PRINT Yd1
```

This command is identical to the MEASure:VDELta command, and is provided for compatibility with other oscilloscopes.

16

MEASure Subsystem

Introduction

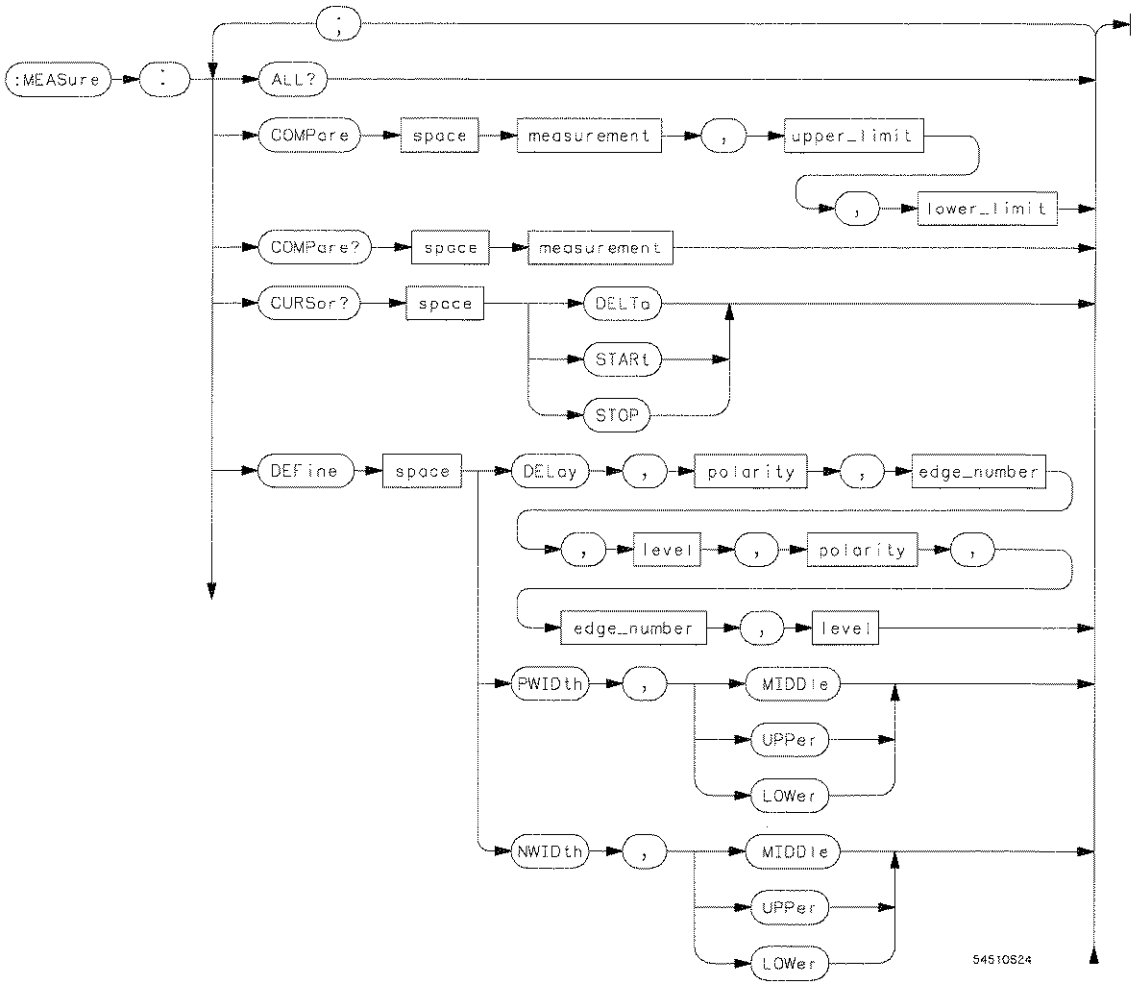
The commands in the MEASure subsystem are used to make parametric measurements on displayed waveforms and to report the settings of the voltage and time markers. Some commands in this subsystem can be used to set the voltage and time markers to specified voltages, times, or events.

The Measure subsystem contains the following commands:

ALL
COMPare
CURSor
DEFine
DELay
DESTination
DUTycycle
ESTArt
ESTOp
EANalysis
FALLtime
FREQuency
LIMittest
LOWer
MODE
MWINdow
NWIDth
OVERshoot
PERiod
POSTfailure
PRESHoot
PWIDth
RESults
RISetime
SCRatch
SOURce
STATistics
TDELta
TMAX
TMIN
TSTArt
TSTOp
TVOLt
UNITs
UPPer
VACRms
VAMPlitude
VAVerage
VBASe
VDCRms
VDELta
VFIFty
VMAX
VMIN
VPP
VRElative
VSTArt
VSTOp
VTIMe
VTOP
WCOMpare

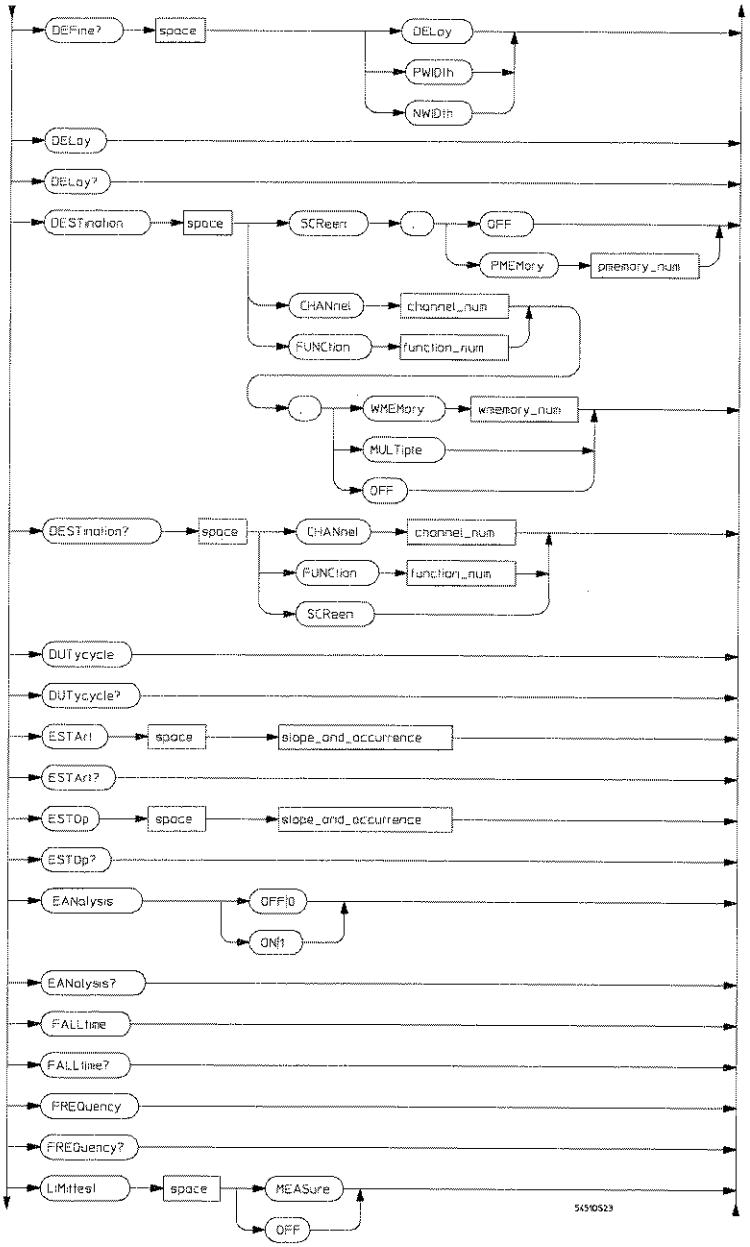
Figure 16-1 lists the syntax diagrams for the Measure subsystem commands.

Figure 16–1



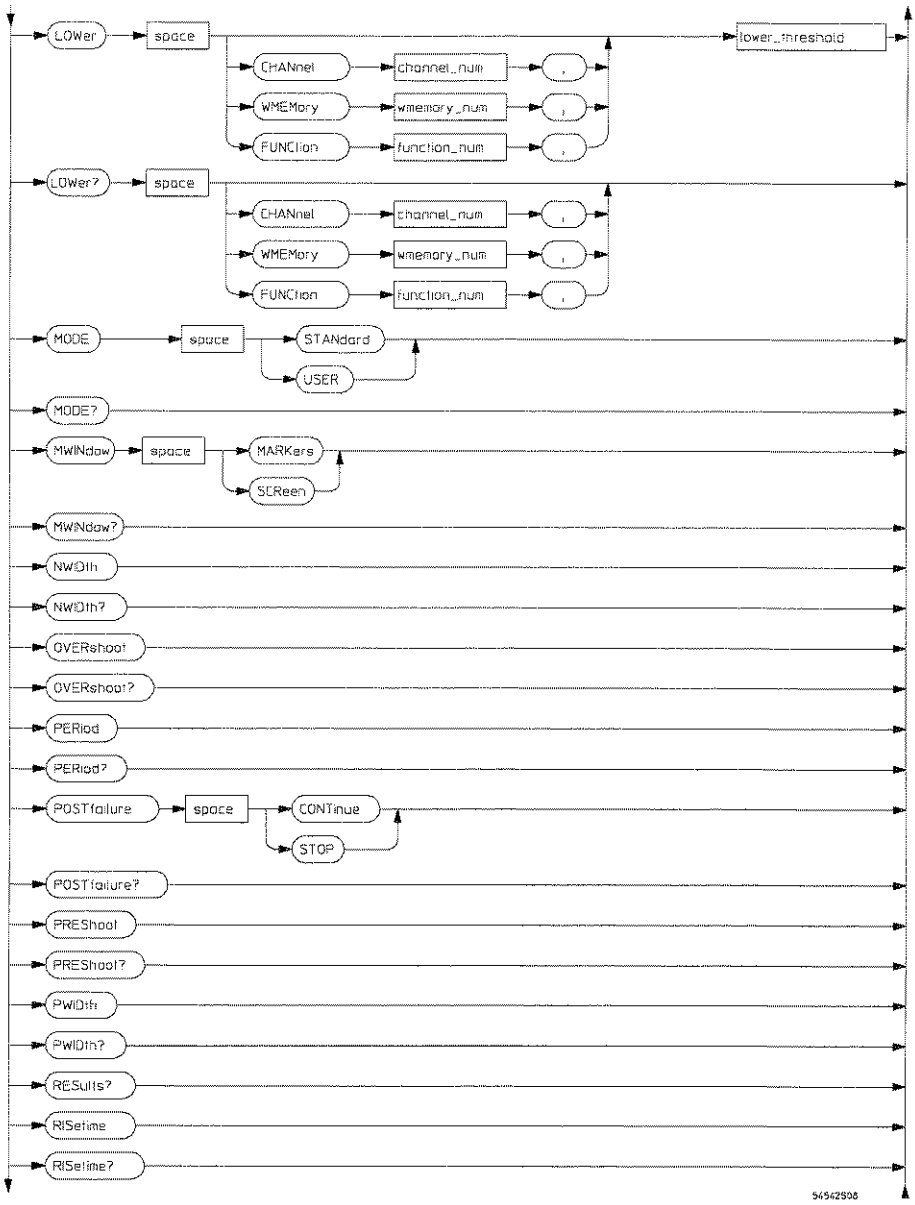
Measure Subsystem Commands Syntax Diagram

Figure 16–1



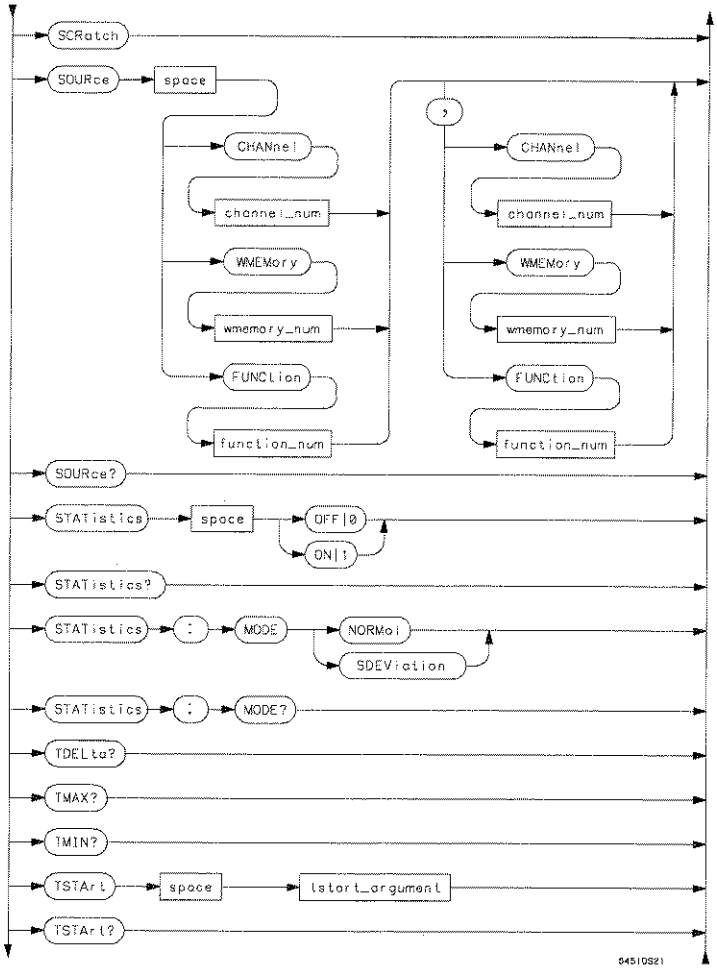
Measure Subsystem Commands Syntax Diagram (continued)

Figure 16–1



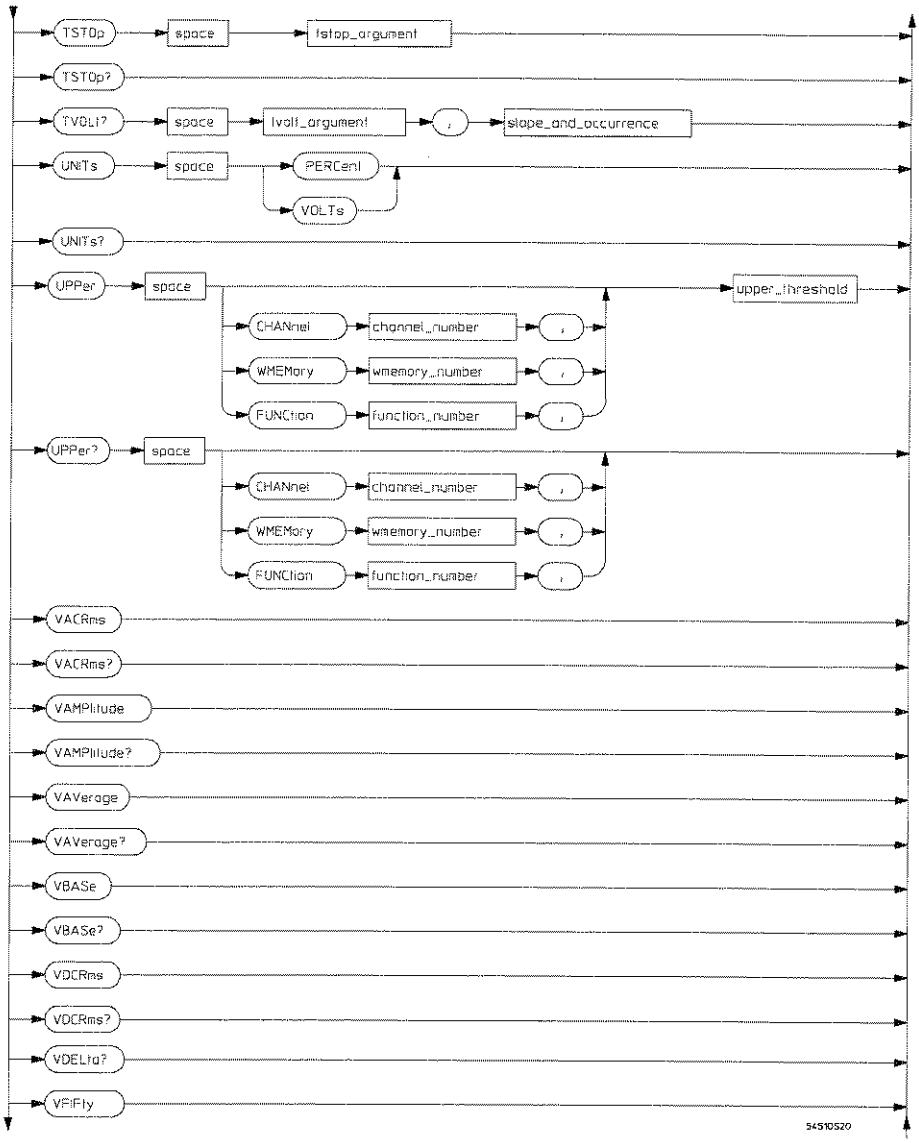
Measure Subsystem Commands Syntax Diagram (continued)

Figure 16–1



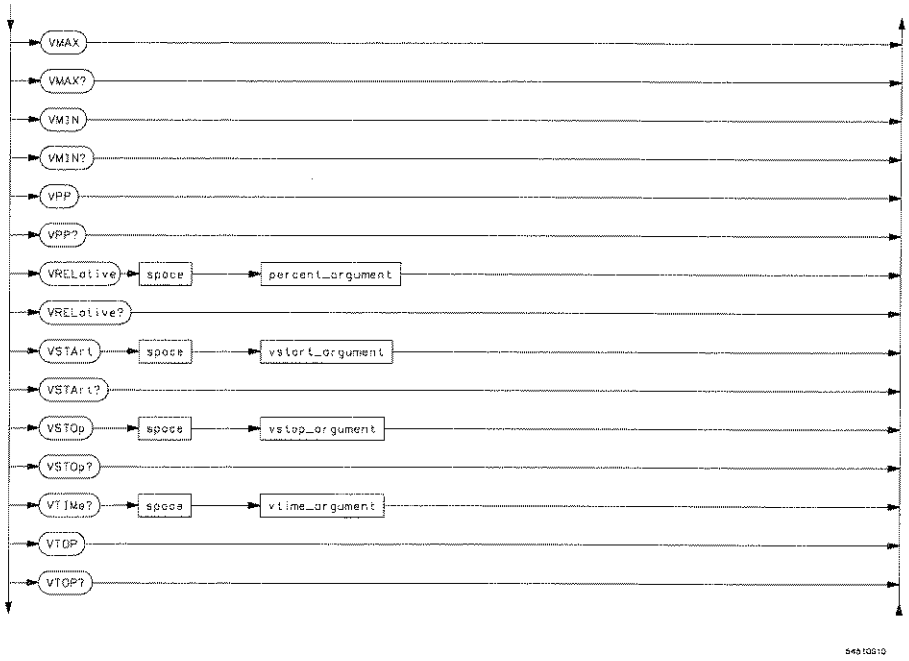
Measure Subsystem Commands Syntax Diagram (continued)

Figure 16–1



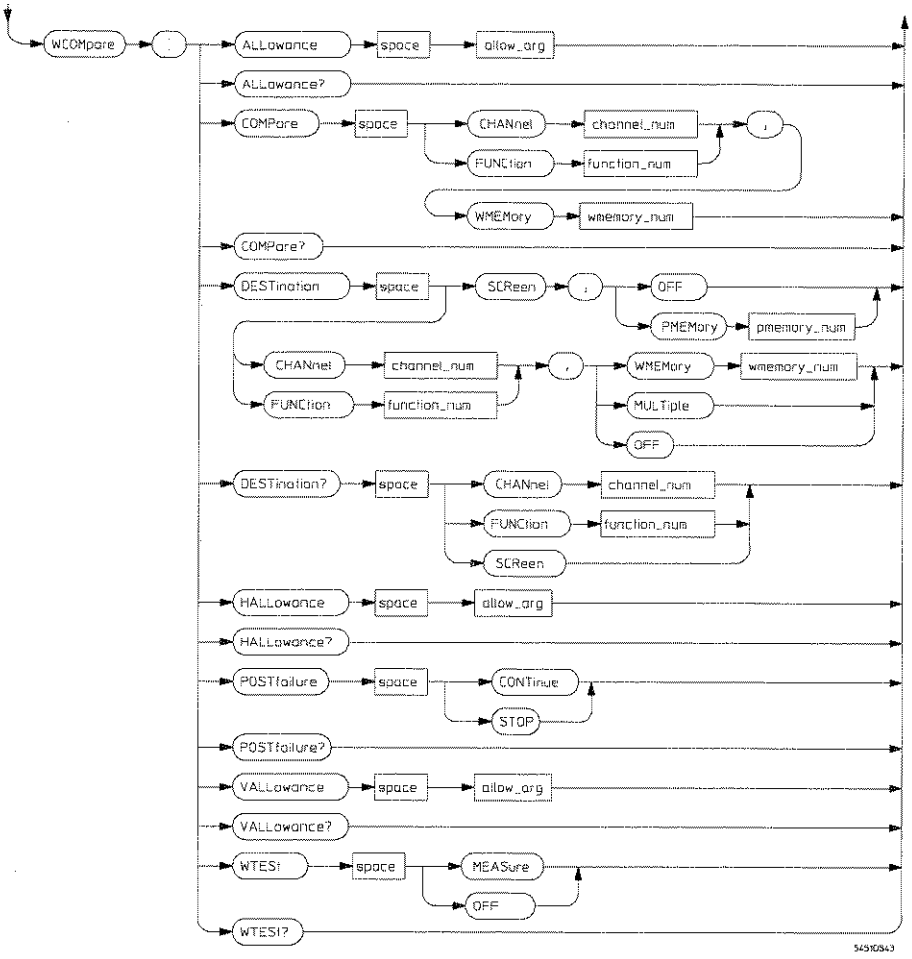
Measure Subsystem Commands Syntax Diagram (continued)

Figure 16–1



Measure Subsystem Commands Syntax Diagram (continued)

Figure 16–1



Measure Subsystem Commands Syntax Diagram (continued)

Figure 16-1

allow_arg =	real number
channel_num =	an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).
edge_number =	an integer, 1 to 4000.
function_num =	an integer , 1 through 4.
level =	MIDDle, UPPer, or LOWer.
lower_limit =	lower limit for compare.
lower_threshold =	lower threshold value in percent or volts.
measurement =	name of the measurement to be compared.
percent_argument =	an integer, 0 to 100.
pmemory_num =	an integer, 1 or 2.
polarity =	POSitive or NEGative.
slope_and_occurrence =	an integer, -4000 to 4000 (excluding 0) specifying a displayed edge.
tstart_argument =	time in seconds from the trigger.
tstop_argument =	time in seconds from the trigger.
tvolt_argument =	a real number specifying voltage.
upper_limit =	upper limit for compare.
upper_threshold =	upper threshold value in percent or volts.
vstart_argument =	a real number within the voltage range.
vstop_argument =	a real number within the voltage range.
vtime_argument =	a real number in the horizontal display window.
wmemory_num =	an integer, 1 through 4.

Measure Subsystem Commands Syntax Diagram (continued)

Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must be displayed on the oscilloscope:

For a period or frequency measurement, at least one complete cycle must be displayed.

For a pulse width measurement, the entire pulse must be displayed.

For a rise time measurement, the leading (positive-going) edge of the waveform must be displayed.

For a fall time measurement, the trailing (negative-going) edge of the waveform must be displayed.

User-Defined Measurements

When User-Defined Measurements are made, the defined parameters must be set before actually sending the measurement command or query. In addition, the user defined mode must be selected using the MEASure:MODE command.

Measurement Error

If the DIGitize command was not used to acquire data after configuring the oscilloscope, or if the oscilloscope was not given sufficient time to acquire data, there may not be enough waveform data available to make a valid measurement. In this case, the error value of 9.9E+37 is returned.

If the signal cannot be analyzed, the error value will also be returned. For example, if the signal is clipped, a Vpp measurement cannot be made. Or, if a whole period of the signal is not display, a frequency measurement cannot be made. In both cases, the error value of 9.9E+37 will be returned.

Making Measurements

If more than one waveform, edge, or pulse is displayed, time measurements are made on the first (left-most) portion of the displayed waveform.

When any of the defined measurements are requested, the oscilloscope first determines the top (100%) and base (0%) voltages of the waveform. From this information, it can determine the other important voltage values (10%, 90%, and 50% voltage values) for making measurements.

The 10% and 90% voltage values are used in the rise time and fall time measurements when standard measurements are selected. The 50% voltage value is used for measuring frequency, period, pulse width, and duty cycle with standard measurements selected.

The measurements can also be made using user-defined parameters instead of the standard measurement values.

When the command form of the measurement or front-panel measurements are used, the instrument is placed into the continuous measurement mode. When the query form of these measurements is used, the continuous measurement mode is turned off. Then the measurement is made one time, the measurement result is returned, and markers are positioned on that measurement.

Except for VAVerage, VRMS, VACRms, and VDCRms measurements, which use only one cycle, voltage measurements are made using the entire display. Therefore, if you want to make a measurement on a particular cycle, display only that cycle on the screen.

All voltage values are returned in volts. Returned voltage values are measured with zero volts as the reference. The value returned for the VDELta? query is the difference between y2marker and y1marker in volts.

All time values are returned in seconds. Returned time values are measured with the trigger point (time 0) as the reference. The value returned for TDELta? is the time difference between the x2marker and x1marker.

Measurements are made on the displayed waveforms specified by the SOURce command (except LOWer and UPPer measurements where a separate source can be specified). The SOURce command allows two sources to be specified. When two sources are specified, y1marker is assigned to the first specified source and y2marker is assigned to the second specified source.

Most measurements can only be made on a single source. If one of these measurements is made with two sources specified, the measurement is made on the first source specified.

If the horizontal scaling is questionable, an error 11, "Questionable horizontal scaling," is placed in the error queue. In this case, the value returned is the most accurate value that can be made using the current scaling. You might be able to obtain a more accurate measurement by rescaling the horizontal to obtain more data points on the edge.

For more information about measurement algorithms, refer to the "Algorithms" chapter.

ALL?

Query

```
:MEASure:ALL?
```

The :MEASure:ALL query makes a set of measurements on the displayed signal and buffers the measurement results for output over the Remote Interface.

To make a measurement, the portion of the waveform required for the measurement must be displayed. Time measurements are made on the first (left-most) displayed edges of the waveforms. To obtain the most accurate measurement possible, use proper horizontal scaling.

Refer to the individual commands for information on how the measurements are made and how the measurement results are returned.

Returned Format

```
[:MEASure][:FREQuency] <result>;[:PERiod]
   <result>;[:PWIDth] <result>;[:NWIDth] <result>;[:RISetime]
   <result>;[:FALLtime] <result>;[:VAMPlitude]
   <result>;[:VPP] <result>;[:PREShoot] <result>;[:OVERshoot]
   <result>;[:DUTycycle] <result>;[:VACRms] <result>;[:VMAX]
   <result>;[:VMIN] <result>;[:VTOP] <result>;[:VBASe]
   <result>;[:VAVerage] <result>;[:VDCRms] <result>; <NL>
```

`<result>` ::= individual measurement results (exponential - NR3 format)

Example

```
DIM A$[500]
OUTPUT 707;":SYSTEM:HEADER ON"
OUTPUT 707;":MEASURE:ALL?"
ENTER 707;A$
PRINT A$
```

COMPare

Command

```
:MEASure:COMPare
<measurement>,<upper_limit>,<lower_limit>
```

The :MEASure:COMPare command specifies the measurement and limits to be used for the measurement comparison (limit test). The first limit is the upper limit, the second is the lower limit.

This command does not start the test, but only sets the test parameters.

> The suffix "HZ" can be used when setting the limits for FREQuency.

`<measurement>` ::= {DELay | DUTycycle | FALLtime | FREQuency | NWIDth | OVERshoot | PERiod | PREShoot | PWIDth | RISetime | VACRms | VAMPlitude | VAVerage | VBASe | VDCRms | VMAX | VMIN | VPP | VTOP }

`<upper_limit>` ::= high limit value (exponential - NR3 format).

`<lower_limit>` ::= low limit value (exponential - NR3 format).

Example

```
OUTPUT 707;":MEASURE:COMPARE RISETIME,10ns,2ns"
```

Query

```
:MEASure:COMPare? <measurement>
```

The COMPare query returns the current specification.

Returned Format

```
[:MEASure:COMPare] <measurement>,<upper_limit>,<lower_
limit><NL>
```

<measurement> ::= {DELay | DUTycycle | FALLtime | FREQuency | NWIDth | OVERshoot | PERiod | PREShoot | PWIDth | RISetime | VACRms | VAMPlitude | VAVerage | VBASe | VDCRms | VMAX | VMIN | VPP | VTOP }

<upper_limit> ::= high limit value (exponential - NR3 format).

<lower_limit> ::= low limit value (exponential - NR3 format).

Example

```
DIM Cmp$[50]
OUTPUT 707;":MEAS:COMP? VPP"
ENTER 707;Cmp$
PRINT Cmp$
```

For example, the sequence required to do a limit test on frequency is:

Example

```
OUTPUT 707;":MEASURE:SCRATCH" !clear measurements
OUTPUT 707;":MEASURE:FREQ"   !Select measurement
OUTPUT 707;":MEASURE:COMPARE FREQ,1000HZ,10HZ"   !Set meas limits
OUTPUT 707;":MEASURE:LIMITTEST MEASURE"   !Start test
OUTPUT 707;":RUN" !start acquisition
```

When a limit test failure occurs, bit 3 (LTER) of the status byte is set to a 1.

CURSor?

Query

```
:MEASure:CURSor? {DELTa | STARt | STOP}
```

The :MEASure:CURSor query returns the time and voltage values of the specified marker as an ordered pair of time/voltage values.

- If DELta is specified, the instrument returns the value of delta V (y) and delta T (x).
- If STARt is specified, the positions of the x1marker and y1marker are returned.
- If STOP is specified, the positions of the x2marker and y2marker are returned.

When the CURSor query is sent, no measurement is made and the cursors are not moved.

Returned Format

```
[:MEASure:CURSor] <time>,<voltage><NL>
```

`<time>` ::= delta time, x1marker time, or x2marker time (exponential - NR3 format).

`<voltage>` ::= delta voltage, y1marker voltage, or y2marker voltage (exponential - NR3 format).

Example

```
OUTPUT 707;":MEASURE:SOURCE CHAN1"    !select measurement
source
OUTPUT 707;":MEASURE:FREQUENCY?" !perform frequency
measurement
ENTER 707;Fr                        !read measurement results
Output 707:":MEASURE:CURSOR? START"
ENTER 707;Tme,Vlt !read voltage and time of freq measurement
PRINT Tme,Vlt
```

DEFine

Command

```
:MEASure:DEFine <define_argument>
```

The :MEASure:DEFine command defines the setup for a measurement.

`<define_argument>` ::= {DELay,<polarity>,<edge_number>,<level>,<polarity>,<edge_number>, <level> | PWIDth,<level> | NWIDth,<level>}

`<polarity>` ::= {POSitive | NEGative}

`<edge_number>` ::= 1 to 4000 specifying a displayed edge (integer - NR1 format)

`<level>` ::= {MIDDle | UPPer | LOWer}

Example

```
OUTPUT 707;":MEASURE:DEFINE
DELAY,POSITIVE,1,UPPER,NEGATIVE,2,MIDDLE"
```

This example sets the parameters for a time measurement from the first positive edge at the upper threshold level to the second negative edge at the middle threshold. If one source is specified, both parameters apply to that signal. If two sources are specified, the measurement is from the first positive edge on source 1 to the second negative edge on source 2.

The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

Query

```
:MEASure:DEFine? {DELay | PWIDth | NWIDth}
```

The DEFine query returns the current setup.

Returned Format

```
[:MEASure:DEFine] <define_argument><NL>
```

<define_argument> ::= {DELay,<polarity>,<edge_number>,<level>,<polarity>,<edge_number>, <level> | PWIDth,<level> | NWIDth,<level>}

<polarity> ::= {POSitive | NEGative}

<edge_number> ::= 1 to 4000 specifying a displayed edge (integer - NR1 format)

<level> ::= {MIDDle | UPPer | LOWer}

Example

```
DIM Dfn$[100]
OUTPUT 707;":MEAS:DEF? DEL"
ENTER 707;Dfn$
PRINT Dfn$
```

DELay

Command

```
:MEASure:DELay
```

The :MEASure:DELay command determines the delay from the first specified edge on one source to the next specified edge on the same source, or to the first specified edge on another source. This measurement is the same as the Δt measurement on the front panel.

One or two sources can be specified with the :MEASure:SOURce command.

If user-defined measurement specifications are selected, make sure the defined measurement is displayed.

Example

```
OUTPUT 707;":MEASURE:DELAY"
```

Query

```
:MEASure:DELay?
```

The DELay query returns the specified delay value.

Returned Format

```
[:MEASure:DELay] <value><NL>
```

<value> ::= time value in seconds (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:DEL?"
ENTER 707;Dly
PRINT Dly
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

DESTination

Command

```
:MEASure:DESTination
<source_argument>,<destination_argument>
```

The :MEASure:DESTination command specifies the source and destination to be used when a measurement limit test violation is found.

Source choices include the screen, any channel, or any function.

Destination choices are as follows:

- Waveform memory: when selected, the memory is overwritten each time a violation is found. Can only be selected when the source is CHANnel or FUNCtion.
- MULTiple memory: when selected, up to 665 records may be saved. If multiple failures occur and MEASure:POSTfailure STOP is selected, 665 records are saved and the test stops. If multiple failures occur and MEASure:POSTfailure CONTinue is selected, the data records wrap around. Can only be selected when the source is CHANnel or FUNCtion.
- Pixel memory: when selected, an accumulated save occurs. Can only be selected when the source is SCReen.
- Off: when selected, no save occurs.

`<source_argument>` ::= {SCReen | CHANnel<n> | FUNCtion{ 1 | 2 | 3 | 4 }}

`<destination_argument>` ::= {WMEMory{ 1 | 2 | 3 | 4 } | MULTiple | OFF} when source argument is CHANnel or FUNCtion

::= {OFF | PMEMory {1 |2}} when source argument is SCReen

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:DESTINATION CHANNEL1,MULTIPLE"
```

Query

```
:MEASure:DESTination {CHANnel<n> |
FUNCtion{1 | 2 | 3 | 4} | SCReen}?
```

The DESTination query returns the destination currently selected for the specified source.

Returned Format

```
[:MEASure:DESTination {{CHANnel<n>} | {FUNCtion{ 1 | 2 | 3 |
4} | SCReen}] {WMEMory{ 1 | 2 | 3 | 4} | PMEMory{ 1 | 2} |
MULTiple | OFF}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Dst$[50]
OUTPUT 707;":MEAS:DEST?"
ENTER 707;Dst$
PRINT Dst$
```

DUTycycle

Command

```
:MEASure:DUTycycle
```

The :MEASure:DUTycycle command places the instrument in the continuous measurement mode and starts the duty cycle measurement.

The positive pulse width and the period of the specified signal are measured, then the duty cycle is calculated with the following formula:

duty_cycle = +pulse_width/period

Example

```
OUTPUT 707;":MEASURE:DUTYCYCLE"
```

Query

```
:MEASure:DUTycycle?
```

The DUTycycle query measures and outputs the duty cycle of the signal specified by the SOURce command. The signal must be displayed for the measurement to be made. The value returned for duty cycle is the ratio of the positive pulse width to the period.

Returned Format

```
[:MEASure:DUTycycle] <value><NL>
```

<value> ::= ratio of positive pulse width to period (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:DUT?"
ENTER 707;Dc
PRINT Dc
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

ESTArt

Command

```
:MEASure:ESTArt <slope_and_occurrence>
```

The :MEASure:ESTArt command positions the x1marker on the specified edge and slope of the displayed waveform. All edges must be displayed and are counted from the left edge of the display. The x1marker is positioned where y1marker intersects the waveform. The desired edge is specified by sending an integer value after the command name. If a positive integer is sent, the x1marker is placed on a positive-going waveform edge. If a negative integer is sent, the x1marker is placed on a negative-going waveform edge. If y1marker does not intersect the waveform as specified, the error message 12, "Edges required not found" is displayed.

> The short form of this command, ESTA, does not follow the defined short form convention. The normal short form "EST" is the same for ESTArt and ESTOp. Sending EST for the ESTArt command produces an error.

`<slope_and_occurrence>` ::= -4000 to 4000 excluding 0 (if a positive value is sent the + sign may be omitted or a space may be used) (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:ESTART 2"
```

This example places the x1marker at the second displayed positive-going intersection of the waveform and y1marker.

Query

```
:MEASure:ESTArt?
```

The ESTArt query returns the currently specified edge.

Returned Format

```
[:MEASure:ESTArt] <slope_and_occurrence><NL>
```

`<slope_and_occurrence>` ::= edge number, -4000 to 4000 (integer - NR1 format)

Example

```
OUTPUT 707;":MEAS:ESTA?"
ENTER 707;Value
PRINT Value
```

ESTOp

Command

```
:MEASure:ESTOp <slope_and_occurrence>
```

The :MEASure:ESTOp command positions the x2marker on the specified edge and slope of the displayed waveform. All edges must be displayed and are counted from the left edge of the display. The x2marker is positioned where y2marker intersects the waveform. The desired edge is specified by sending an integer value after the command name. If a positive integer is sent, the oscilloscope places the x2marker on a positive-going waveform edge. If a negative integer is sent, the x2marker is placed on a negative-going waveform edge.

If the y2marker does not intersect the waveform as specified, the error message 12, "Edges required not found" is displayed.

> The short form of this command, ESTO, does not follow the defined short form convention. The normal short form "EST" is the same for ESTArt and ESTOp. Sending EST for the ESTOp command produces an error.

`<slope_and_occurrence>` ::= -4000 to 4000 excluding 0 (if a positive value is sent the + sign may be omitted or a space may be used) (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:ESTOP -2"
```

This example places the x2marker at the second displayed negative-going intersection of the waveform and the y2marker.

Query

```
:MEASure:ESTOp?
```

The ESTOp query returns the currently specified edge.

Returned Format

```
[:MEASure:ESTOp] <slope_and_occurrence><NL>
```

`<slope_and_occurrence>` ::= edge number, -4000 to 4000 (integer - NR1 format)

Example

```
OUTPUT 707;":MEAS:ESTO?"
ENTER 707;Value
PRINT Value
```

EANalysis

Command

```
:MEASure:EANalysis {{OFF | 0} | {ON | 1}}
```

The :MEASure:EANalysis command turns the extended analysis function on and off. When this mode is on, up to 1,000 measurements are averaged and returned for the following measurement types.

- DUTYcycle
- FALLtime
- FREQuency
- NWIDth
- PERiod
- PWIDth
- RISetime
- TDELta
- VACRms
- VAVerage
- VDCRms
- VDELta

When extended analysis is on, measurement results may take as long as several minutes to complete.

Example

```
OUTPUT 707;":MEASURE:EANALYSIS ON"
```

Query

```
:MEASure:EANalysis?
```

The EANalysis query returns the current mode.

Returned Format

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

Example

```
OUTPUT 707;":MEAS:EAN?"
ENTER 707;Exa
PRINT Exa
```

FALLtime

Command

```
:MEASure:FALLtime
```

The :MEASure:FALLtime command places the instrument in the continuous measurement mode and starts a fall time measurement.

Example

```
OUTPUT 707;":MEASURE:FALLTIME"
```

Query

```
:MEASure:FALLtime?
```

The FALLtime query measures and outputs the fall time of the first displayed falling (negative-going) edge. For highest measurement accuracy, set the sweep speed as fast as possible while leaving the falling edge of the waveform on the display. The fall time is determined by measuring the time at the upper threshold of the falling edge, then measuring the time at the lower threshold of the falling edge and calculating the fall time with the following formula:

fall_time = time_at_lower_threshold_point - time_at_upper_threshold_point.

Returned Format

```
[:MEASure:FALLtime] <value><NL>
```

<value> ::= time in seconds between lower threshold and upper threshold voltage points (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:FALL?"
ENTER 707;F11
PRINT F11
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

FREQuency

Command

```
:MEASure:FREQuency
```

The :MEASure:FREQuency command places the instrument in the continuous measurement mode and starts a frequency measurement.

Example

```
OUTPUT 707;":MEASURE:FREQUENCY"
```

Query

```
:MEASURE:FREQuency?
```

The FREQuency query measures and outputs the frequency of the first complete cycle on the screen. This command uses the 50% levels when Standard measurements are selected and the mid-threshold value when User-Defined measurements are selected.

The algorithm is:

```
if_first_edge_on_screen_is_rising then
    frequency=1/(time_at_2nd_rising_edge - time_at_1st_rising_edge)
else
    frequency=1/(time_at_2nd_falling_edge - time_at_1st_falling_ edge)
```

Returned Format

```
[:MEASure:FREQuency] <value><NL>
```

<value> ::= frequency in Hertz (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:FREQ?"
ENTER 707;Frq
PRINT Frq
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

LIMittest

Command

```
:MEASure:LIMittest {MEASure | OFF}
```

The :MEASure:LIMittest command is used to start or stop a limit test. To perform a limit test, the following conditions must be satisfied:

- The measurement must be selected.
- The measurement comparison must be specified.
- The instrument must be acquiring data.

If LIMittest is sent with the MEASure parameter, then the instrument starts the test. If the OFF parameter is sent, the test is stopped.

The LTF (limit test failure) bit of the status byte is set when a failure is found. This bit can be read with a *STB? or LTER? query, or by doing a remote interface Serial Poll. The *STB? query does not clear the bit, and the results require some additional evaluation. The LTER? query returns the actual value of the bit and clears the bit so that the next limit test failure can be monitored. For greater speed, the *SRE command can be used in conjunction with a remote interface Serial Poll. For more information, refer to the individual commands, queries, or functions.

Example

```
OUTPUT 707;":MEASURE:RISETIME" !Select measurement
OUTPUT 707;":MEASURE:COMPARE RISETIME,10NS,2NS" !Specify
measurement limits
OUTPUT 707;":DIGITIZE CHAN1" !Acquire data
OUTPUT 707;":MEAS:LIM MEAS"
OUTPUT 707; "LTER?"
ENTER 707;Test  !0=PASS 1=FAIL
```

LOWer

Command

```
:MEASure:LOWer [<source>,]<lower_threshold>
```

The :MEASure:LOWer command sets the lower measurement threshold. This command sends a value to the instrument in the units selected with the UNITs command. Source can be specified using the optional <source> parameter. If <source> is not specified, the source set using the MEASure:SOURce command is used.

> Set the measurement units with the :MEASure:UNITs command prior to sending the lower threshold value.

`<source>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

`<lower_threshold>` ::= user defined lower threshold in percent or volts (integer - NR1 format)

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:MODE USER"
OUTPUT 707;":MEASURE:UNITS PERC"
OUTPUT 707;":MEASURE:LOWER CHAN1,47"
```

Query `:MEASure:LOWer? [<source>]`

The LOWer query returns the current setting of the lower measurement threshold.

Returned Format `[:MEASure:LOWer] <source>,<lower_threshold><NL>`

`<source>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

`<lower_threshold>` ::= user defined lower threshold in percent or volts (selected by :MEASure:UNITs) (integer - NR1 format)

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Lwr$[50]
OUTPUT 707;":MEAS:LOW?"
ENTER 707;Lwr$
PRINT Lwr$
```

The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

MODE

Command

```
:MEASure:MODE {STANdard | USER}
```

The :MEASure:MODE command selects standard or user-defined definitions and thresholds for voltage and time measurements.

Example

```
OUTPUT 707;":MEASURE:MODE STANDARD"
```

Query

```
:MEASure:MODE?
```

The MODE query returns the current measurement mode setting.

Returned Format

```
[:MEASure:MODE] {STANdard | USER}<NL>
```

Example

```
DIM Md$[50]
OUTPUT 707;":MEAS:MODE?"
ENTER 707;Md$
PRINT Md$
```

The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

MWINdow

Command

```
:MEASure:MWINdow {MARKers | SCReen}
```

The :MEASure:MWindows sets measurements to marker or screen mode.

Example

```
OUTPUT 707;":MEASURE:MWINdow MARKers"
```

Query

```
:MEASure:MWINdow?
```

The MODE query returns the current MWINdows setting.

Returned Format

```
[:MEASure:MWINdows] {MARKers | SCReen}<NL>
```

Example

```
DIM Md$[50]
OUTPUT 707;":MEAS:MWINdows?"
ENTER 707;Md$
PRINT Md$
```

> Measurement queries such as :MEASure:VPP? or :MEASure:RISEtime? will reset the instrument to Continuous off and move the markers to show where the measurement was made, thereby erasing any marker setup. To obtain measurement results without moving the markers, use simple commands to turn on the measurement and use the :MEASure:RESults? query.

Example

```
DIM Md$[100]
OUTPUT 707;":MEAS:SCRATCH"          !clear previous measurement
OUTPUT 707;":MEAS:MWINdows MARK"    !put in marker mode
OUTPUT 707;":DIG CHAN1"             !capture data
OUTPUT 707;":MEAS:VPP"              !do vpp measurement
OUTPUT 707;":MEAS:RESults?"         obtain results
ENTER 707;Md$
PRINT Md$
```

NWIDth

Command

```
:MEASure:NWIDth
```

The :MEASure:NWIDth command places the instrument in the continuous measurement mode and starts an NWIDth measurement.

If User Defined measurements are selected, the measurement is made at the threshold value currently selected using the :MEASure:DEFine, :UPPer, and :LOWer commands.

The algorithm is:

```
if_the_first_edge_on_screen_is_rising then
    width = (time_at_second_rising_edge - time_at_first_falling_edge)
else
    width = (time_at_first_rising_edge - time_at_first falling_edge)
```

Example

```
OUTPUT 707;":MEASURE:NWIDTH"
```

Query

```
:MEASure:NWIDth?
```

If Standard measurements are selected, the NWIDth query measures and outputs the width of the first negative pulse on the screen using 50% levels.

Returned Format

```
[:MEASure:NWIDth] <value><NL>
```

<value> ::= negative pulse width in seconds (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:NWID?"
ENTER 707;Nwd
PRINT Nwd
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

OVERshoot

Command

```
:MEASure:OVERshoot
```

The :MEASure:OVERshoot command places the instrument in the continuous measurement mode and selects the overshoot measurement.

Example

```
OUTPUT 707;":MEASURE:OVERSHOOT"
```

Query

```
:MEASure:OVERshoot?
```

The OVERshoot query measures and outputs the overshoot of the first displayed edge of the selected signal. Overshoot is measured with the following algorithm:

```
if_the_first_edge_on_screen_is_rising
then
    overshoot = (Vmax - Vtop)/Vamplitude
else
    overshoot = (Vbase - Vmin)/Vamplitude
```

Returned Format

```
[:MEASure:OVERshoot] <value><NL>
```

<value> ::= ratio of overshoot to Vamplitude in percent (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:OVER?"
ENTER 707;Ovr
PRINT Ovr
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

PERiod

Command

```
:MEASure:PERiod
```

The :MEASure:PERiod command places the instrument in the continuous measurement mode and selects the period measurement.

Example

```
OUTPUT 707;":MEASURE:PERIOD"
```

Query

```
:MEASure:PERiod?
```

The PERiod query measures and outputs the period of the first complete cycle on the screen. The period is measured at the 50% point when standard measurements are selected and at the mid-threshold voltage level of the waveform when user-defined measurements are selected.

The algorithm for this measurement is:

```
if_the_first_edge_on_screen_is_rising
then
    period = (time_at_second_rising_edge - time_at first_rising_edge)
else
    period = (time_at_second_falling_edge - time_at first_falling_edge)
```

Returned Format

```
[:MEASure:PERiod] <value><NL>
```

<value> ::= waveform period in seconds (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:PER?"
ENTER 707;Prd
PRINT Prd
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

POSTfailure

Command

```
:MEASure:POSTfailure {CONTinue | STOP}
```

The :MEASure:POSTfailure command specifies what the instrument will do after a violation is found by the limit test.

- If STOP is selected, the instrument stops the limit test when the first violation is found. If multiple memory is selected as the destination, then the test will terminate after all available memory space is filled.
- If CONTinue is selected and a violation is found, the violation is written to the desired location, and the instrument continues to look for another violation. If a waveform memory is selected as the destination, then all subsequent violations will overwrite the previous violation. If multiple memory is selected as the destination, then the data records will wrap around.

The destination can be specified with the :MEASure:DESTination command.

Example

```
OUTPUT 707;":MEASURE:POSTFAILURE CONTINUE"
```

Query

```
:MEASure:POSTfailure?
```

The POSTfailure query returns the current selection.

Returned Format

```
[:MEASure:POSTfailure] {CONTinue | STOP}<NL>
```

Example

```
DIM Pf$[50]
OUTPUT 707;":MEAS:POST?"
ENTER 707;Pf$
PRINT Pf$
```

PREShoot

Command

```
:MEASure:PREShoot
```

The :MEASure:PREShoot command places the instrument in the continuous measurement mode and starts the preshoot measurement.

Example

```
OUTPUT 707;":MEASURE:PRESHOOT"
```

Query

```
:MEASure:PREShoot?
```

The PREShoot query measures and outputs the preshoot of the first displayed edge of the selected signal. Preshoot is measured with the following algorithm:

```
if_the_first_edge_on_screen_is_rising
then
    preshoot = (Vbase - Vmin)/Vamplitude
else
    preshoot = (Vmax - Vtop)/Vamplitude
```

Returned Format

```
[:MEASure:PREShoot] <value><NL>
```

<value> ::= ratio of preshoot to Vamplitude in percent (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:PRES?"
ENTER 707;Prs
PRINT Prs
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

PWIDth

Command

```
:MEASure:PWIDth
```

The :MEASure:PWIDth command places the instrument in the continuous measurement mode and starts the Pwidth measurement.

Example

```
OUTPUT 707;":MEASURE:PWIDTH"
```

Query

```
:MEASure:PWIDth?
```

The PWIDth query measures and outputs the width of the first displayed positive pulse. Pulse width is measured at the 50% voltage level with standard measurements selected. If User Defined measurements are selected, the measurement is made at the threshold value currently selected using the :MEASure:DEFine, :UPPer, and :LOWer commands.

The algorithm for this measurement is:

```
if_the_first_edge_on_screen_is_falling then
    width = (time_at_second_ falling_edge - time_at_first_rising_edge)
else
    width = (time_at_first_falling_edge - time_at_first_rising_edge)
```

Returned Format

```
[:MEASure:PWIDth] <value><NL>
```

`<value>` ::= width of positive pulse in seconds (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:PWID?"
ENTER 707;Pwd
PRINT Pwd
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

RESults?

Query

```
:MEASure:RESults?
```

The :MEASure:RESults query returns the currently active measurements. If statistics are on (and MEASure:STATistics:MODE NORMal is selected), the minimum, maximum, and average is returned for each measurement. If the limit test is on and POSTfailure is set to CONTinued, the pass ratio is returned instead of the average. If MEASure:STATistics:MODE SDEViation is selected, mean and standard deviation measurements are returned. If the number of measurements returned is 0, then no measurements are returned.

Returned Format

```
[:MEASure:RESults] <number_of_meas>[;<measurement>]...<NL>
```

<number_of_meas> ::= number of measurements displayed on the screen, 0 through 23 (integer - NR1 format).

<measurement> ::= {DELay <result> | DUTycycle <result> | FALLtime <result> | FREQuency <result> | NWIDth <result> | OVERshoot <result> | PERiod <result> | PREShoot <result> | PWIDth <result> | RISetime <result> | TMAX <result> | TMIN <result> | TVOLt <result> | VACRms <result> | VAMPlitude <result> | VAVerage <result> | VBASe <result> | VDCRms <result> | VMAX <result> | VMIN <result> | VPP <result> | VTIMe <result> | VTOP <result>}

<result> ::= individual measurement results (exponential - NR3 format)

TMAX, TMIN, TVOLt, and VTIMe measurement results are only displayed on the screen by using the front panel keys or the :SYSTem:KEY command.

Example

```
DIM Mr$[100]
OUTPUT 707;":MEASURE:RESULTS?"
ENTER 707;Mr$
PRINT Mr$
```

measurement_results returned are dependent on the current :MEASure:STATistics and :MEASure:STATistics:MODE selections. When ON and MODE is set to NORMal, the current, minimum, maximum, and average (or pass) measurements are returned. When ON and MODE is set to STDViation, the current, average, and standard deviation (or pass) measurements are returned. When OFF, only the current measurement is returned.

RISetime

Command

```
:MEASure:RISetime
```

The :MEASure:RISetime command places the instrument in the continuous measurement mode and starts a rise time measurement.

Example

```
OUTPUT 707;":MEASURE:RISETIME"
```

Query

```
:MEASure:RISetime?
```

The RISetime query measures and outputs the rise time of the first displayed rising (positive-going) edge. For maximum measurement accuracy set the sweep speed as fast as possible while leaving the leading edge of the waveform on the display. The rise time is determined by measuring the time at the lower threshold of the rising edge and the time at the upper threshold of the rising edge, then calculating the rise time with the following formula:

rise_time = (time_at_upper_threshold_point - time_at_lower threshold_point)

Returned Format

```
[:MEASure:RISetime] <value><NL>
```

<value> ::= rise time in seconds (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:RIS?"
ENTER 707;Rs
PRINT Rs
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

SCRatch

Command

```
:MEASure:SCRatch
```

The :MEASure:SCRatch command clears the measurement results from the oscilloscope display.

Example

```
OUTPUT 707;":MEASURE:SCRATCH"
```

SOURce

Command

```
:MEASure:SOURce <source>[,<source>]
```

The :MEASure:SOURce command selects the sources for the measurements. The specified source becomes the source for the MEASure subsystem commands.

Two sources can be specified with this command. All measurements except DELay are made on the first specified source. The DELay measurement uses two sources if two have been specified. If only one source is specified, the DELay measurement uses that source for both of its parameters.

`<source>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:SOURCE CHANNEL1,WMEMORY1"
```

Query

```
:MEASure:SOURce?
```

The SOURce query returns the current source selection. If the specified sources are different, both are returned. Otherwise, one source is returned.

Returned Format

```
[:MEASure:SOURce] <source>[,<source>]<NL>
```

`<source>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[50]
OUTPUT 707;":MEAS:SOUR?"
ENTER 707;Src$
PRINT Src$
```

STATistics

Command

```
:MEASure:STATistics {{OFF | 0} | {ON | 1}}
```

The :MEASure:STATistics command turns the statistics function on and off. When this mode is on, and the measurements are in the continuous mode, either the minimum, maximum, average, and current measurements; or standard deviation measurements are shown as the active measurements (depending on the current MEASure:STATistics:MODE selection). If a RESults query is executed, all of the displayed data is returned to the controller.

> "Average" is replaced by "pass ratio" when the "limit test" is selected and "after failure" is set to continue. Pass ratio lists the percentage of times a certain test passed.

Example

```
OUTPUT 707;":MEASURE:STATISTICS ON"
```

Query

```
:MEASure:STATistics?
```

The STATistics query returns the current mode.

Returned Format

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

Example

```
DIM Stt$[50]
OUTPUT 707;":MEAS:STAT?"
ENTER 707;Stt$
PRINT Stt$
```

STATistics:MODE

Command

```
:MEASure:STATistics:MODE {NORMal | SDEViation}}
```

The :MEASure:STATistics:MODE command is used to select the measurement results returned when STATistics mode is set to on.

When set to NORMal the current, minimum, maximum, and average measurements are shown as the active measurements.

When set to SDEViation, the current mean and standard deviation measurements are shown as the active measurements. Standard deviation is calculated as follows:

$$standard\_deviation = square\ root\ of\ \left[\ \frac{1}{ave}\ \sum_{1}^{n}\ (meas - avg)^{2)}\right]$$

Where:

ave = number of averages taken
n = number of data points
meas = measurement result
avg = average measurement

Example

```
OUTPUT 707;":MEASURE:STATISTICS:MODE SDEVIATION"
```

Query

```
:MEASure:STATistics:MODE?
```

The STATistics:MODE query returns the current mode.

Returned Format

```
[:MEASure:STATistics:MODE] {NORMal | SDEViation}<NL>
```

Example

```
DIM Stm$[50]
OUTPUT 707;":MEAS:STAT:MODE?"
ENTER 707;Stm$
PRINT Stm$
```

TDELta?

Query

```
:MEASure:TDELta?
```

The :MEASure:TDELta query returns the time difference between the x1 and x2 time markers:

Tdelta = x2marker - x1marker

This is the same value that is displayed as delta x on the front panel screen.

> This command is identical to the MARKer:XDELta command, and is provided for compatibility with other oscilloscopes.

Returned Format

```
[:MEASure:TDELta] <value><NL>
```

<value> ::= difference between x2 and x1 markers (exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:TDELTA?"
ENTER 707;Tdl
PRINT Tdl
```

TMAX?

Query

```
:MEASure:TMAX?
```

The :MEASure:TMAX query returns the time at which the first maximum voltage occurred.

Returned Format

```
[:MEASure:TMAX] <time><NL>
```

<time> ::= time at maximum voltage (exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:TMAX?"
ENTER 707;Tmx
PRINT Tmx
```

TMIN?

Query

```
:MEASure:TMIN?
```

The :MEASure:TMIN query returns the time at which the first minimum voltage occurred.

Returned Format

```
[:MEASure:TMIN] <time><NL>
```

::= time at minimum voltage (exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:TMIN?"
ENTER 707;Tmn
PRINT Tmn
```

TSTArt

Command

```
:MEASure:TSTArt <tstart_argument>
```

The :MEASure:TSTArt command moves the x1marker to the specified time with respect to the trigger time.

> This command is identical to the MARKer:X1Position command, and is provided for compatibility with other oscilloscopes.

> The short form of this command, TSTA, does not follow the defined short form convention. The normal short form "TST" is the same for TSTArt and TSTOp. Sending TST for the TSTArt command produces an error.

`<tstart_argument>` ::= time at x1marker in seconds (double exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:TSTART 30 NS"
```

Query

```
:MEASure:TSTArt?
```

The TSTArt query returns the time at the x1marker.

Returned Format

```
[:MEASure:TSTArt]<tstart_argument><NL>
```

`<tstart_argument>` ::= time at x1marker in seconds (double exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:TSTA?"
ENTER 707;Tst
PRINT Tst
```

TSTOp

Command

```
:MEASure:TSTOp <tstop_argument>
```

The :MEASure:TSTOp command moves the x2marker to the specified time with respect to the trigger time.

> This command is identical to the MARKer:X2Position command, and is provided for compatibility with other oscilloscopes.

> The short form of this command, TSTO, does not follow the defined short form convention. The normal short form "TST" is the same for TSTArt and TSTOp. Sending TST for the TSTOp command produces an error.

`<tstop_argument>` ::= time at x2marker in seconds (double exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:TSTOP 40 NS"
```

Query

```
:MEASure:TSTOp?
```

The TSTOp query returns the time at the x2marker.

Returned Format

```
[:MEASure:TSTOp]<tstop_argument> <value><NL>
```

`<tstop_argument>` ::= time at x2marker in seconds (double exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:TSTO?"
ENTER 707;Tst
PRINT Tst
```

TVOLt?

Query

```
:MEASure:TVOLt?
<tvolt_argument>,<slope_and_occurrence>
```

When the :MEASure:TVOLt query is sent, the displayed signal is searched for the defined voltage level and transition. The time interval between the trigger event and this defined occurrence is returned as the response to this query.

The voltage can be specified as a negative or positive voltage. To specify a negative voltage, use a minus (–) sign. The sign of the slope selects a rising (+) or falling (–) edge.

The magnitude of occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the specified voltage level in the positive direction. Once this voltage crossing is found, the oscilloscope outputs the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope outputs +9.99999E+37. This value is returned if the waveform does not cross the specified voltage, or if the waveform does not cross the specified voltage for the specified number of times in the specified direction.

`<tvolt_argument>` ::= real number representing positive or negative voltage level that the waveform must cross.

`<slope_and_occurrence>` ::= slope is the direction of the waveform when <voltage> is crossed - rising (space character or +) or falling (-).

::= occurrence is the number of crossings to be reported (if one - the first crossing is reported, if two - the second crossing is reported.)

Returned Format

```
[:MEASure:TVOLt] <time><NL>
```

`<time>` ::= time in seconds of specified voltage crossing (exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:TVOLT? -.250,+3"
ENTER 707;Tvlt
PRINT Tvlt
```

UNITs

Command

```
:MEASure:UNITs {PERCent | VOLTs}
```

The :MEASure:UNITs command sets the measurement threshold units when the user defined measurement mode is selected. The UNITs can be set to PERCent or VOLTs.

Example

```
OUTPUT 707;":MEASURE:MODE USER"
OUTPUT 707;":MEASURE:UNITS PERCENT"
```

Query

```
:MEASure:UNITs?
```

The UNITs query returns the currently selected units.

Returned Format

```
[:MEASure:UNITs] {PERCent | VOLTs}<NL>
```

Example

```
DIM Unt$[50]
OUTPUT 707;":MEAS:UNIT?"
ENTER 707;Unt$
PRINT Unt$
```

The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

UPPer

Command

```
:MEASure:UPPer [<source>,]<upper_threshold>
```

The :MEASure:UPPer command sets the upper measurement threshold. This command sends a value to the instrument in the units selected with the UNITS command. Source can be specified using the optional <source> parameter. If <source> is not specified, the source set using the MEASure:SOURce command is used.

> Set the measurement units with the :MEASure:UNITs command prior to sending the upper threshold value.

`<source>` ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

`<upper_threshold>` ::= user defined upper threshold in percent or volts (integer - NR1 format)

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:UPPER CHANNEL1,90"
```

Query

```
:MEASure:UPPer? [<source>]
```

The UPPer query returns the value of the upper measurement threshold.

Returned Format

```
[:MEASure:UPPer] <source>,<upper_threshold><NL>
```

<source> ::= {CHANnel<n> | FUNCtion{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

<upper_threshold> ::= user defined upper threshold in percent or volts (selected by :MEASure:UNITs) (integer - NR1 format)

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Up$[50]
OUTPUT 707;":MEAS:UPP?"
ENTER 707;Up$
PRINT Up$
```

The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

VACRms

Command

```
:MEASure:VACRms
```

The :MEASure:VACRms command and query replaces the :MEASure:VRMS command and query in older oscilloscopes.

The :MEASure:VACRms command places the instrument in the continuous measurement mode and starts an ac rms voltage measurement.

> In computing the VACRms measurement, the average value of the waveform is subtracted from each data point before the RMS voltage is computed.

Example

```
OUTPUT 707;":MEASURE:VACRMS"
```

Query

```
:MEASure:VACRms?
```

The VACRms query measures and outputs the AC RMS voltage of the selected waveform. The AC RMS voltage is measured on the first cycle of the displayed signal. If a complete cycle is not present, the oscilloscope computes the RMS value on all displayed data points.

Returned Format

```
[:MEASure:VACRms] <value><NL>
```

<value> ::= calculated ac rms voltage (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VACR?"
ENTER 707;Vrm
PRINT Vrm
```

> The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VAMPlitude

Command

```
:MEASure:VAMPlitude
```

The :MEASure:VAMPlitude command places the instrument in the continuous measurement mode and starts a Vamplitude measurement.

Example

```
OUTPUT 707;":MEASURE:VAMPLITUDE"
```

Query

```
:MEASure:VAMPlitude?
```

The VAMPlitude query returns the difference between the top and base voltage of the displayed signal. The VAMPlitude value is not normally the same as the Vp-p value when the input signal is a pulse.

The Vamplitude value is calculated with the formula:

Vamplitude = Vtop - Vbase

Returned Format

```
[:MEASure:VAMPlitude] <value><NL>
```

<value> ::= difference between top and base voltages (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VAMP?"
ENTER 707;Vmp
PRINT Vmp
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VAVerage

Command

```
:MEASure:VAVerage
```

The :MEASure:VAVerage command places the instrument in the continuous measurement mode and starts a Vaverage measurement.

Example

```
OUTPUT 707;":MEASURE:VAVERAGE"
```

Query

```
:MEASure:VAVerage?
```

The VAVerage query measures the average voltage of the first cycle of the displayed signal. If a complete cycle is not present, the oscilloscope averages all data points.

Returned Format

```
[:MEASure:VAVerage] <value><NL>
```

<value> ::= calculated average voltage (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VAV?"
ENTER 707;Vv
PRINT Vv
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VBASe

Command

```
:MEASure:VBASe
```

The :MEASure:VBASe command places the instrument in the continuous measurement mode and starts a Vbase measurement.

Example

```
OUTPUT 707;":MEASURE:VBASE"
```

Query

```
:MEASure:VBASe?
```

The VBASe query measures and outputs the voltage value at the base of the waveform. The base voltage of a pulse is normally not the same as the minimum value.

Returned Format

```
[:MEASure:VBASe] <value><NL>
```

<value> ::= voltage at base of selected waveform (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VBAS?"
ENTER 707;Vbs
PRINT Vbs
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VDCRms

Command

```
:MEASure:VDCRms
```

The :MEASure:VDCRms command places the instrument in the continuous measurement mode and starts a dc rms voltage measurement.

> The VDCRms measurement is a true RMS measurement.

Example

```
OUTPUT 707;":MEASURE:VDCRMS"
```

Query

```
:MEASure:VDCRms?
```

The VDCRms query measures and outputs the RMS voltage of the selected waveform. The RMS voltage is measured on the first cycle of the displayed signal. If a complete cycle is not present, the oscilloscope computes the RMS value on all displayed data points.

Returned Format

```
[:MEASure:VDCRms] <value><NL>
```

<value> ::= calculated dc rms voltage (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VDCR?"
ENTER 707;Vrm
PRINT Vrm
```

> The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VDELta?

Query

```
:MEASure:VDELta?
```

The :MEASure:VDELta query outputs the voltage difference between the y1marker and y2marker. No measurement is made when the VDELta query is received by the oscilloscope. The delta voltage value that is output is the current value. This is the same value as the front panel delta y value.

VDELTA = y2marker - y1marker 1

> This command is identical to the MARKer:YDELta command, and is provided for compatibility with other oscilloscopes.

Returned Format

```
[:MEASure:VDELta] <value><NL>
```

`<value>` ::= delta V value in volts (exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:VDELTA?"
ENTER 707;Vd1
PRINT Vd1
```

VFIFty

Command

```
:MEASure:VFIFty
```

The :MEASure:VFIFTty command finds the top and base values of the specified waveforms, then places the y1 and y2markers at the 50% voltage point on the specified sources.

If only one source has been specified with the source command, the VFIFty command sets both y1 and y2 markers to the 50% voltage level on that source.

If two sources are specified with the source command, the y1marker is set to the 50% level of the first specified source and the y2marker is set to the 50% level of the second specified source.

There is no query form of this command.

Example

```
OUTPUT 707;":MEASURE:VFIFTY"
```

VMAX

Command

```
:MEASure:VMAX
```

The :MEASure:VMAX command places the instrument in the continuous measurement mode and starts a Vmax measurement.

Example

```
OUTPUT 707;":MEASURE:VMAX"
```

Query

```
:MEASure:VMAX?
```

The VMAX query measures and outputs the absolute maximum voltage present on the selected waveform.

Returned Format

```
[:MEASure:VMAX] <value><NL>
```

<value> ::= maximum voltage of selected waveform (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VMAX?"
ENTER 707;Vmx
PRINT Vmx
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VMIN

Command

```
:MEASure:VMIN
```

The :MEASure:VMIN command places the instrument in the continuous measurement mode and starts a VMIN measurement.

Example

```
OUTPUT 707;":MEASURE:VMIN"
```

Query

```
:MEASure:VMIN?
```

The VMIN query measures and outputs the absolute minimum voltage present on the selected waveform.

Returned Format

```
[:MEASure:VMIN] <value><NL>
```

<value> ::= minimum voltage value of the selected waveform (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VMIN?"
ENTER 707;Vmn
PRINT Vmn
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VPP

Command

```
:MEASure:VPP
```

The :MEASure:VPP command places the instrument in the continuous measurement mode and starts a VPP measurement.

Example

```
OUTPUT 707;":MEASURE:VPP"
```

Query

```
:MEASure:VPP?
```

The VPP query measures the maximum and minimum voltages for the selected source, then calculates the peak-to-peak voltage and outputs that value. The peak-to-peak voltage (Vpp) is calculated with the following formula:

Vpp = Vmax - Vmin

Vmax and Vmin are the maximum and minimum voltages present on the selected source.

Returned Format

```
[:MEASure:VPP] <value><NL>
```

<value> ::= voltage peak to peak (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VPP?"
ENTER 707;Vp
PRINT Vp
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

VRELative

Command

```
:MEASure:VRELative <percent_argument>
```

The :MEASure:VRELative command moves the y1 and y2markers to the specified percentage points of their last established position. The last established position is not necessarily on the currently displayed waveform.

For example, after a :MEASure:VAMPlitude? query is sent, the y1marker is located at the base (0%) of the signal and the y2marker is at the top (100%) of the signal. If the VRELative 10 command is executed, the y1marker is moved to the 10% level and the y2marker to the 90% level of the signal.

Any value between 0% and 100% can be used. If VRELative 0 is sent, the ymarkers are not moved because the command indicates 0% movement from the current position.

As an example, when the following values are sent, the markers are moved to the following percentage values of their current position:

- 10 moves y1marker to 10% and y2marker to 90%
- 20 moves y1marker to 20% and y2marker to 80%
- 50 moves both markers to 50%
- 80 moves y1marker to 20% and y2marker to 80%
- 90 moves y1marker to 10% and y2marker to 90%

The starting position of the markers must be known for this command to be meaningful. The markers can be set to a known position on the selected waveform using the :MEASure:VAMPlitude? query.

> **The VRELative command does not affect the upper and lower thresholds selected by the UPPer and LOWer commands.**

`<percent_argument>` ::= 0 through 100 (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:VRELATIVE 20"
```

Query

```
:MEASure:VRELative?
```

The VRELative query returns the current relative position (in percent) of VMarker2, which is always in the range of 50% through 100%.

Returned Format

```
[:MEASure:VRELative] <percent_argument><NL>
```

<percent_argument> ::= Vmarker2 relative position in percent {50 through 100} (integer - NR1 format)

Example

```
OUTPUT 707;":MEAS:VREL?"
ENTER 707;Vrl
PRINT Vrl
```

VSTArt

Command

```
:MEASure:VSTArt <vstart_argument>
```

The :MEASure:VSTArt command moves the y1marker to the specified voltage. The values are limited to the currently defined channel, function, or memory range.

> This command is identical to the MARKer:Y1Position command, and is provided for compatibility with other oscilloscopes.

> The short form of this command, VSTA, does not follow the defined short form convention. The normal short form "VST" is the same for VSTArt and VSTOp. Sending VST for the VSTArt command produces an error.

`<vstart_argument>` ::= real number representing voltage value for the y1marker

Example

```
OUTPUT 707;":MEASURE:VSTART -10MV"
```

Query

```
:MEASure:VSTArt?
```

The VSTArt query returns the current voltage level of the y1marker.

Returned Format

```
[:MEASure:VSTArt] <vstart_argument><NL>
```

`<vstart_argument>` ::= voltage at y1marker (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VSTA?"
ENTER 707;Vst
PRINT Vst
```

VSTOp

Command

```
:MEASure:VSTOp <vstop_argument>
```

The :MEASure:VSTOp command moves the y2marker to the specified voltage.

> This command is identical to the MARKer:Y2Position command, and is provided for compatibility with other oscilloscopes.

> The short form of this command, VSTO, does not follow the defined short form convention. The normal short form "VST" is the same for VSTArt and VSTOp. Sending VST for the VSTOp command produces an error.

`<vstop_argument>` ::= real number representing voltage value for the y2marker

Example

```
OUTPUT 707;":MEASURE:VSTOP -100MV"
```

Query

```
:MEASure:VSTOp?
```

The VSTOp query returns the current voltage level of the y2marker.

Returned Format

```
[:MEASure:VSTOp] <vstop_argument><NL>
```

`<vstop_argument>` ::= voltage at y2marker (exponential - NR3 format)

Example

```
OUTPUT 707;":MEAS:VSTO?"
ENTER 707;Vst$
PRINT Vst$
```

VTIMe?

Query

```
:MEASure:VTIMe? <vtime_argument>
```

The :MEASure:VTIMe query returns the voltage at a specified time. The specified time must be on screen and is referenced to the trigger event.

Returned Format

```
[:MEASure:VTIMe] <voltage><NL>
```

`<vtime_argument>` ::= real number representing the displayed time from the trigger in seconds

`<voltage>` ::= voltage at specified time (exponential - NR3 format)

Example

```
OUTPUT 707;":MEASURE:VTIME? .001"
ENTER 707;Vtm
PRINT Vtm
```

VTOP

Command

```
:MEASure:VTOP
```

The :MEASure:VTOP command places the instrument in the continuous measurement mode and starts a Vtop measurement.

Example

```
OUTPUT 707;":MEASURE:VTOP"
```

Query

```
:MEASure:VTOP?
```

The VTOP query returns the voltage at the top of a waveform.

Returned Format

```
[:MEASure:VTOP] <value><NL>
```

<value> ::= voltage at the top of the waveform (exponential - NR3 format)

Example

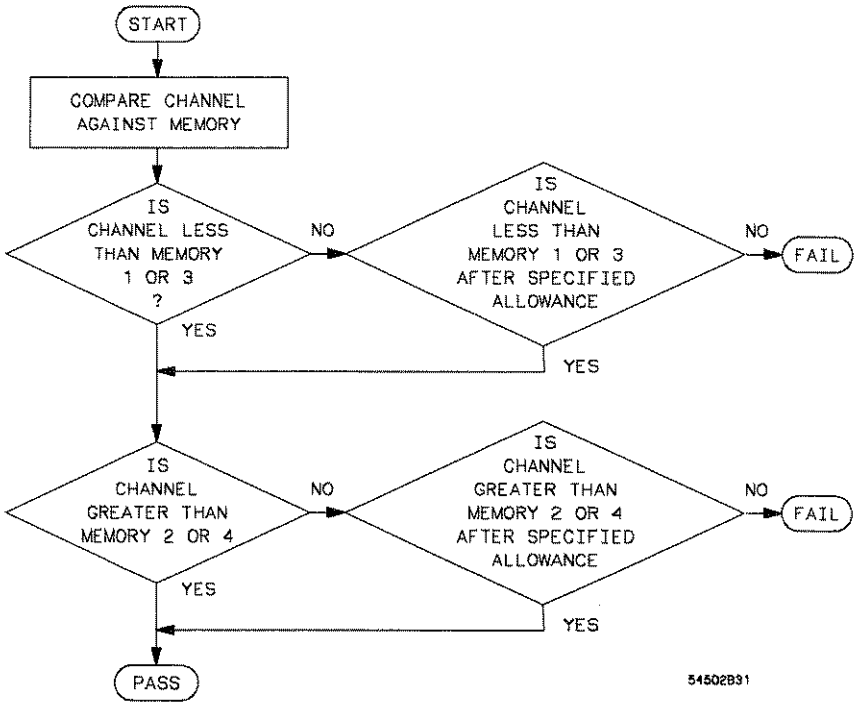
```
OUTPUT 707;":MEAS:VTOP?"
ENTER 707;Vtp
PRINT Vtp
```

The command form places the instrument in continuous mode, while the query form turns off the continuous measurement mode.

WCOMpare

When a waveform comparison test fails, a second test is performed on the waveform data and memory against a specified tolerance or allowance, called a Waveform Comparison Test. The chart in figure 16-2 lists the flow for the waveform comparison test of, for example, channel 2 with respect to memories 3 and 4.

Figure 16–2



Waveform Compare Flowchart Example

WCOMpare:ALLowance

Command

```
:MEASure:WCOMpare:ALLowance <allow_argument>
```

The :MEASure:WCOMpare:ALLowance command allows you to specify an allowance representing the allowable number of vertical divisions that the waveform comparison test can deviate and still pass. The allowance value that you specify over the bus is automatically rounded to the nearest 0.025 divisions. The chart in figure 16-2 lists the flow for the waveform comparison test.

> This command provides the same functionality as the VALLowance command and is retained for compatibility with previous versions of software.

`<allow_argument>` ::= real number representing number of vertical divisions of allowance (0.0 to 8.0)

Example

```
OUTPUT 707;":MEASURE:WCOMPARE:ALLOWANCE 0.0"
```

Query

```
:MEASure:WCOMpare:ALLowance?
```

The :ALLowance query returns the current number of vertical divisions of allowance setting.

Returned Format

```
[:MEASure:WCOMpare:ALLowance] <allow_argument><NL>
```

`<allow_argument>` ::= real number representing number of vertical divisions of allowance (0.0 to 8.0)

Example

```
OUTPUT 707:":MEAS:WCOM:ALL?"
ENTER 707;Wc1
PRINT Wc1
```

WCOMpare:COMPare

Command

```
:MEASure:WCOMpare:COMPare {CHANnel<n> |
FUNCtion{1 | 2 | 3 | 4}}, WMEMory{1 | 2 | 3 | 4}
```

The :MEASure:WCOMpare:COMPare command selects the channel and memory pair to be compared by the waveform comparison test. Specifying wmemory1 or wmemory2 selects both wmemory1 (m1) and wmemory2 (m2) as the mask pair that the channel will be compared to. Specifying wmemory3 or wmemory4 selects both wmemory3 (m3) and wmemory4 (m4) as the mask pair that the channel will be compared to.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MEASURE:WCOMPARE:COMPARE CHANNEL1,WMEMORY1"
```

Query

```
:MEASure:WCOMPare:COMpare?
```

The :COMPare query returns the current channel and memory pair setting.

Returned Format

```
[:MEASure:WCOMpare:COMPare] {CHANnel<n> |
FUNCtion{1 | 2 | 3 ||4}}, WMEMory{1 | 2 | 3 | 4}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Wcc$[50]
OUTPUT 707;":MEAS:WCOM:COMP?"
ENTER 707;Wcc
PRINT Wcc
```

WCOMpare:DESTination

Command

```
:MEASure:WCOMpare:DESTination
<source_argument>,<destination_argument>
```

The :MEASure:WCOMPare:DESTination command specifies the source and destination to be used when a waveform comparison violation is found. Data may be stored to one of the waveform memories, but not to the pair of memories used for the mask in the comparison test. Data can also be stored to one of the pixel memories, the multiple memories, or discarded by specifying OFF.

Source choices include the screen, any channel, or any function.

Destination choices are as follows:

- Waveform memory: when selected, the memory is overwritten each time a violation is found. Can only be selected when the source is CHANnel or FUNCtion.
- Multiple memory: when selected, up to 665 records may be saved. If multiple failures occur and MEASure:POSTfailure STOP is selected, records are saved until available memory is filled and the test stops. If multiple failures occur and MEASure:POSTfailure CONTinue is selected, the data records wrap around. Can only be selected when the source is CHANnel or FUNCtion.
- Pixel memory: when selected, an accumulated save occurs. Can only be selected when the source is SCReen.
- Off: when selected, no save occurs.

`<source_argument>` ::= {SCReen | CHANnel<n> | FUNCtion{ 1 | 2 | 3 | 4 }}

`<destination_argument>` ::= {WMEMory{ 1 | 2 | 3 | 4 } | MULTiple | OFF} when source argument is CHANnel or FUNCtion

::= {OFF | PMEMory {1 |2}} when source argument is SCReen

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

If the waveform memory selected for the destination is the same as the waveform memories used for the mask, the destination is set to OFF. Otherwise, failure information would be written into the mask.

Example

```
OUTPUT 707;":MEASURE:WCOMPARE:DESTINATION CHANNEL1,MULTIPLE"
```

Query

```
:MEASure:WCOMpare:DESTination? {CHANnel<n> |
FUNCtion{ 1 | 2 | 3 | 4 } | SCReen}
```

The DESTination query returns the destination currently selected for the specified source.

Returned Format

```
[:MEASure:WCOMpare:DESTination {{CHANnel<n>} | {FUNCtion{ 1 |
2 | 3 | 4} | SCReen}] {WMEMory{ 1 | 2 | 3 | 4} | PMEMory{ 1 |
2} | MULTiple | OFF}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Dst$[50]
OUTPUT 707;":MEAS:WCOM:DEST? CHAN1"
ENTER 707;Dst$
PRINT Dst$
```

WCOMpare:HALLowance

Command

```
:MEASure:WCOMpare:HALLowance <allow_argument>
```

The :MEASure:WCOMpare:HALLowance command allows you to specify an allowance representing the allowable number of horizontal divisions that the waveform comparison test can deviate and still pass. The allowance value that you specify over the bus is automatically rounded to the nearest 0.025 divisions. The chart in figure 16-2 lists the flow for the waveform comparison test.

`<allow_argument>` ::= real number representing number of horizontal divisions of allowance (0.0 to 8.0)

Example

```
OUTPUT 707;":MEASURE:WCOMPARE:HALLOWANCE 0.0"
```

Query

```
:MEASure:WCOMpare:HALLowance?
```

The :HALLowance query returns the current number of horizontal divisions of allowance setting.

Returned Format

```
[:MEASure:WCOMpare:HALLowance] <allow_argument><NL>
```

`<allow_argument>` ::= real number representing number of horizontal divisions of allowance (0.0 to 8.0)

Example

```
OUTPUT 707;":MEAS:WCOM:HALL?"
ENTER 707;Wcl
PRINT Wcl
```

WCOMpare:POSTfailure

Command

```
:MEASure:WCOMpare:POSTfailure {CONTinue | STOP}
```

The :MEASure:WCOMpare:POSTfailure command specifies what will occur after a violation has been found by the waveform comparison test.

If STOP is selected, the instrument stops the waveform comparison test and acquisition when the first violation is found. If multiple memory is selected as the destination, then the test will terminate after all available memory space is filled.

If CONTinue is selected and a violation is found, the violation is written to the desired location, and the instrument continues to look for another violation. If a waveform memory is selected as the destination, then all subsequent violations will overwrite the previous violation. If multiple memory is selected as the destination, then the data records will wrap around.

The destination can be specified with the :MEASure:WCOMpare:DESTination command.

Example

```
OUTPUT 707;":MEASURE:WCOMPARE:POSTFAILURE CONTINUE"
```

Query

```
:MEASure:WCOMpare:POSTfailure?
```

The :POSTfailure query returns the current postfailure setting.

Returned Format

```
[:MEASure:WCOMpare:POSTfailure] {CONTinue | STOP}<NL>
```

Example

```
DIM Pst$[100]
OUTPUT 707;":MEAS:WCOM:POST?"
ENTER 707; Pst$
PRINT Pst$
```

WCOMpare:VALLowance

Command

```
:MEASure:WCOMpare:VALLowance <allow_argument>
```

The :MEASure:WCOMpare:VALLowance command allows you to specify an allowance representing the allowable number of vertical divisions that the waveform comparison test can deviate and still pass. The allowance value that you specify over the bus is automatically rounded to the nearest 0.025 divisions. The chart in figure 16-2 lists the flow for the waveform comparison test.

`<allow_argument>` ::= real number representing number of vertical divisions of allowance (0.0 to 8.0)

Example

```
OUTPUT 707;":MEASURE:WCOMPARE:VALLOWANCE 0.0"
```

Query

```
:MEASure:WCOMpare:VALLowance?
```

The VALLowance query returns the current number of vertical divisions of allowance setting.

Returned Format

```
[:MEASure:WCOMpare:VALLowance] <allow_argument><NL>
```

`<allow_argument>` ::= real number representing number of vertical divisions of allowance (0.0 to 8.0)

Example

```
OUTPUT 707:":MEAS:WCOM:VALL?"
ENTER 707;Wc1
PRINT Wc1
```

WCOMpare:WTESt

Command

```
:MEASure:WCOMpare:WTESt {MEASure | OFF}
```

The :MEASure:WCOMpare:WTESt command turns the waveform compare test on and off. When the waveform compare test is on, the channel data is compared with either memories 1 and 2, or memories 3 and 4. To pass the test, the channel data must be less than the first memory data (memory 1 or 3) and greater than the second memory data (memory 2 or 4) plus or minus the value set by the :WCOMpare:ALLowance command. Figure 16-2 lists the flow for the waveform comparison test.

The WCOMpare:WTESt runs only on 500 or 501 point records. If the COMPare:WTESt is on and the record sizes are not 500 or 501, the error message "Er 2" will be displayed in the upper-left hand corner of the screen.

> LIMittest and WCOMpare:WTESt cannot run simultaneously. Only one test may be on at one time. If you attempt to run these tests simultaneously, the error message "legal command but safety conflict" (error 211) appears on the screen.

Example

```
OUTPUT 707;":MEASURE:WCOMPARE:WTEST OFF"
```

Query

```
:MEASure:WCOMpare:WTESt?
```

The :WTESt query returns the current waveform compare test state.

Returned Format

```
[MEASure:WCOMpare:WTESt] {MEASure | OFF}<NL>
```

Example

```
DIM Wte$[20]
OUTPUT 707; ":MEAS:WCOM:WTES?"
ENTER 707;Wte$
PRINT Wte$
```

The LTER bit is set when a failure occurs in the :WCOMpare:WTESt. This bit will remain set after a failure until it is read by the :LTER query.

17

Multiple Memory (MMEMory) Subsystem

Introduction

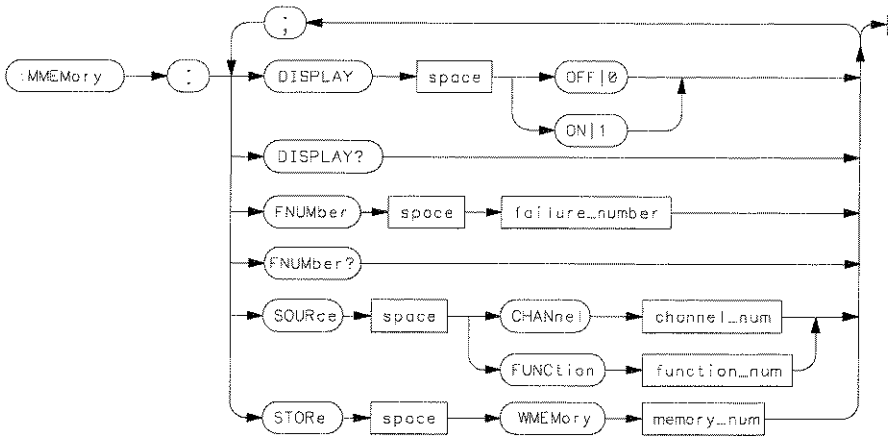
The Multiple Memory (MMEMory) subsystem commands control up to 665 volatile multiple memories of the oscilloscope (data is lost when power is removed from oscilloscope). Multiple memories contain 500 points of waveform information resulting from the compare and limit test violations. See :MEASure:COMPare and :MEASure:WCOMpare:COMPare commands for more information on compare and limit tests. Failure data stored in multiple memory can be displayed on screen, or transferred to nonvolatile waveform memory.

The Multiple Memory subsystem contains the following commands:

DISPlay
FNUMber
SOURce
STORe

Figure 17-1 lists the syntax diagrams for the Multiple Memory subsystem commands.

Figure 17–1



failure_number = an integer, from 1 to 665 defining the failure record.

channel_num = an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).

function_num = an integer, 1 through 4.

memory_num = an integer, 1 through 4.

Multiple Memory Subsystem Commands Syntax Diagram

DISPlay

Command

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

The :MMEMory:DISPlay command controls the multiple memory display. ON allows failure data to be displayed on the screen. Specific failure data is selected using the MMEMory:FNUMber and :MMEMory:SOURce commands.

Example

```
OUTPUT 707;":MMEMORY:DISPLAY ON"
```

Query

```
:MMEMory:DISPlay?
```

The DISPlay query returns the current setting of the command.

Returned Format

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

Example

```
OUTPUT 707;":MMEM:DISP?"
ENTER 707;Disp
PRINT Disp
```

Valid data MUST be present before selecting multiple memory or an error will be generated.

FNUMber

Command

```
:MMEMory:FNUMber <failure_number>
```

The :MMEMory:FNUMber command selects the specific multiple memory failure to be viewed or transferred. Source of the failure is specified using the :MMEMory:SOURce command.

<failure_number> ::= 1 to 665 (integer - NR1 format)

Example

```
OUTPUT 707;":MMEMORY:SOURCE CHAN1"
OUTPUT 707;":MMEMORY:FNUMBER 35"
```

Query

```
:MMEMory:FNUMber?
```

The FNUMber query returns the currently selected failure for the multiple memory commands. A returned "0" indicates that there is not any failure data available for the source selected.

Returned Format

```
[:MMEMory:FNUMber] <failure_number><NL>
```

<failure_number> ::= 0 to 665 (integer - NR1 format)

Example

```
OUTPUT 707;":MMEM:FNUM?"
ENTER 707;Fail
PRINT Fail
```

SOURce

Command

```
:MMEMory:SOURce {CHANnel<n> | {FUNCtion{1|2|3|4}}
```

The :MMEMory:SOURce command selects the channel or function to be used for the source of the multiple memory commands. Specific failure number is selected using the :MMEMory:FNUMber command. Source may be selected only after data failures have been saved to multiple memory. The error message "Settings conflict" is displayed if no failures have been saved.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":MMEMORY:SOURCE CHANNEL1"
```

Query

```
:MMEMory:SOURce?
```

The SOURce query returns the currently selected source for the multiple memory commands.

Returned Format

```
[:MMEMory:SOURce] {CHANnel<n> | {FUNCtion{1|2|3|4}}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[30]
OUTPUT 707;":MMEM:SOUR?"
ENTER 707;Src$
PRINT Src$
```

STORe

Command

```
:MMEMory:STORe {WMEMory{1|2|3|4}
```

The :MMEMory:STORe command moves the selected multiple memory record (source and failure number) to a waveform memory. Specific multiple memory source and failure number is selected using the :MMEMory:SOURce and MMEMory:FNUMber commands.

Example

```
OUTPUT 707;":MMEMORY:SOURCE CHANNEL1"
OUTPUT 707;":MMEMORY:FNUMBER 35"
OUTPUT 707;":MMEMORY:STORE WMEMORY1"
```

18

Pixel Memory (PMEMory) Subsystem

Introduction

The Pixel Memory (PMEMory) subsystem commands control the two volatile pixel memories of the oscilloscope. The memory number specified in the command selects the memory that is affected by the command.

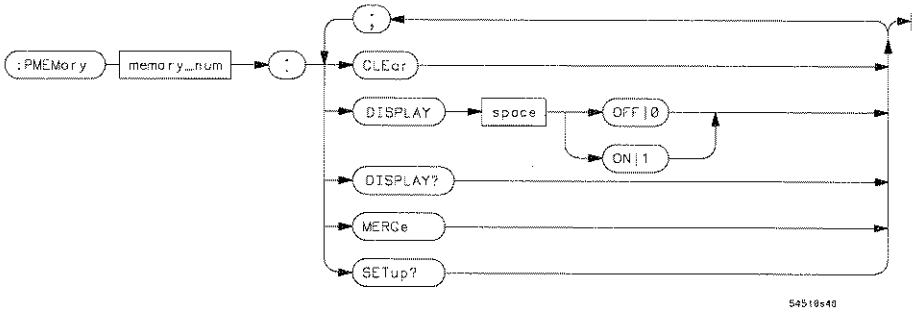
The pixel displays are toggled on and off with the root level commands VIEW and BLANk, or using the PMEMory:DISPlay command.

The Pixel Memory subsystem contains the following commands:

CLEar
DISPlay
MERGe
SETup

Figure 18-1 lists the syntax diagrams for the Pixel Memory subsystem commands.

Figure 18–1



memory_number = an integer, 1 or 2.

Pixel Memory Subsystem Commands Syntax Diagram

CLEar

Command

```
:PMEMory{1 | 2}:CLEar
```

The :PMEMory{1 | 2}:CLEar command clears the contents of the selected pixel memory.

Example

```
OUTPUT 707;":PMEMORY1:CLEAR"
```

This command is similiar to the root level ERASe command.

DISPlay

Command

```
:PMEMory{1 | 2}:DISPlay {{OFF | 0} | {ON | 1}}
```

The :PMEMory{1 | 2}:DISPlay command controls the individual pixel memory displays. ON displays, and OFF blanks the memory selected.

Example

```
OUTPUT 707;":PMEMORY2:DISPLAY ON"
```

Query

```
:PMEMory{1 | 2}:DISPlay?
```

The DISPlay query returns the current setting of the command.

Returned Format

```
[:PMEMory{1 | 2}:DISPlay] {0 | 1}<NL>
```

Example

```
OUTPUT 707;":PMEM1 DISP?"
ENTER 707;Disp
PRINT Disp
```

MERGe

Command

```
:PMEMory{1 | 2}:MERGe
```

The :PMEMory{1 | 2}:MERGe command adds the contents of the active display to the current contents of the specified pixel memory.

Example

```
OUTPUT 707;":PMEMORY2:MERGE"
```

This command is similar to the root level MERGe command.

SETup?

Query

```
:PMEMory{1 | 2}:SETup?
```

The :PMEMory{1 | 2}:SETup query returns the current settings for the PMEMory Subsystem commands.

Returned Format

```
:PMEM{1 | 2}:DISP{0 | 1}<NL>
```

Example

```
OUTPUT 707;":PMEMORY1:SETUP?"
ENTER 707;Stp
PRINT Stp
```

19

SEQuential Subsystem

Introduction

The SEQuential subsystem commands control how data is acquired, captured, and defined during sequential single-shot mode operation.

Sequential single-shot is available only when :TIMebase:SAMPle is set to REALtime. When sequential single-shot mode is selected (:SEQuential:DISPlay ON), all data acquisition is stopped. The acquisition is defined using the :SEQuential:NPOints and :SEQuential:NSEGments commands. Data is acquired when the DIGitize root command is executed.

The :ACQuire:TYPE currently selected determines how the captured data is processed. See "Acquire Subsystem" for more information on acquisition type available. Individual segment selection is possible using the :SEQuential:INCLude, :SEQuential:EXCLude, and :SEQuential:SNUMber commands.

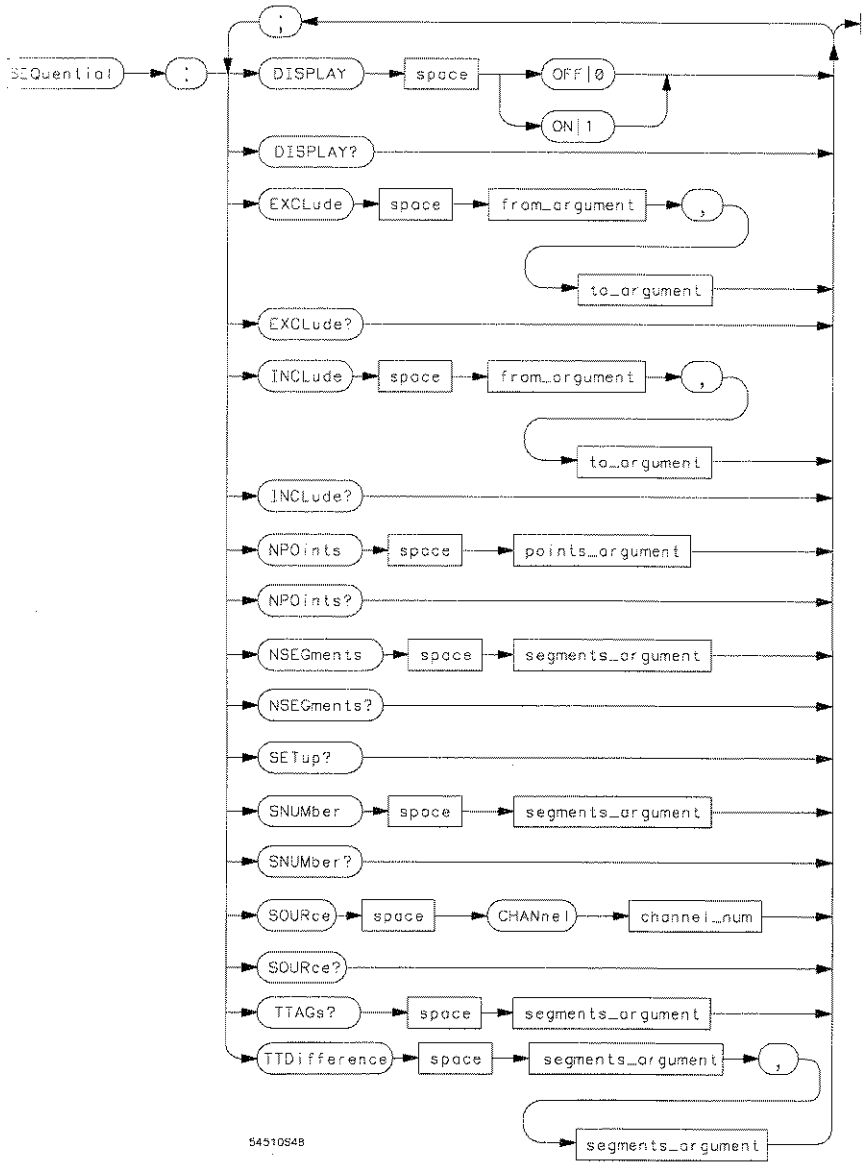
Captured and processed segments can be displayed on the screen (DISPlay Subsystem), measured (MEASure Subsystem), and/or have waveform math functions (FUNCtion Subsystem) performed using the appropriate subsystem commands.

The Sequential subsystem contains the following commands:

DISPlay
EXCLude
INCLude
NPOints
NSEGments
SETup
SNUMber
SOURce
TTAGs
TTDifference

Figure 19-1 lists the syntax diagrams for the Sequential subsystem commands.

Figure 19–1



Sequential Subsystem Commands Syntax Diagram

Figure 19-1

points_argument =	an integer, from 4 to 32,768 defining the points being acquired.
segments_argument =	an integer, from 1 to 8888 dependent on the points_argument.
channel_number =	an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).
from_argument =	an integer, dependent on the segments_argument.
to_argument =	an integer, dependent on the segments_argument.

Sequential Subsystem Commands Syntax Diagram (continued)

DISPlay

Command

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

The :SEQuential:DISPlay command controls sequential single-shot mode operation. "ON" places the oscilloscope in sequential single-shot mode, and stops all data acquisition. "OFF" turns off sequential single-shot mode.

After data has been acquired (using the DIGitize root command), captured segments are displayed on the screen. The type of data displayed is selected using the :ACQuire:TYPE command. When :ACQuire:TYPE is NORMal, a specific segment can be selected using the :SEQuential:SNUMber command.

Example

```
OUTPUT 707;":SEQUENTIAL:DISPLAY ON"
```

Query

```
:SEQuential:DISPlay?
```

The DISPlay query returns the current setting of the command.

Returned Format

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

Example

```
OUTPUT 707;":SEQ:DISP?"
ENTER 707;Disp
PRINT Disp
```

EXCLude

Command

```
:SEQuential:EXCLude <from_argument>,<to_argument>
```

The :SEQuential:EXCLude command selects a range of previously captured segments that will not be selectable for display unless explicitly included using the SEQuential:INCLude command. This command is used to exclude unwanted captured segments (from the most recent sequential acquisition) from measurement, statistics, and display.

When entering the range of segment numbers:

- All segment numbers specified must have been previously acquired (as defined by the SEQuential:NSEGments command).
- The "from segment" specified cannot exceed the "to segment" specified.
- Multiple segment ranges can be excluded.

`<from_argument>` ::= segment number of the lower end of the excluded range (integer - NR1 format)

`<to_argument>` ::= segment number of the upper end of the excluded range (integer - NR1 format)

Example

```
OUTPUT 707;":TIMEBASE:SAMPLE REALTIME"
OUTPUT 707;":SEQUENTIAL:DISPLAY ON
OUTPUT 707;":ACQUIRE:TYPE AVERAGE"
OUTPUT 707;":SEQUENTIAL:NPOINTS 500"
OUTPUT 707;":SEQUENTIAL:NSEGMENTS 10"
OUTPUT 707;":SEQUENTIAL:SOURCE CHANNEL1"
OUTPUT 707;":DIGITIZE CHANNEL1"
OUTPUT 707;":SEQUENTIAL:EXCLUDE 5,10"
```

Query `:SEQuential:EXCLude?`

The EXCLude query returns a list of currently selected excluded segments for the sequential single-shot commands. If no segments have been excluded, returns <NL>.

Returned Format `[:SEQuential:EXCLude] <exclude_list><NL>`

`<exclude_list>` ::= a list of previously captured segment numbers separated by commas (integer - NR1 format)

Example

```
DIM Exc$[50]
OUTPUT 707;":SEQ:EXCL?"
ENTER 707;Exc$
PRINT Exc$
```

INCLude

Command

```
:SEQuential:INCLude <from_argument>,<to_argument>
```

The :SEQuential:INClude command selects a range of previously captured and excluded segments. This command is used to include previously excluded segments (from the most recent sequential acquisition) for measurement, statistics, and display.

> Because all segments are initially acquired as INCLude, this command should only be used to include segments that were previously excluded using the :SEQuential:EXClude command.

When entering the range of segment numbers:

- All segment numbers specified must have been previously acquired (as defined by the SEQuential:NSEGments command).
- The "from segment" specified cannot exceed the "to segment" specified.

`<from_argument>` ::= segment number of the lower end of the included range (integer - NR1 format)

`<to_argument>` ::= segment number of the upper end of the included range (integer - NR1 format)

Example

```
OUTPUT 707;"::SEQUENTIAL:INCLUDE 5,10"
```

Query `:SEQuential:INCLude?`

The INCLude query returns a list of currently selected included segments for the sequential single-shot commands.

Returned Format `[:SEQuential:INCLude] <include_list><NL>`

`<include_list>` ::= a list of previously captured segment numbers separated by commas (integer - NR1 format)

Example

```
DIM Inc$[50]
OUTPUT 707;":SEQ:INCL?"
ENTER 707;Inc$
PRINT Inc$
```

NPOints

Command

```
:SEQuential:NPOints <points_argument>
```

The :SEQuential:NPOints command specifies the number of time buckets for each acquired segment. The number of points specified also affects the total number of SEQuential:NSEGments available.

`<points_argument>` ::= 4 to 32768 (integer - NR1 format)

Example

```
OUTPUT 707;":SEQUENTIAL:NPOINTS 400"
```

Query

```
:SEQuential:NPOints?
```

The NPOints query returns the number of time buckets to be acquired.

Returned Format

```
[:SEQuential:NPOints] <points_argument><NL>
```

`<points_argument>` ::= 4 to 32768 (integer - NR1 format)

Example

```
OUTPUT 707;":SEQ:NPO?"
ENTER 707;Pts
PRINT Pts
```

NSEGments

Command

```
:SEQuential:NSEGments <segments_argument>
```

The :SEQuential:NSEGments command specifies the number of segments that will be acquired for the sequential single-shot mode. The number of time buckets acquired for each segment is specified using the :SEQuential:NPOints command.

The specified number of segments are acquired when the RUN or DIGitize command is executed. After data acquisition, the oscilloscope is placed in the stopped state.

> The range of segment numbers is from 1 to 8888; however, the maximum number of segments allowable is dependent on the current number of points specified using the SEQuential:NPOints command, and the active channel.

`<segments_argument>` ::= 1 to 8888 dependent on the SEQuential:NPOints selection (integer - NR1 format)

Example

```
OUTPUT 707;":SEQUENTIAL:NPOINTS 8000"
OUTPUT 707;":SEQUENTIAL:NSEGMENTS 49"
```

Query

```
:SEQuential:NSEGments?
```

The NSEGments query returns the number of segments to be acquired.

Returned Format

```
[:SEQuential:NSEGments] <segments_argument><NL>
```

<segments_argument> ::= 1 to 8888 dependent on the SEQuential:NPOints selection (integer - NR1 format)

Example

```
OUTPUT 707;":SEQ:NSEG?"
ENTER 707;Seg
PRINT Seg
```

SETup?

Query

```
:SEQuential:SETup?
```

The :SEQuential:SETup query returns the current settings for the SEQuential Subsystem commands.

Returned Format

```
:SEQ:DISP {0 | 1};
  EXCL <exclude_list>;
  INCL <include_list>;
  NPO <points_argument>;
  NSEG <segments_argument>;
  SNUM <segments_argument>    (Sequential Single Shot Mode Only);
 SOURce {CHANnel<n>}<NL>      (Sequential Single Shot Mode Only)
```

`<exclude_list>` ::= a list of previously captured segment numbers separated by commas (integer - NR1 format)

`<include_list>` ::= a list of previously captured segment numbers separated by commas (integer - NR1 format)

`<points_argument>` ::= 4 to 32768 (integer - NR1 format)

`<segments_argument>` ::= 1 to 8888 dependent on the points selected (integer - NR1 format)

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Stp$[300]
OUTPUT 707;":SEQUENTIAL:SETUP?"
ENTER 707;Stp$
PRINT Stp$
```

SNUMber

Command

```
:SEQuential:SNUMber <segments_argument>
```

The :SEQuential:SNUMber command selects one previously defined and captured segment. ACQuire:TYPE NORMal must be specified to select an individual segment.

`<segments_argument>` ::= 1 to 8888 dependent on the number of segments acquired (integer - NR1 format)

Example

```
OUTPUT 707;":SEQUENTIAL:SNUMBER 5"
```

Query

```
:SEQuential:SNUMber?
```

The SNUMber query returns a segment number dependent on the ACQuire:TYPE currently selected as follows.

- ACQuire:TYPE NORMal - returns the segment number selected using the current :SEQuential:SNUMber command.
- ACQuire:TYPE AVERage or ENVelope - returns the total number of segments used to generate the data.

Returned Format

```
[:SEQuential:SNUMber] <segments_argument><NL>
```

`<segments_argument>` ::= 1 to 8888 dependent on the number of segments acquired and ACQuire:TYPE currently selected (integer - NR1 format)

Example

```
OUTPUT 707;":SEQ:SNUM?"
ENTER 707;Seg
PRINT Seg
```

This command corresponds to the front panel display feature that allows display of a single specific segment.

SOURce

Command

```
:SEQuential:SOURce {CHANnel<n>}
```

The :SEQuential:SOURce command selects the channel to be used for the source of the sequential single-shot commands. Data can only be acquired from an active channel. Attempting to acquire data from a channel turned off will cause an error to be generated.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":SEQUENTIAL:SOURCE CHANNEL1"
```

Query

```
:SEQuential:SOURce?
```

The SOURce query returns the currently selected source for the sequential single-shot commands.

Returned Format

```
[:SEQuential:SOURce] {CHANnel<n>}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[30]
OUTPUT 707;":SEQ:SOUR?"
ENTER 707;Src$
PRINT Src$
```

TTAGs?

Query

```
:SEQuential:TTAGs? <segments_argument>
```

The :SEQuential:TTAGs query returns the time difference between segment number one's trigger (0 seconds) and the selected segment's trigger. Value returned is in seconds and is always 12 digits.

Segments are acquired when the RUN or DIGitize command is executed, and the measurement is performed using the most recent sequential acquisition. If the segment specified is not valid, an error is returned.

> The range of segment numbers is from 1 to 8888; however, the actual number of segments acquired are dependent on the current number of segments specified using the SEQuential:NSEGments command.

`<segments_argument>` ::= 1 to 8888 dependent on the SEQuential:NSEGments selection (integer - NR1 format)

Example

```
OUTPUT 707;":SEQUENTIAL:TTAGS? 1000"
```

TTDifference?

Query

```
:SEQuential:TTDifference?
<segments_argument>,<segments_argument>
```

The :SEQuential:TTDifference query returns the time difference between the first segment's trigger and the second segment's trigger. Value returned is in seconds and is always 12 digits.

Segments are acquired when the RUN or DIGitize command is executed, and the measurement is performed using the most recent sequential acquisition. If the segments specified are not valid, an error is returned.

> The range of segment numbers is from 1 to 8888; however, the actual number of segments acquired are dependent on the current number of segments specified using the SEQuential:NSEGments command.

`<segments_argument>` ::= 1 to 8888 dependent on the SEQuential:NSEGments selection (integer - NR1 format)

Example

```
OUTPUT 707;":SEQUENTIAL:TTDIFFERENCE? 1000,1500"
```

20

TIMebase Subsystem

Introduction

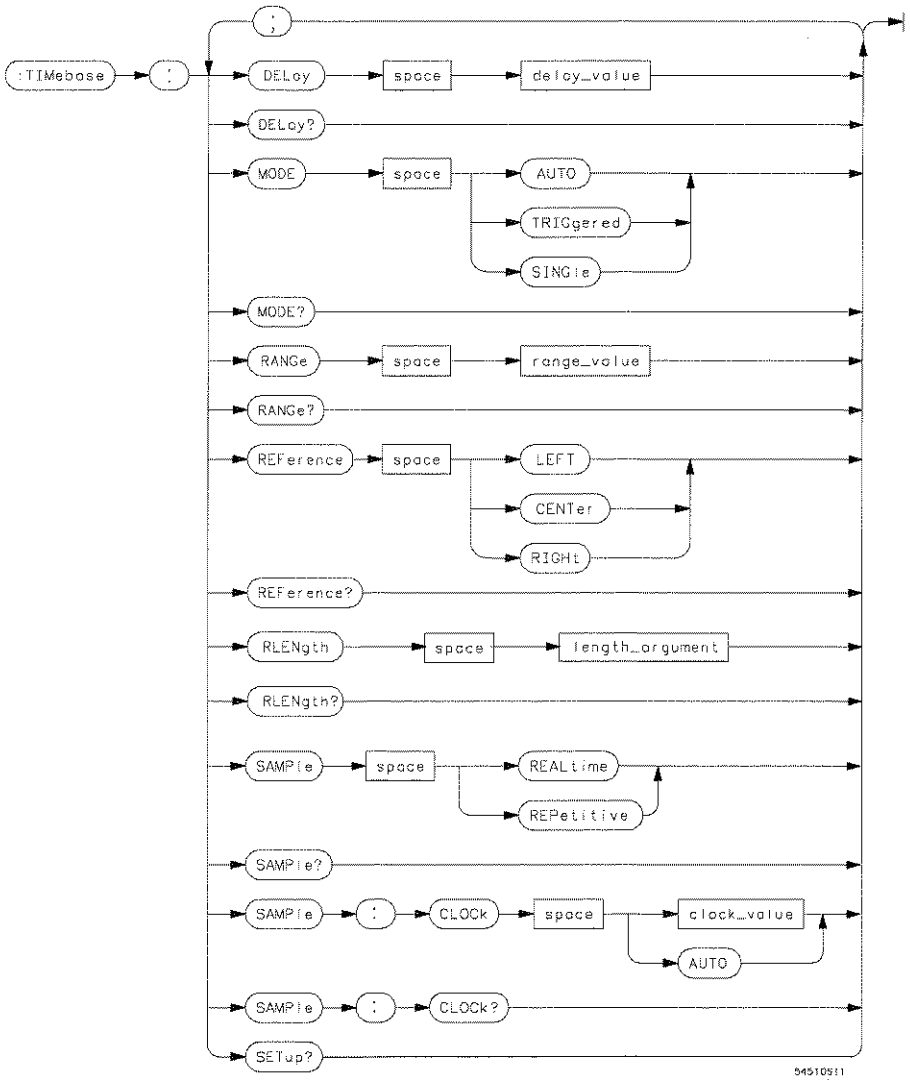
The TIMebase subsystem commands control the horizontal (X-axis) oscilloscope functions corresponding to the horizontal menu on the front panel.

The Timebase subsystem contains the following commands:

DELay
MODE
RANGe
REFerence
RLENgth
SAMPle
SETup

Figure 20-1 lists the syntax diagrams for the Timebase subsystem commands.

Figure 20–1



clock_value	a real number , 10S/s to 2GS/s (in a 1,2.5,5 squence).
delay_value	a real number, the maximum value depends on the sweep range.
length_argument	a real number 4 to 32768 depending on the sample and sequential modes.
range_value	a real number , 5ns through 50 s (in a 1,2,5 squence).

Timebase Subsystem Commands Syntax Diagram

DELay

Command

```
:TIMebase:DELay <delay_value>
```

The :TIMebase:DELay command sets the time base delay. This delay is the internal time between the trigger event and the on-screen delay reference point. The display reference point is the left edge of the display, the right edge of the display, or the center of the display, and is set with the :TIMebase:REFerence command.

`<delay_value>` ::= time in seconds from the trigger to the on screen display reference point. The maximum value depends on the time/division setting (double exponential - NR3 format)

Example

```
OUTPUT 707;":TIMEBASE:DELAY 2MS"
```

Query

```
:TIMebase:DELay?
```

The DELay query returns the current delay value.

Returned Format

```
[:TIMebase:DELay] <delay_value><NL>
```

`<delay_value>` ::= time from trigger to display reference in seconds. (double exponential - NR3 format)

Example

```
OUTPUT 707;":TIM:DEL?"
ENTER 707;D1
PRINT D1
```

MODE

Command

```
:TIMebase:MODE {AUTO | TRIGgered | SINGle}
```

The :TIMebase:MODE command selects the time base mode. This function is the same as the Auto/Trig'd key in the trigger menu and the STOP SINGLE key on the front panel.

If the AUTO mode is selected, a baseline is provided on the display in the absence of a signal. If a signal is present but the oscilloscope is not triggered, the unsynchronized signal is displayed instead of a baseline.

If the TRIGgered mode is selected and no trigger is present, the instrument does not trigger, and the data acquired on the previous trigger remains on the screen until a RUN or DIGitize command is sent.

If the SINGle mode is selected, the screen is cleared and the instrument is stopped. The RUN or DIGitize command arms the trigger, and data is acquired when the trigger is found. The RUN or DIGitize command must be sent to make a single acquisition.

Example

```
OUTPUT 707;":TIMEBASE:MODE TRIGGERED"
```

Query

```
:TIMebase:MODE?
```

The MODE query returns the current mode.

Returned Format

```
[:TIMebase:MODE] {AUTO | TRIGgered | SINGle}<NL>
```

Example

```
DIM Mode$[30]
OUTPUT 707;":TIM:MODE?"
ENTER 707;Mode$
PRINT Mode$
```

RANGe

Command

```
:TIMebase:RANGe <range_value>
```

The :TIMebase:RANGe command sets the full-scale horizontal time in seconds. The RANGe value is ten times the time per division.

`<range_value>` ::= 10 ns to 50 s in a 1,2,5 sequence (exponential - NR3 format)

Example

```
OUTPUT 707;":TIMEBASE:RANGE 100 MS"
```

This example sets the full-scale horizontal time to 100ms (10ms/Div).

Query

```
:TIMebase:RANGe?
```

The RANGE query returns the current full-scale range value.

Returned Format

```
[:TIMebase:RANGe] <range_value><NL>
```

`<range_value>` ::= 10 ns to 50 s (exponential - NR3 format)

Example

```
OUTPUT 707;":TIM:RANG?"
ENTER 707;Rng
PRINT Rng
```

REFerence

Command

```
:TIMebase:REFerence {LEFT | CENTer | RIGHt}
```

The :TIMebase:REFerence command sets the display reference to the left side of the screen, to the right side of the screen, or to the center of the screen.

Example

```
OUTPUT 707;":TIM:REF LEFT"
```

Query

```
:TIMebase:REFerence?
```

The REFerence query returns the current display reference.

Returned Format

```
[:TIMebase:REFerence] {LEFT | CENTer | RIGHt}<NL>
```

Example

```
DIM Rf$[30]
OUTPUT 707;":TIMEBASE:REFERENCE?"
ENTER 707;Rf$
PRINT Rf$
```

RLENgth

Command

```
:TIMebase:RLENgth <length_argument>
```

The :TIMebase:RLENgth command specifies the number of points (time buckets) for each acquisition record.

- When operating in the repetitive mode, the legal setting is 500. If a value is sent that is not a legal value, it is set to 500.
- When operating in the real-time mode (sequential mode OFF), the legal settings are 512, 1024, 2048, 4196, 8192, 16384, and 32768. If a value is sent that is not a legal value, it is set to the next higher allowable setting (e.g., 580 is set to 1024).
- When operating in the real-time mode (sequential mode ON), the legal settings are from 4 to 32768, dependent on the number of segments selected.

Always query the Waveform Subsystem Points value to determine the actual number of acquired time buckets.

`<length_argument>`

::= 500 in the repetitive mode
::= 512, 1024, 2048, 4196, 8192, 16384, or 32768 in the real-time mode (sequential off)
::= 4 to 32768 in the real-time mode (sequential on), depending on the current :SEQuential:NSEGments value (integer - NR1 format)

Example

```
OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE" !select sample mode
OUTPUT 707;":TIMEBASE:RLENGTH 500"
```

Query

```
:TIMebase:RLENgth?
```

The RLENgth query returns the number of time buckets to be acquired.

Returned Format

```
[:TIMebase:RLENgth] <length_argument><NL>
```

<length_argument> ::= 500 in the repetitive mode
::= 512, 1024, 2048, 4196, 8192, 16384, or 32768 in the real-time mode (sequential off)
::= 4 to 32768 in the real-time mode (sequential on), depending on the current :SEQuential:NSEGments value (integer - NR1 format)

Example

```
OUTPUT 707;":TIM:RLEN?"
ENTER 707;Len
PRINT Len
```

The RLENgth command/query is identical to the ACQuire:POINts command/query.

SAMPle

Command

```
:TIMebase:SAMPle {REALtime | REPetitive}
```

The :TIMebase:SAMPle controls the sampling mode of the instrument. Two sample modes are available: REALtime and REPetitive.

- Real-time implies "Single-Shot" capture of data, which means that a complete data record is collected on one trigger event.
- Repetitive implies that the instrument is in a repetitive or equivalent time mode. In this mode the data record is collected over multiple trigger events.

Example

```
OUTPUT 707;":TIMEBASE:SAMPLE REALTIME"
```

Query

```
:TIMebase:SAMPle?
```

The SAMPle query returns the current sampling mode.

Returned Format

```
[:TIMebase:SAMPle] {REALtime | REPetitive}<NL>
```

Example

```
DIM Smp$[30]
OUTPUT 707;":TIM:SAMP?"
ENTER 707;Smp$
PRINT Smp$
```

SAMPle:CLOCk

Command

```
:TIMebase:SAMPle:CLOCk {AUTO | <clock_value>}
```

The :TIMebase:SAMPle:CLOCk command is used to set the oscilloscope's sample rate to a specific value (from 10S/s to 2GS/s in a 1, 2.5, 5 sequence), or set to automatic mode. In automatic mode (default), when the time base is changed the sample rate is automatically changed to maintain 500 points of acquired data on the screen. When a specific sample rate is specified by the user <clock_value>, changes to the time base do not affect sample rate. Note that this may produce a partial display on the screen.

<clock_value> ::= 10S/s to 2GS/s (double exponential - NR3 format)

Example

```
OUTPUT 707;":TIMEBASE:SAMPLE:CLOCK 1000"
```

Query

```
:TIMebase:SAMPle:CLOCk?
```

The SAMPle query returns the current sample rate.

Returned Format

```
[:TIMebase:SAMPle:CLOCk] <clock_value><NL>
```

<clock_value> ::= 10S/s to 2GS/s (double exponential - NR3 format)

Example

```
OUTPUT 707;":TIM:SAMP:CLOC?"
ENTER 707;Smp_rate
PRINT Smp_rate
```

SETup?

Query

```
:TIMebase:SETup?
```

The :TIMebase:SETup query returns the current settings for all the Timebase Subsystem commands.

Returned Format

```
:TIM:DEL <delay_value>;
  MODE {AUTO|TRIG|SING};
  RANGe <range_value>;
  RLEN <length_argument>;
  REF {LEFT|CENT|RIGH};
  SAMP {REAL|REP};
  SAMP:CLOC {AUTO | <clock_value>}<NL>
```

`<delay_value>` ::= time from trigger to display reference in seconds (exponential - NR3 format)

`<range_value>` ::= 5ns to 50s (exponential - NR3 format)

`<length_argument>` ::= 500 in the repetitive mode
::= 512, 1024, 2048, 4196, 8192, 16384, or 32768 in the real-time mode (sequential off)
::= 4 to 32768 in the real-time mode (sequential on), depending on the current :SEQuential:NSEGments value (integer - NR1 format)

`<clock_value>` ::= 10S/s to 2GS/s (exponential - NR3 format)

Example

```
DIM Stp$[300]
OUTPUT 707;":TIM:SET?"
ENTER 707;Stp$
PRINT Stp$
```

21

TRIGger Subsystem

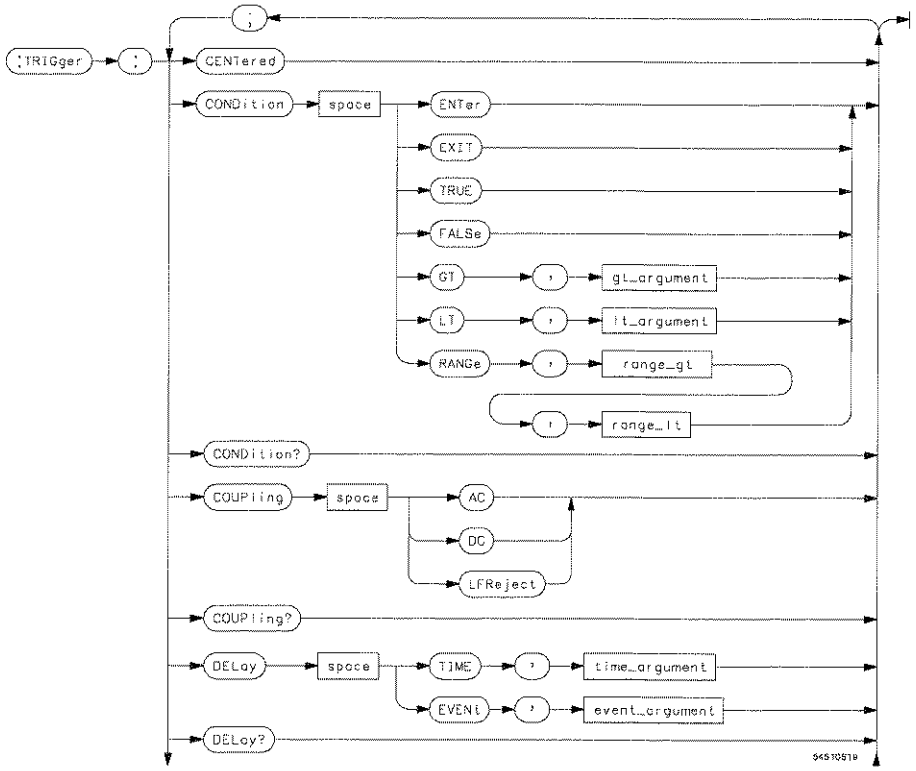
Introduction

The commands in the TRIGger subsystem are used to define the conditions for a trigger. Many of the commands in the Trigger subsystem are valid in more than one of the trigger modes. If the command is a valid command for a trigger mode, that setting is accepted. If the command is not valid for a trigger mode, an error is generated. This subsystem contains the following commands:

CENTered
CONDition
COUPling
DELay
FIELd
GLITch
HOLDoff
LEVel
LINE
LOGic
MODE
NREJect
OCCurrence
PATH
POLarity
QUALify
SETup
SLOPe
SOURce
STANdard

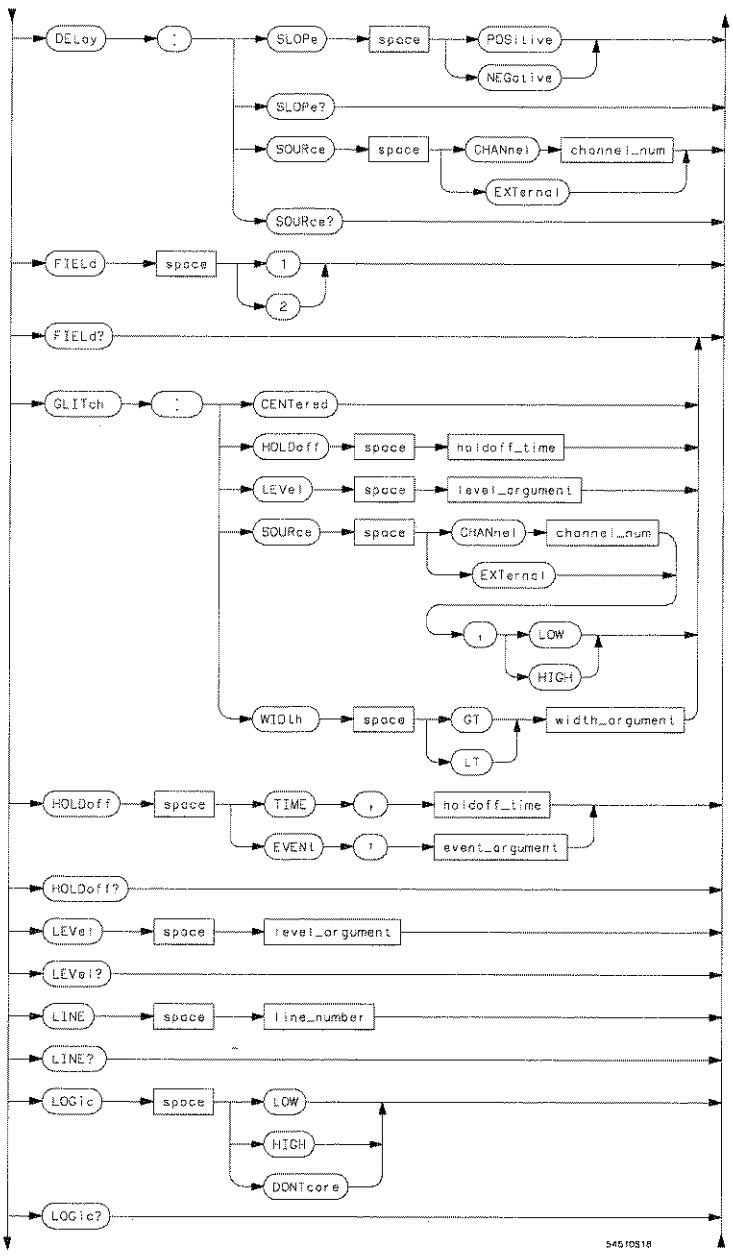
Figure 21-1 lists the syntax diagrams for the Trigger subsystem commands.

Figure 21–1



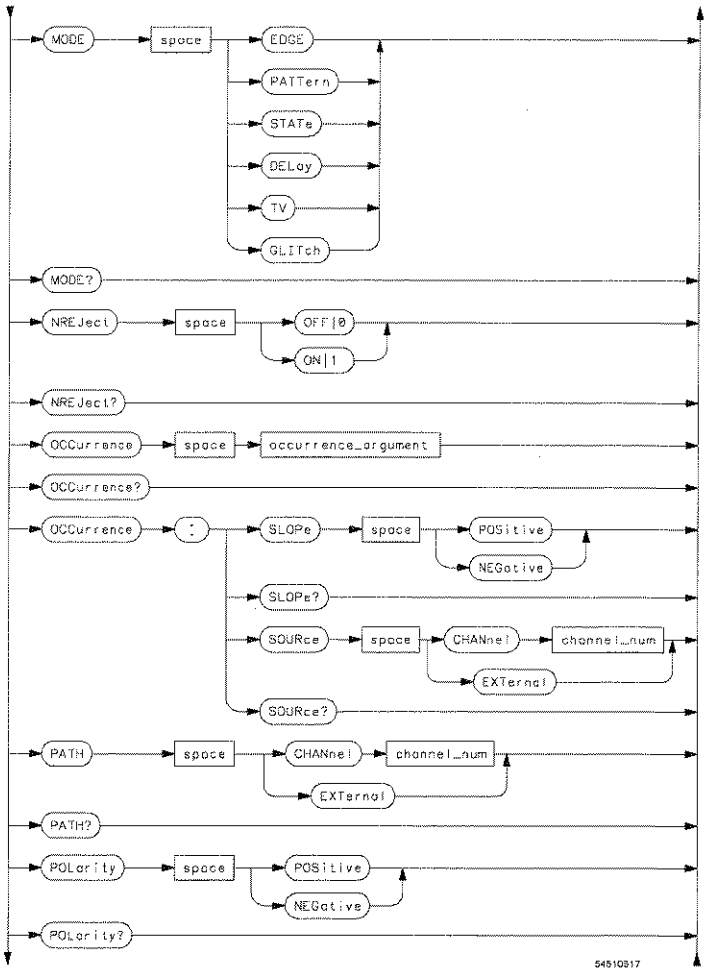
Trigger Subsystem Commands Syntax Diagram

Figure 21–1



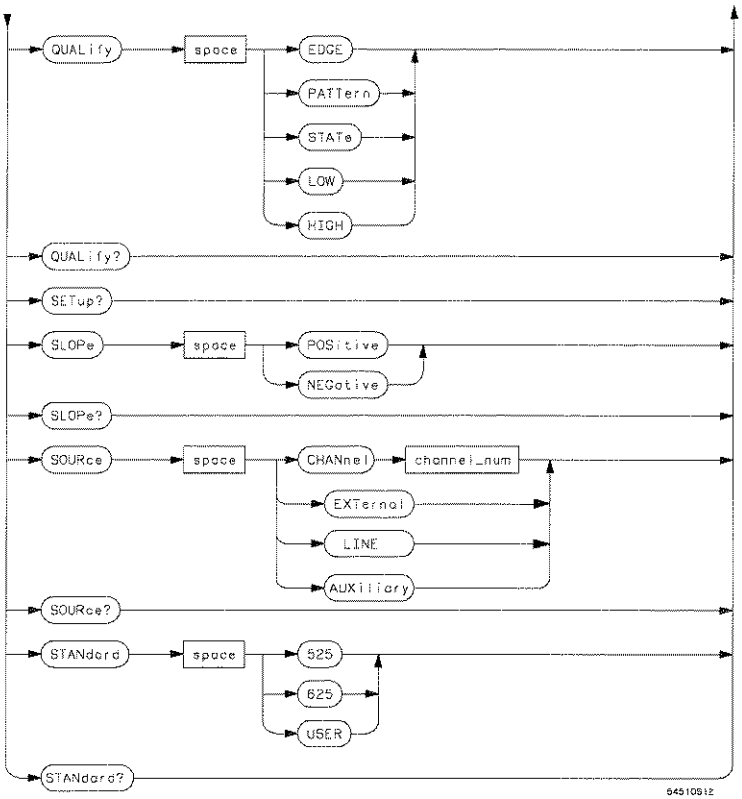
Trigger Subsystem Commands Syntax Diagram (continued)

Figure21–1



Trigger Subsystem Commands Syntax Diagram (continued)

Figure21–1



Trigger Subsystem Commands Syntax Diagram (continued)

Figure 21-1

attenuation_factor =	a real number, 0.9 to 1000.
channel_num =	an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).
event_argument =	an integer, 1 to 16000000.
gt_argument =	a time value, 20 ns to 160 ms.
holdoff_time =	a time value, 40 ns to 320 ms.
level_argument =	a real number specifying the trigger level in volts.
line_argument =	an integer, 1 to 625 (depending on the video STANDARD selected).
lt_argument =	a time value, 20 ns to 160 ms.
range_gt =	a time value, 20 ns to 159.999 ms (the value must be less than range_lt).
range_lt =	a time value, 30 ns to 160 ms (the value must be greater than range_gt).
time_argument =	a time value, 30 ns to 160 ms.
occurrence_argument =	an integer, 1 to 16000000.
width_argument =	a time value, 5 ns to 160 ms.

Trigger Subsystem Commands Syntax Diagram (continued)

Trigger Mode

Make sure the instrument is in the proper trigger mode for the command you want to send. One method of insuring the instrument is in the proper trigger mode is to send the :TRIGger:MODE command in the same program message as the parameter to be set. For example,

":TRIGGER:MODE TV;LEVEL 200 MV"

places the instrument in the TV Trigger Mode and sets the selected source's trigger level to 200 mV. This process is necessary because the LEVel command is also valid for some of the other trigger modes.

The trigger modes are described on the next few pages prior to the descriptions of the individual commands. Table 21-1 lists the different TRIGger subsystem commands that are available for each trigger mode.

> Auto or triggered mode is selected with the TIMebase:MODE command. For more information, refer to the "Timebase Subsystem" chapter.

Table 21-1

Valid Commands for Specific Trigger Modes

EDGE	PATTERN	STATE	DELAY	TV	GLITCH
CENTered	CENTered	CENTered	CENTered	CENTered	COUPling
COUPling	CONDition	CONDition	CONDition	CONDition	GLITch:CENTered
HOLDoff	COUPling	COUPling	COUPling	COUPling	GLITch:HOLDoff
LEVel	HOLDoff	HOLDoff	DELay	FIELd	GLITch:LEVel
NREJect	LEVel	LEVel	DELay:SLOPe	HOLDoff	GLITch:SOURce
SLOPe	LOGic	LOGic	DELay:SOURce	LEVel	GLITch:WIDth
SOURce	NREJect	NREJect	LEVel	LINE	NREJect
	PATH	PATH	LOGic	NREJect	
		SLOPe	NREJect	OCCurrence	
		SOURce	OCCurrence	OCCurrence:SLOPe	
			OCCurrence:SLOPe	POLarity	
			OCCurrence:SOURce	QUALify	
			PATH	SOURce	
			QUALify	STANdard	
			SLOPe		
			SOURce		

The EDGE Trigger Mode

The Edge Trigger Mode is the easiest trigger mode to understand and use from the front panel or over the remote interface, because it has the least number of parameters to be set. This explanation of the trigger mode commands follow the front-panel keys very closely. Refer to the *HP 54520A/54522A/54540A/54542A Front-Panel Reference* for further explanations of the trigger operation.

In the Edge Trigger Mode you must set the trigger source using the :TRIGger:SOURce command. This selects the source that the oscilloscope triggers on. The argument for the :TRIGger:SOURce command is CHANnel<n>, where n = 1 or 2 (HP 54520A/54522A) or 1 through 4 (HP 54540A/54542A); AUXiliary; LINE; or EXTernal (HP 54520A/54522A only).

After setting the trigger source, you need to set the trigger level. This value is set using the :TRIGger:LEVel command and can be set for each trigger source. The trigger level values that are set in the Edge Trigger Mode are used for all modes except the TV and Glitch Trigger Mode. Any levels set in the PATTern, STATe, or DELay modes set the levels for the EDGE mode.

The actual edge that creates the trigger is set with the :TRIGger:SLOPe command. This command can be set to POSitive or NEGative for each of the sources.

The last setting in the Edge Trigger Mode is the Trigger Holdoff value. This value is used only for the EDGE mode.

The Pattern Trigger Mode

This description of the Pattern Trigger Mode follows the front-panel keys very closely. There are additional parameters in this mode that are not used in the Edge Trigger Mode and one parameter that is carried over from the Edge Trigger Mode. The one parameter that carries over is LEVel. If the level command is used in this mode it also changes the level value for that source in the Edge Trigger Mode. In this trigger mode, the :TRIGger:LEvel command defines the voltage that determines if the input voltage is a logic high or logic low for the logic triggers.

The logic pattern for the Pattern Trigger Mode is set using the PATH and LOGic commands. The PATH command specifies which of the two inputs is selected for the logic pattern. Once the path is selected, the pattern can be set using the LOGic command. The LOGic command uses the arguments HIGH, LOW, and DONtcare to set the "trigger on" bit pattern.

The CONDition command sets the "when" field on the front panel. This command is used in several of the trigger modes; therefore, it has parameters that are not valid in this mode. The valid parameters for the CONDition command in the Pattern Trigger Mode are ENTer, EXIT, GT (Greater Than), LT (Less Than), and RANGe.

When the TRIGger:CONDition GT or LT option is used, a time value must be sent to define the limit. When the RANGe option is used, two time values must be sent to define the lower and upper limit. The correct syntax for the RANGE option is :TRIGger:CONDition,<range_low>,<range_high>.

The holdoff time is set in the Pattern Trigger Mode with the :TRIGger:HOLDoff command.

The State Trigger Mode

When the State Trigger Mode is selected, the :TRIGger:SOURce command is used to select the clock source. The syntax for selecting channel 2 as the clock source is :TRIGger:SOURce CHANnel2.

After the clock source is selected, the correct edge for the clock can be selected using the :TRIGger:SLOPe command, which can be set to NEGative or POSitive.

The :TRIGger:PATH command can be used with the :TRIGger:LOGic command to set the three bit logic pattern to qualify the clock trigger. These commands can be sent using the syntax :TRIGGER:PATH CHAN2;:TRIGger:LOGic LOW, or :TRIGger:PATH CHAN2; LOGic LOW.

In this trigger mode, the :TRIGger:LEVel command defines the voltage that determines if the input voltage is a logic high or a logic low for the logic triggers.

The :TRIGger:CONDition command sets the "is/is not present" state using the parameters TRUE for "is present" and FALSe for "is not present."

In the State Trigger Mode, a holdoff value can be set as in all other modes.

The Delay Trigger Mode

In the Delay Trigger Mode, the TRIGger:QUALify command can be used to select the EDGE, PATTern, or STATe mode as a qualifier. When the EDGE qualifier is selected, all Edge parameters and commands can be used to set the Source and Slope. When the PATTern qualifier is selected, all Pattern commands can be used to set the pattern mode parameters. When the STATe qualifier is selected, all State commands can be used to set the state mode parameters.

The next settings (in front panel order) are the delay settings. The DELay command is used to set the Time or Count parameter and the amount of delay. To set the delay to time use the command :TRIGger:DELay TIME,<time>. To set the delay to count use the command :TRIGger:DELay EVENt<number_events>.

If the trigger delay is set to Event (count) you can then set the delay source and slope. To set the delay source, use the command :TRIGger:DELay:SOURce CHANnel2. To set the delay slope, use the command TRIGger:DELay:SLOPe POSitive.

The values within the front-panel "trigger on" field are set with the OCCurrence command. To set the number of occurrences to 3333, use the command syntax TRIGger:OCCurrence 3333. To set the source for the number of occurrences to channel 2, use the command syntax :TRIGger:OCCurrence:SOURce CHANnel2. To set the slope of the trigger occurrence to negative, use the command syntax :TRIGger:OCCurrence:SLOPe NEGative.

The TV Trigger Mode

The TV Trigger Mode is used for triggering on clamped television signals. This mode allows you to select one of the TV signal frames and one of the lines within that frame.

Once the TV Trigger Mode is selected, the Television Signal Standard can be selected using the :TRIGger:STANdard command. The three parameters for this command are 525, 625, and USER. Any of these modes allow you to select the trigger level and the source of the trigger signal.

The trigger level is set by sending the command :TRIGger:LEVel <value>.

The trigger source is set by sending the :TRIGger:SOURce command. This command allows you to select channel 1 or channel 2 as the trigger source.

With the standard set to 525 or 625, the commands that can be used are POLarity, FIELd and LINE. The POLarity command sets the edge for the trigger. This edge can be set to NEGative or POSitive. The FIELd command selects the first or second field of the television signal. The LINE command specifies the horizontal line in which the instrument will trigger on. Refer to the LINE command to determine the correct values.

The HOLDoff value can also be set in the TV trigger mode, as in all modes.

When the USER (user defined) standard is selected, the source and level are set in the same manner as before.

The QUALify command is used to set the "qualify on" field. This command can be set to HIGH or LOW.

The edge defined by the QUALify command must occur within the range of time values that are displayed in the front panel field. The TRIGger:CONDition RANGe command sets the greater than and less than time values. In order to actually generate a trigger, the qualified conditions must be met within the specified time. To set the time values, send the command :TRIGger:CONDition RANge, <greater_than_value>, <less_than_value>.

The field, "trigger on," is set with the OCCurrence command and OCCurrence:SLOPe command. To set the number of occurrences, send the command :TRIGger:OCCurrence <number>. To set the slope for the occurrences, send the command :TRIGger:OCCurrence:SLOPe <slope>. The slope can be set to POSitive or NEGative.

The description for each command tells you in which modes that command is valid.

The Glitch Trigger Mode

When the Glitch Trigger Mode is selected, the :TRIGger:GLITch:SOURce command is used to select both the trigger source and the pattern state. The syntax for selecting channel 2 as the trigger source, and high as the state, is :TRIGger:GLITch:SOURce CHANnel2,HIGH.

The trigger level is set by sending the command :TRIGger:GLITch:LEVel <value>.

The condition under which a trigger is generated is set using the TRIGger:GLITch:WIDTh command. This command is used to specify that the trigger will be generated if the glitch width is less than (LT) or greater than (GT) a specified time value. The correct syntax for the WIDTh option is :TRIGger:GLITch: {GT | LT},<width>.

The holdoff time is set in the Glitch Trigger Mode with the :TRIGger:GLITch:HOLDoff command.

CENTered

Command

```
:TRIGger:CENTered
```

The :TRIGger:CENTered command sets the trigger level for all channels to the center of the display. Changes in :CHANnel:OFFSet (vertical position) will cause the source trigger level to change.

When TRIGger:SOURce AUXiliary is selected, trigger level is set to 0. This command is not valid when TRIGger:SOURce is set to LINE or EXTernal.

> To return to the ADJust mode, specify a level with the TRIGger:LEVel command.

Example

```
OUTPUT 707;":TRIGGER:CENTERED"
```

CONDition

Command

```
:TRIGger:CONDition
{ENTer|EXIT|TRUE|FALSe|GT,<gt_argument>|LT,<lt_
argument>|RANGe,<range_gt>,<range_lt>}
```

The :TRIGger:CONDition command is valid in the PATTern, STATe, DELay, and TV trigger modes. The time values entered using this command are rounded to the nearest 10 ns.

In the Pattern Trigger Mode, the valid arguments for the CONDition command are ENTer, EXIT, GT, LT, RANGe.

- In the **`Pattern Trigger Mode`**, the CONDition command is used to specify whether the trigger is generated upon entering (ENTer) or leaving (EXIT) the specified logic pattern. Also, the CONDition command specifies whether the pattern must be present for a specified amount of time. The time in the pattern trigger mode can be greater than a value (GT), less then a value (LT), or between two values (RANGe). These settings can also be specified from the front panel using the "when" key in the Pattern Trigger Mode.
- In the **`State Trigger Mode`**, the valid parameters for the CONDition command are TRUE (is present) and FALSe (is not present).
- In the **`Delay Trigger Mode`**, the CONDition command is valid when PATTern or STATe is selected as the qualifier. All arguments for this command that are valid in the Pattern or State Trigger Modes are valid here.
- In the **`TV Trigger Mode`**, the CONDition command is used to set the range of time values on which the trigger occurs. This command is only valid in the "user defined" mode.

Example

```
OUTPUT 707;":TRIGGER:CONDITION EXIT"
```

Query `:TRIGger:CONDition?`

The CONDition query returns the currently selected condition for the currently selected mode.

Returned Format `[:TRIGger:CONDition] <argument><NL>`

`<argument>` ::= {ENTer | EXIT | GT,<gt_argument> | LT,<lt_argument> | RANGe,<range_gt,<range_lt>} in PATTERN or DELay with QUALify:PATTern selected
::= {TRUE | FALSe} (in STATE or DELay with QUALify:STATe selected
::= RANGe,<range_gt>,<range_lt> (in TV mode)

`<gt_argument>` ::= 20 ns to 160 ms (exponential - NR3 format)

`<lt_argument>` ::= 20 ns to 160 ms (exponential - NR3 format)

`<range_gt>` ::= 20 ns to 159.999 ms (must be less than range_lt) (exponential - NR3 format)

`<range_lt>` ::= 30 ns to 160 ms (must be greater than range_gt) (exponential - NR3 format)

Example

```
DIM Con$[50]
OUTPUT 707;":TRIG:COND?"
ENTER 707;Con$
PRINT Con$
```

COUPling

Command

```
:TRIGger:COUPling {AC | DC | LFReject}
```

The :TRIGger:COUPling command selects the input coupling for the selected trigger source. The coupling can be set to AC, DC or LFReject. This command is valid only in the Edge Trigger Mode (in all other modes an error is returned). When a mode other than edge is selected, coupling automatically changes to DC.

Example

```
OUTPUT 707;":TRIGGER:COUPLING DC"
```

Query

```
:TRIGger:COUPling?
```

The COUPling query returns the current selection.

Returned Format

```
[:TRIGger:COUPling] {AC | DC | LFReject}<NL>
```

Example

```
DIM Mode$[50]
OUTPUT 707;":TRIG:COUP?"
ENTER 707;Mode$
PRINT Mode$
```

DELay

Command

```
:TRIGger:DELay {TIME,<time_argument> |
EVENt,<event_argument>}
```

The :TRIGger:DELay command is valid only in the **Delay Trigger Mode**. This command sets a delay value in either time or number of events. In the time delay mode, this command specifies the delay value in seconds. In the events delay mode, this command specifies the delay value in number of trigger events.

`<time_argument>` ::= amount of delay from 30 ns to 160 ms (exponential - NR3 format)

`<event_argument>` ::= number of events from 1 to 16000000 (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:MODE DELAY"    !select trigger mode
OUTPUT 707;":TRIGGER:DELAY TIME,1.23E-01"
```

Query

```
:TRIGger:DELay?
```

The DELay query returns the current delay value.

Returned Format

```
[:TRIGger:DELay] {TIME,<time_argument> |
EVENt,<event_argument>}<NL>
```

`<time_argument>` ::= amount of delay from 30 ns to 160 ms (exponential - NR3 format)

`<event_argument>` ::= number of events from 1 to 16000000 (integer - NR1 format)

Example

```
DIM Value$[50]
OUTPUT 707;":TRIG:DEL?"
ENTER 707;Value$
PRINT Value$
```

DELay:SLOPe

Command

```
:TRIGger:DELay:SLOPe {POSitive | NEGative}
```

The :TRIGger:DELay:SLOPe command selects the edge that will be counted by the delay command. The parameters for this command are NEGative or POSitive. This command is only valid in the **Delay Trigger Mode**.

Example

```
OUTPUT 707;":TRIGGER:MODE DELAY"     !select trigger mode
OUTPUT 707;":TRIGGER:DELAY:SLOPE POSITIVE"
```

Query

```
:TRIGger:DELay:SLOPe?
```

The DELay:SLOPe query returns the current delay slope.

Returned Format

```
[:TRIGger:DELay:SLOPe] {POSitive | NEGative}<NL>
```

Example

```
DIM Tos$[50]
OUTPUT 707;":TRIG:DEL:SLOP?"
ENTER 707;Tos$
PRINT Tos$
```

DELay:SOURce

Command

```
:TRIGger:DELay:SOURce {CHANnel<n> | EXTernal}
```

The :TRIGger:DELay:SOURce command selects the source that is counted by the delay command. EXTernal is only valid when the oscilloscope is an HP 54520A/54522A. This command is only valid in the **Delay Trigger Mode**.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:MODE DELAY"    !select trigger mode
OUTPUT 707;":TRIGGER:DELAY:SOURCE CHANNEL2"
```

Query

```
:TRIGger:DELay:SOURce?
```

The DELay:SOURce query returns the source of the delay in the Delay Trigger Mode.

Returned Format

```
[:TRIGger:DELay:SOURce] {CHANnel<n> | EXTernal}
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Tos$[50]
OUTPUT 707;":TRIG:DEL:SOUR?"
ENTER 707;Tos$
PRINT Tos$
```

FIELd

Command

```
:TRIGger:FIELd {1 | 2}
```

The :TRIGger:FIELd command is only valid in the **TV Trigger Mode**. This command selects the field of the TV signal when the STANdard is set to 525 or 625. The valid parameters for this command are 1 or 2.

If the :TRIGger:FIELd command is set in any other trigger mode, error -221, "Settings conflict," is placed in the error queue.

Example

```
OUTPUT 707;":TRIGGER:MODE TV"     !select trigger mode
OUTPUT 707;":TRIGGER:FIELD 2"
```

Query

```
:TRIGger:FIELd?
```

The FIELd query returns the current setting of the FIELD command.

Returned Format

```
[:TRIGger:FIELd] {1 | 2}<NL>
```

Example

```
OUTPUT 707;":TRIG:FIEL?"
ENTER 707;F
PRINT F
```

GLITch

The :TRIGger:GLITch Trigger Mode. All of the commands in the Glitch Trigger subsystem are not used in any of the other trigger modes. This subsystem contains the following commands:

- GLITch:CENTered
- GLITch:HOLDoff
- GLITch:LEVel
- GLITch:SOURce
- GLITch:WIDth

GLITch:CENTered

Command

```
:TRIGger:GLITch:CENTered
```

The :TRIGger:GLITch:CENTered command sets the trigger level to the centered mode (at the center of the graph). This command is not valid when the external source is selected (HP 54520A/54522A). This command is only valid in the Glitch Trigger Mode.

> To return to the ADJust mode, specify a level with the TRIGger:GLITch:LEVel command.

Example

```
OUTPUT 707;":TRIGGER:GLITCH:SOURCE CHANNEL1,HIGH"     !select
trigger source and state
OUTPUT 707;":TRIGGER:GLITCH:CENTERED"
```

GLITch:HOLDoff

Command

```
:TRIGger:GLITch:HOLDoff <holdoff_time>
```

The :TRIGger:GLITch:HOLDoff command is used to enter the amount of time the trigger circuit is disabled after the trigger event has occurred. This command is only valid in the Glitch Trigger Mode.

`<holdoff_time>` ::= 40 ns to 320 ms rounded to nearest 20 ns increment. (exponential - NR3 format)

Example

```
OUTPUT 707;":TRIGGER:GLITCH:HOLDOFF 216 US"
```

Query

```
:TRIGger:GLITch:HOLDoff?
```

The GLITch:HOLDoff query returns the value of the holdoff for the Glitch Trigger Mode.

Returned Format

```
[:TRIGger:GLITch:HOLDoff] <holdoff_time><NL>
```

`<holdoff_time>` ::= 40 ns to 320 ms (exponential - NR3 format)

Example

```
OUTPUT 707;":TRIG:GLIT:HOLD?"
ENTER 707;Ho
PRINT Ho
```

GLITch:LEVel

Command

```
:TRIGger:GLITch:LEVel <level_argument>
```

The :TRIGger:GLITch:LEVel command sets the trigger level voltage for the glitch trigger mode. This command is only valid in the Glitch Trigger Mode.

<level_argument> ::= for internal triggers, ±1.5 x full-scale voltage from center screen. (exponential - NR3 format)

::= for external triggers, ±2 volts with probe attenuation at 1:1 (exponential - NR3 format)

Examples

```
OUTPUT 707;":TRIGGER:GLITCH:LEVEL .30"
OUTPUT 707;":TRIGGER:GLITCH:LEV 300MV"
OUTPUT 707;":TRIGGER:GLITCH:LEV 3E-1"
```

Query

```
:TRIGger:GLITch:LEVel?
```

The GLITch:LEVel query returns the trigger level of the Glitch Trigger Mode.

Returned Format

```
[:TRIGger:GLITch:LEVel] <level_argument><NL>
```

<level_argument> ::= trigger level in volts (exponential - NR3 format)

Example

```
OUTPUT 707;":TRIG:GLIT:LEV?"
ENTER 707;Tlevel
PRINT Tlevel
```

GLITch:SOURce

Command

```
:TRIGger:GLITch:SOURce {CHANnel<n> |
EXTernal},{HIGH | LOW}
```

The :TRIGger:GLITch:SOURce command selects the channel and pattern state that will actually produce the trigger. EXTernal is only valid when the oscilloscope is an HP 54520A/54522A. This command is only valid in the Glitch Trigger Mode.

The state for the source selected can be specified as either HIGH or LOW, where HIGH is higher than the current GLITch:LEVel value, and LOW is lower than the current GLITch:LEVel value.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:GLITCH:SOURCE CHANNEL2,LOW"
```

Query

```
:TRIGger:GLITCH:SOURce?
```

The GLITch:SOURce query returns the current source for the Glitch Trigger Mode.

Returned Format

```
[:TRIGger:GLITch:SOURce] {CHANnel<n> | EXTernal},{HIGH | LOW}
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[30]
OUTPUT 707;":TRIG:GLIT:SOUR?"
ENTER 707;Src$
PRINT Src$
```

GLITch:WIDth

Command

```
:TRIGger:GLITch:WIDTh {GT | LT},<width_argument>
```

The :TRIGger:GLITch:WIDTh command is used to specify whether the trigger is generated when the pattern is present for greater than (GT), or less than (LT) a specified amount of time.

`<width_argument>` ::= 5 ns to 160 ms (exponential - NR3 format)

Example

```
OUTPUT 707;":TRIGGER:GLITCH:WIDTH GT,33ms"
```

Query

```
:TRIGger:GLITch:WIDTh?
```

The GLITch:WIDTh query returns the current condition and time value for the Glitch Trigger Mode.

Returned Format

```
[:TRIGger:GLITch:WIDTh] {GT | LT},<width_argument><NL>
```

`<width_argument>` ::= 5 ns to 160 ms (exponential - NR3 format)

Example

```
OUTPUT 707;":TRIG:GLIT:WIDT?"
ENTER 707;Con
PRINT Con
```

HOLDoff

Command

```
:TRIGger:HOLDoff {{TIME,<holdoff_time>} |
{EVENt,<event_argument>}}
```

The :TRIGger:HOLDoff command is valid in the **Edge**, **Pattern**, **State**, and **TV Trigger Modes**. The HOLDoff command enters a holdoff value in terms of time or events.

`<holdoff_time>` ::= 40 ns to 320 ms rounded to nearest 20 ns increment (exponential - NR3 format)

`<event_argument>` ::= 1 to 16000000 (integer - NR1 format)

Examples

```
OUTPUT 707;":TRIGGER:HOLDOFF TIME,216 US"
OUTPUT 707;":TRIGGER:HOLDOFF EVENT,10"
```

Query

```
:TRIGger:HOLDoff?
```

The HOLDoff query returns the value of the holdoff for the current mode.

Returned Format

```
[:TRIGger:HOLDoff] {{TIME,<holdoff_time>} |
{EVENt,<event_argument>}}<NL>
```

`<holdoff_time>` ::= 40 ns to 320 ms (exponential - NR3 format)

`<event_argument>` ::= 1 to 16000000 (integer - NR1 format)

Example

```
DIM Ho$[50]
OUTPUT 707;":TRIG:HOLD?"
ENTER 707;Ho$
PRINT Ho$
```

LEVel

Command

```
:TRIGger:LEVel <level_argument>
```

The :TRIGger:LEVel command sets the trigger level voltage of the active trigger. This command can be sent in any mode; however, only two separate levels are stored. One value is kept for the TV Trigger Mode and another value is kept for all other modes (except Glitch). If you are in the Pattern Trigger Mode and set a trigger level value, that level is also used for the Edge, State, and Delay Trigger Modes.

`<level_argument>`
::= for internal triggers, ±1.5 x full-scale voltage from center screen.
::= for external triggers, ±2 volts with probe attenuation at 1:1
(exponential - NR3 format)

Examples

```
OUTPUT 707;":TRIGGER:LEVEL .30"
OUTPUT 707;":TRIGGER:LEVEL 300MV"
OUTPUT 707;":TRIGGER:LEVEL 3E-1"
```

Query

```
:TRIGger:LEVel?
```

The LEVel query returns the trigger level of the current trigger mode.

Returned Format

```
[:TRIGger:LEVel] <level_argument><NL>
```

`<level_argument>`
::= trigger level in volts (exponential - NR3 format)

Example

```
OUTPUT 707;":TRIG:LEV?"
ENTER 707;Tlevel
PRINT Tlevel
```

LINE

Command

```
:TRIGger:LINE <line_number>
```

The :TRIGger:LINE command specifies the horizontal line in which the instrument will trigger on. The LINE command is valid in the **TV Trigger Mode** when the STANdard selected is 525 or 625. If one of the these standards is selected when the TV Trigger Mode is entered, the line value is set in that standard and selected field.

`<line_number>` ::= 1 to 625 (depends on STANdard and FIELd selection) (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:MODE TV"          !select trigger mode
OUTPUT 707;":TRIGGER:STANDARD 525"     !select TV signal
standard
OUTPUT 707;":TRIGGER:LINE 22"
```

Query

```
:TRIGger:LINE?
```

The LINE query returns the current line of the selected standard.

Returned Format

```
[:TRIGger:LINE] <line_number><NL>
```

`<line_number>` ::= 1 to 625 (depends on STANdard and FIELd selection) (integer - NR1 format)

Example

```
OUTPUT 707;":TRIG:LINE?"
ENTER 707;Ln
PRINT Ln
```

LOGic

Command

```
:TRIGger:LOGic {HIGH | LOW | DONTcare}
```

The :TRIGger:LOGic command is valid in the **Pattern** and **State Trigger Modes**, as well as the **DELay Trigger Mode** when qualifying by PATTern or STATe. The LOGic command specifies the relationship between the signal and the defined voltage level that must exist before that part of the pattern is considered valid. If the signal on a selected path is greater than the trigger level, that signal is considered HIGH. If the signal is less than the trigger level, it is considered LOW.

Example

```
OUTPUT 707:":TRIGGER:PATH CHANNEL1"       !select channel
OUTPUT 707;":TRIGGER:MODE DELAY"          !select trigger mode
OUTPUT 707;":TRIGGER:QUALIFY PATTERN"     !qualify trigger
OUTPUT 707;":TRIGGER:LOGIC DONTCARE"
```

Query

```
:TRIGger:LOGic?
```

The LOGic query returns the last specified logic level of the currently enabled path.

Returned Format

```
[:TRIGger:LOGic] {HIGH | LOW | DONTcare}<NL>
```

Example

```
DIM L$[50]
OUTPUT 707;":TRIG:LOG?"
ENTER 707;L$
PRINT L$
```

MODE

Command

```
:TRIGger:MODE {EDGE | PATTern | STATe | DELay |
TV | GLITch}
```

The :TRIGger:MODE command selects the trigger mode. This command can be sent from any trigger mode.

Six trigger modes provide many distinctive techniques to trigger and capture data. Select the desired trigger mode using the following guidelines:

:EDGe Provides simple edge triggering. Easiest mode to understand and use. Use the following TRIGger commands in the order presented to setup EDGe triggering.

- :SOURce - Use to select the channel that the instrument will trigger on. See TRIGger:SOURce for more information.
- :LEVel - Use to select the trigger level that the instrument will trigger on. Can be set for each trigger source. See TRIGger:LEVel and CENTered for more information.
- :SLOPe - Use to select the actual edge that will create the trigger. Can be set for each trigger source. See TRIGger:SLOPe for more information.
- :NREJect - Use to select noise reject on or off. Can be set for each trigger source. See TRIGger:NREJect for more information.
- :HOLDoff - Use to select the time or number of events to holdoff after the trigger event. See TRIGger:HOLDoff for more information.

:PATTern Defines up to four patterns for the instrument to recognize, and then generate a trigger event. Use the following TRIGger commands in the order presented to setup PATTern triggering.

Up to four logic patterns are defined using the PATH and LOGic commands.

- :PATH - Use to select which of the four inputs are used for the logic pattern. See TRIGger:PATH for more information.
- :LOGic - Use to select the "logic" conditions that must be satisfied. Level is set for each active path. See TRIGger:LOGic for more information.
- :LEVel - Use to select the trigger level that the instrument will use to determine logic level. Level is set for each active path. See TRIGger:LEVel and CENTered for more information.

- :NREJect - Use to select noise reject on or off. Can be set for each trigger source. See TRIGger:NREJect for more information.
- :CONDition - Use to select the "when" conditions that must be satisfied before a trigger event is generated. See TRIGger:CONDition for more information.
- :HOLDoff - Use to select the time or number of events to holdoff after the trigger event. See TRIGger:HOLDoff for more information.

:STATe Similiar to PATTern mode, except one input is selected as a clock edge and the other three inputs define a pattern. When the pattern becomes true, the instrument triggers on the next clock edge. Use the following TRIGger commands in the order presented to setup STATe triggering.

- :SOURce - Use to select the channel for the clock source. See TRIGger:SOURce for more information.
- :SLOPe - Use to select the edge for the clock source. See TRIGger:SLOPe for more information.
- Up to three logic patterns are defined using the PATH and LOGic commands.
  - :PATH - Use to select which of the three inputs are used for the logic pattern. See TRIGger:PATH for more information.
  - :LOGic - Use to select the "logic" conditions that must be satisfied. Level is set for each active path. See TRIGger:LOGic for more information.
- :LEVel - Use to select the trigger level that the instrument will use to determine logic level. Level is set for each active path. See TRIGger:LEVel and CENTered for more information.
- :NREJect - Use to select noise reject on or off. Can be set for each trigger source. See TRIGger:SENSitivity for more information.
- :CONDition - Use to select the true/false condition that must be satisfied before a trigger event is generated. See TRIGger:CONDition for more information.
- :HOLDoff - Use to select the time or number of events to holdoff after the trigger event. See TRIGger:HOLDoff for more information.

:DELay Qualifies on a signal (edge, pattern, or state), delays for a period of time or occurrence of events, and then enables a trigger event on a selected edge from any source. Use the following TRIGger commands in the order presented to setup DELay triggering.

- :QUALify - Use to select which mode (EDGe, PATTern, or STATe) to qualify the trigger before a delay is defined. Selection of these modes is described previously. See TRIGger:QUALify for more information.
- :DELay - Use to select the type (time or event) and amount of delay. If events are selected, the source and slope must also be specified. See TRIGger:DELay for more information.
- :OCCurrence - Use to select the source, slope, and number of trigger events that occur before the sweep is triggered. See TRIGger:OCCurrence for more information.

:TV Used for triggering on clamped television signals. This mode allows selection of one TV signal frame and one of the lines within that frame. Use the following TRIGger commands in the order presented to setup TV triggering.

- :STANdard - Use to select the TV standard signal. See TRIGger:STANdard for more information.
- :SOURce - Use to select the channel that the instrument will trigger on. See TRIGger:SOURce for more information.
- :LEVel - Use to select the trigger level that the instrument will trigger on. See TRIGger:LEVel and CENTered for more information.
- :NREJect - Use to select noise reject on or off. Can be set for each trigger source. See TRIGger:NREJect for more information.
- :POLarity - Use to select the edge that will create the trigger. See TRIGger:POLarity for more information.
- :FIELd - Use to select the field that will create the trigger. See TRIGger:FIELd for more information.
- :LINe - Use to select the line in the field that will create the trigger. See TRIGger:LINe for more information.
- :HOLDoff - Use to select the time or number of events to holdoff after the trigger event. See TRIGger:HOLDoff for more information.
- :QUALify - Use to select the qualify on field. See TRIGger:QUALify for more information.

- :CONDition Use to select a range that the qualify on field must occur in before a trigger event is generated. See TRIGger:CONDition for more information.
- :OCCurrence Use to select the source, slope, and number of trigger events that occur before the sweep is triggered. See TRIGger:OCCurrence for more information.

Example

```
OUTPUT 707;":TRIGGER:MODE PATTERN"
```

Query

```
:TRIGger:MODE?
```

The MODE query returns the currently selected trigger mode.

Returned Format

```
[:TRIGger:MODE] {EDGE | PATTern | STATe | DELay | TV |
GLITch}<NL>
```

Example

```
DIM Mode$[50]
OUTPUT 707;":TRIG:MODE?"
ENTER 707;Mode$
PRINT Mode$
```

NREJect

Command

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

The :TRIGger:NREJect command sets the noise reject (trigger sensitivity) for the selected source to on or off.

Example

```
OUTPUT 707;":TRIGGER:SOURCE CHANNEL1"     !select trigger
source
OUTPUT 707;":TRIGGER:NREJECT ON"
```

Query

```
:TRIGger:NREJect?
```

The NREJect query returns the current sensitivity for the selected source.

Returned Format

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

Example

```
OUTPUT 707;":TRIG:NREJ?"
ENTER 707;NREJ
PRINT NREJ
```

OCCurrence

Command

```
:TRIGger:OCCurrence <occurrence_argument>
```

The :TRIGger:OCCurrence command sets the number of trigger events that must occur before the oscilloscope is actually triggered. This command is valid in the **Delay Trigger Mode** and in the **TV Trigger Mode**.

<occurrence_argument> ::= 1 to 16000000 (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:MODE DELAY"     !select trigger mode
OUTPUT 707;":TRIGGER:OCCURRENCE 14"
```

Query

```
:TRIGger:OCCurrence?
```

The OCCurrence query returns the current value of the OCCurrence command when the oscilloscope is in the Delay Trigger Mode, or in the TV Trigger Mode with TRIGger:STANdard USER selected.

Returned Format

```
[:TRIGger:OCCurrence] <occurrence_argument><NL>
```

<occurrence_argument> ::= 1 to 16000000 (integer - NR1 format)

Example

```
OUTPUT 707;":TRIG:OCC?"
ENTER 707;Oc
PRINT Oc
```

OCCurrence:SLOPe

Command

```
:TRIGger:OCCurrence:SLOPe {POSitive | NEGative}
```

The :TRIGger:OCCurrence:SLOPe command selects the edge that will be counted by the occurrence command. The parameters for this command are NEGative or POSitive. This command is valid in the **Delay Trigger Mode** and the **TV Trigger Mode**.

Example

```
OUTPUT 707;":TRIGGER:MODE DELAY"     !select trigger mode
OUTPUT 707;":TRIGGER:OCCURRENCE:SLOPE POSITIVE"
```

Query

```
:TRIGger:OCCurrence:SLOPe?
```

The OCCurrence:SLOPe query returns the slope of the current mode.

Returned Format

```
[:TRIGger:OCCurrence:SLOPe] {POSitive | NEGative}<NL>
```

Example

```
DIM Tos$[50]
OUTPUT 707;":TRIG:OCC:SLOP?"
ENTER 707;Tos$
PRINT Tos$
```

OCCurrence:SOURce

Command

```
:TRIGger:OCCurrence:SOURce {CHANnel<n> | EXTernal}
```

The :TRIGger:OCCurrence:SOURce command selects the source that will be counted by the TRIGger:OCCurrence command. EXTernal is only valid when the oscilloscope is an HP 54520A/54522A. This command is valid only in the Delay Trigger Mode.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:MODE DELAY"     !select trigger mode
OUTPUT 707;":TRIGGER:OCCURRENCE:SOURCE CHANNEL2"
```

Query

```
:TRIGger:OCCurrence:SOURce?
```

The OCCurrence:SOURce query returns the source of the occurrence in the Delay Trigger Mode.

Returned Format

```
[:TRIGger:OCCurrence:SOURce] {CHANnel<n> | EXTernal}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Tos$[50]
OUTPUT 707;":TRIG:OCC:SOUR?"
ENTER 707;Tos$
PRINT Tos$
```

PATH

Command

```
:TRIGger:PATH {CHANnel<n> | EXTernal}
```

The :TRIGger:PATH command is valid in the **Pattern Trigger Mode**, **State Trigger Mode**, and **Delay Trigger Mode** when "qualify on" pattern or state is selected. This command selects a pattern bit as the source for future TRIGger:LOGic commands. EXTernal is only valid when the oscilloscope is an HP 54520A/54522A.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:MODE PATTERN"     !select trigger mode
OUTPUT 707;":TRIGGER:PATH CHANNEL2"
OUTPUT 707;":TRIGGER:LOGIC HIGH"       !select logic level
```

Query

```
:TRIGger:PATH?
```

The PATH query returns the current trigger source of the present mode.

Returned Format

```
[:TRIGger:PATH] {CHANnel<n> | EXTernal}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Tp$[50]
OUTPUT 707;":TRIG:PATH?"
ENTER 707;Tp$
PRINT Tp$
```

POLarity

Command

```
:TRIGger:POLarity {POSitive | NEGative}
```

The :TRIGger:POLarity command is valid only in the **TV Trigger Mode**. This command sets the polarity for the trigger when the STANdard is set to 525 or 625. The valid parameters for this command are POSitive and NEGative.

Example

```
OUTPUT 707;":TRIGGER:MODE TV"     !select trigger mode
OUTPUT 707;":TRIGGER:POLARITY NEGATIVE"
```

Query

```
:TRIGger:POLarity?
```

The POLarity query returns the current polarity setting.

Returned Format

```
[:TRIGger:POLarity] {POSitive | NEGative}<NL>
```

Example

```
DIM Tp$[50]
OUTPUT 707;":TRIG:POL?"
ENTER 707;Tp$
PRINT Tp$
```

QUALify

Command

```
:TRIGger:QUALify {{EDGE | PATTern | STATe} | {LOW
| HIGH}}
```

The :TRIGger:QUALify command is valid in the Delay and TV Trigger Mode. When in the Delay Trigger Mode, the parameters for this command are:

- EDGE
- PATTERN
- STATE

In the TV Trigger Mode, the parameters for this command are:

- LOW
- HIGH

Example

```
OUTPUT 707;":TRIGGER:MODE DELAY"    !select trigger mode
OUTPUT 707;":TRIGGER:QUALIFY PATTERN"
```

Query

```
:TRIGger:QUALify?
```

The QUALify query returns the current setting of the QUALify command in the currently selected mode.

Returned Format

```
[:TRIGger:QUALify] {{EDGE | PATTern | STATe} | {LOW |
HIGH}}<NL>
```

Example

```
DIM Tq$[50]
OUTPUT 707;":TRIG:QUAL?"
ENTER 707;Tq$
PRINT Tq$
```

Refer to the trigger mode descriptions at the beginning of this chapter for more information on the QUALify command/query.

SETup

Query

```
:TRIGger:SETup?
```

The :TRIGger:SETup query returns the current settings for the Trigger Subsystem commands. The settings returned depend on the current TRIGger:MODE as follows:

Returned Format

```
:TRIG:MODE EDGE;
  COUP {AC | DC | LFR};
  HOLD {{TIME,<holdoff_time>} | {EVEN, <event_argument>}};
  LEV <level_argument>;
  NREJ {0 | 1};
  SLOP {POS | NEG};
  SOUR {CHAN<n> | EXT | LINE | AUX}<NL>     (EXT is only valid for the
                                             54520/22)

:TRIG:MODE PATT;
  COND {ENT | EXIT | GT,<gt_argument> | LT,<lt_argument> |
    RANG,<range_gt>,<range_lt>};
  HOLD {{TIME,<holdoff_time>} | {EVEN, <event_argument>}};
  LEV <level_argument>;
  LOG {HIGH | LOW | DONT};
  NREJ {0 | 1};
  PATH {CHAN<n> | EXT}<NL>     (EXT is only valid for the 54520/22)

:TRIG:MODE STAT;
  COND {TRUE | FALS};
  HOLD {{TIME,<holdoff_time>} | {EVEN, <event_argument>}};
  LEV <level_argument>;
  LOG {HIGH | LOW | DONT};
  NREJ {0 | 1};
  PATH {CHAN<n> | EXT}     (EXT is only valid for the 54520/22)
  SLOP {POS | NEG};
  SOUR {CHAN<n> | EXT}<NL>  (EXT is only valid for the 54520/22)
```

```
:TRIG:MODE DEL;
  COND {ENT | EXIT | TRUE | FALS | GT,<gt_argument> |
    LT,<lt_argument> | RANG,<range_gt>,<range_lt>};
  DEL {{TIME,<time_value>} | {EVEN, <event_value>}};
  DEL:SLOP {POS | NEG};
  DEL:SOUR {CHAN<n> | EXT};     (EXT is only valid for the 54520/22)
  LEV <level_argument>;
  LOG {HIGH | LOW | DONT};
  NREJ {0 | 1};
  OCC <occurrence_argument>;
  OCC:SLOP {POS | NEG};
  OCC:SOUR {CHAN<n> | EXT};   (EXT is only valid for the 54520/22)
  PATH {CHAN<n> | EXT}  (EXT is only valid for the 54520/22)
  QUAL {EDGE | PATT | STAT};
  SLOP {POS | NEG};
  SOUR {CHAN<n> | EXT}<NL>  (EXT is only valid for the 54520/22)
```

**COND and PATH are only valid during delay/pattern and delay/state modes.
SLOP and SOUR are only valid during delay/edge and delay/state modes.**

```
:TRIG:MODE TV;
  COND {RANG,<range_gt>,<range_lt>};
  FIEL {1 | 2};
  HOLD {{TIME,<holdoff_time>} | {EVEN, <event_argument>}};
  LEV <level_argument>;
  LINE <line_number>;
  NREJ {0 | 1};
  OCC <occurrence_argument>;
  OCC:SLOP {POS | NEG};
  POL {POS | NEG};
  QUAL {LOW | HIGH};
  SOUR {CHAN<n> | EXT}     (EXT is only valid for the 54520/22)
  STAN {525 | 625 | USER}<NL>
```

COND and LINE are only valid when TRIGger:STANdard USER is selected.

```
:TRIG:MODE GLIT;
  GLITch:HOLD TIME,<holdoff_time>;
  GLITch:LEV <level_argument>;
  GLITch:NREJ {0 | 1};
  GLITch:SOUR {CHAN<n> | EXT}, {LOW | HIGH};     (EXT is only valid
                                                  for the 54520/22)
  GLITch:WIDth {GT | LT}, <width_argument><NL>
```

`<n>` ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

`<gt_argument>` ::= 20 ns to 160 ms (exponential - NR3 format)

`<lt_argument>` ::= 20 ns to 160 ms (exponential - NR3 format)

`<range_gt>` ::= 20 ns to 159.999 ms (must be less than range_lt) (exponential - NR3 format)

`<range_lt>` ::= 30 ns to 160 ms (must be greater than range_gt) (exponential - NR3 format)

`<time_value>` ::= amount of delay from 30ns to 160ms (exp - NR3 format)

`<event_value>` ::= number of events from 1 to 16000000 (int - NR1 format)

`<holdoff_time>` ::= 40 ns to 320 ms rounded to nearest 20 ns increment (exponential - NR3 format)

`<level_argument>` ::= trigger level in volts (exponential - NR3 format)

`<width_argument>` ::= 5 ns to 160 ms (exponential - NR3 format)

`<event_argument>` ::= 1 to 16000000 (integer - NR1 format)

`<line_number>` ::= 1 to 625 (depends on STANdard and FIELd selection) (integer - NR1 format)

`<occurrence_argument>` ::= 1 to 16000000 (integer - NR1 format)

Example

```
DIM Stp$[500]
OUTPUT 707;":TRIG:SET?"
ENTER 707;Stp$
PRINT Stp$
```

SLOPe

Command

```
:TRIGger:SLOPe {POSitive | NEGative}
```

The :TRIGger:SLOPe command specifies the slope of the edge for the trigger on the TRIGger:SOURce selected. The SLOPe command is valid in the **Edge Trigger Mode**, **State Trigger Mode**, and **Delay Trigger Mode** when EDGE or STATe is selected as the qualifier.

Example

```
OUTPUT 707;":TRIGGER:SOURCE CHAN1       !select source
OUTPUT 707;":TRIGGER:MODE DELAY"        !select trigger mode
OUTPUT 707;":TRIGGER:QUALIFY EDGE"      !qualify trigger
OUTPUT 707;":TRIGGER:SLOPE POSITIVE"
```

Query

```
:TRIGger:SLOPe?
```

The SLOPe query returns the current slope for the currently selected trigger mode for the TRIGger:SOURce selected.

Returned Format

```
[:TRIGger:SLOPe] {POSitive | NEGative}<NL>
```

Example

```
DIM Ts$[50]
OUTPUT 707;":TRIG:SLOP?"
ENTER 707;Ts$
PRINT Ts$
```

SOURce

Command

```
:TRIGger:SOURce {CHANnel<n> | EXTernal | LINE |
AUXiliary}
```

The :TRIGger:SOURce command selects the channel that actually produces the trigger. The SOURce command is valid in the **Edge Trigger Mode**, **State Trigger Mode**, **Delay Trigger Mode**, and **TV Trigger Mode**. In the Delay Trigger Mode this command is valid when EDGE or STATe is selected as the qualifier. In State Trigger Mode, this command sets the clock source. EXTernal is only valid when the oscilloscope is an HP 54520A/54522A.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":TRIGGER:SOURCE CHANNEL2"
```

Query

```
:TRIGger:SOURce?
```

The SOURce query returns the current source for the selected trigger mode.

Returned Format

```
[:TRIGger:SOURce] {CHANnel<n> | EXTernal | LINE |
AUXiliary}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[30]
OUTPUT 707;":TRIG:SOUR?"
ENTER 707;Src$
PRINT Src$
```

STANdard

Command

```
:TRIGger:STANdard {525 | 625 | USER}
```

The :TRIGger:STANdard command selects the television signal standard to be used in the **TV Trigger Mode**. The valid parameters for this command 525, 625, and USER (user defined).

Example

```
OUTPUT 707;":TRIGGER:MODE TV"     !select trigger mode
OUTPUT 707;":TRIGGER:STANDARD USER"
```

Query

```
:TRIGger:STANdard?
```

The STANdard query returns the currently selected standard.

Returned Format

```
[:TRIGger:STANdard] {525 | 625 | USER}<NL>
```

Example

```
DIM Ts$[50]
OUTPUT 707;":TRIG:STAN?"
ENTER 707;Ts$
PRINT Ts$
```

22

Waveform Memory (WMEMory) Subsystem

Introduction

The Waveform Memory (WMEMory) subsystem commands control the four nonvolatile waveform memories of the oscilloscope. Each memory is independently programmable for protect, offset, and range functions. The memory number specified in the command selects the memory that is affected by the command.

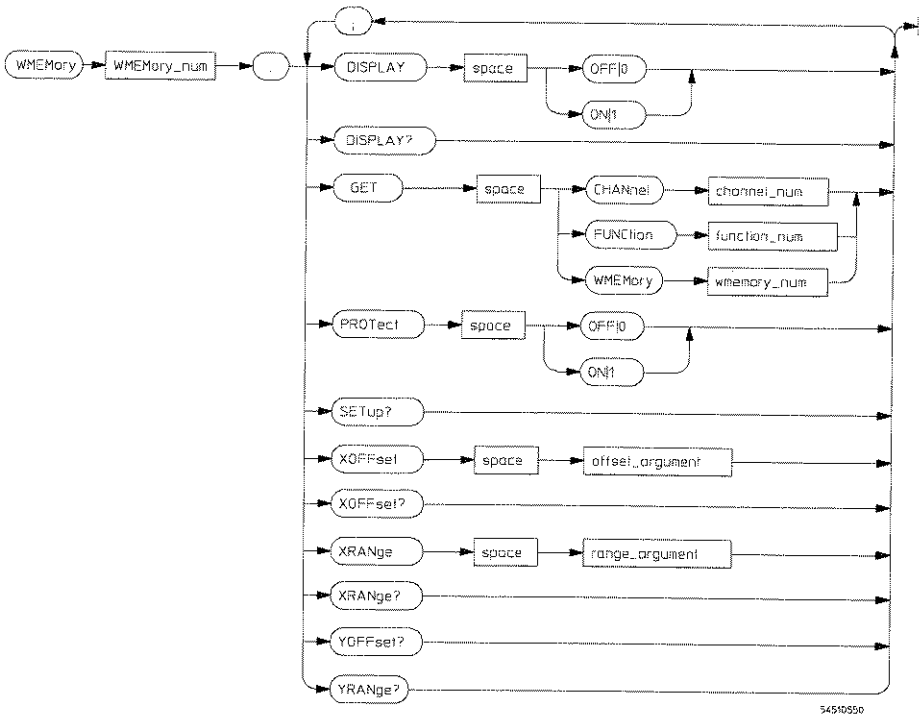
The channel displays are toggled on and off with the root level commands VIEW and BLANK, or using the WMEMory:DISPlay command.

The Waveform Memory subsystem contains the following commands:

DISPlay
GET
PROTect
SETup
XOFFset
XRANge
YOFFset
YRANge

Figure 22-1 lists the syntax diagrams for the Waveform Memory subsystem commands.

Figure22–1



channel_num = an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).

function_num = an integer, 1 through 4.

wmemory_num = an integer, 1 through 4.

offset_argument = a real number, the maximum value depends on the range selected.

range_argument = a real number, 5ns through 50 s (in a 1,2,5 sequence).

Waveform Memory Subsystem Commands Syntax Diagram

DISPlay

Command

```
:WMEMory{1 | 2 | 3 | 4}:DISPlay {{OFF | 0} |
{ON | 1}}
```

The :WMEMory{1 | 2 | 3 | 4}:DISPlay command controls the individual waveform memory displays. ON starts displaying, and OFF stops displaying the memory selected.

Example

```
OUTPUT 707;":WMEMORY2:DISPLAY ON"
```

Query

```
:WMEMory{1 | 2 | 3 | 4}:DISPlay?
```

The DISPlay query returns the current setting of the command.

Returned Format

```
[:WMEMory{1 | 2 | 3 | 4}:DISPlay] {0 | 1}<NL>
```

Example

```
OUTPUT 707;":WMEM1:DISP?"
ENTER 707;Disp
PRINT Disp
```

This command is similiar to the VIEW and BLANk root level commands.

GET

Command

```
:WMEMory{1 | 2 | 3 | 4}:GET {CHANnel<n> |
WMEMory{1 | 2 | 3 | 4} | FUNCtion{1 | 2 | 3 | 4}}
```

The :WMEMory{1 | 2 | 3 | 4} :GET command transfers selected channel, function, or waveform memory data into the waveform memory specified. When executed, an immediate erase of the waveform memory selected is performed, followed by a write of the new data. An error is generated if the destination waveform memory write protect is set to ON.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":WMEMORY1:GET CHANNEL3"
```

This command is similiar to the STORe root level command.

PROTect

Command

```
:WMEMory{1 | 2 | 3 | 4}:PROTect {{OFF | 0} |
{ON | 1}}
```

The :WMEMory{1 | 2 | 3 | 4} :PROTect command controls the write protect state of the selected nonvolatile memory. ON enables the selected waveform memories write protect state. OFF disables the selected waveform memories write protect state, and allows data to be written in memory. When ON, any attempted save will generate an error, and the contents remain unchanged.

Example

```
OUTPUT 707;":WMEMORY2:PROTECT ON"
```

Query

```
:WMEMory{1 | 2 | 3 | 4}:PROTect?
```

The PROTECT query returns the current setting of the command.

Returned Format

```
[:WMEMory{1 | 2 | 3 | 4}:PROTect] {0 | 1}<NL>
```

Example

```
OUTPUT 707;":WMEM:PROT?"
ENTER 707;Disp
PRINT Disp
```

SETup?

Query

```
:WMEMory{1 | 2 | 3 | 4}:SETup?
```

The :WMEMory{1 | 2 | 3 | 4}:SETup query returns the current settings for the WMEMory Subsystem commands.

Returned Format

```
:WMEM{1 | 2 | 3 | 4}:DISP {0 | 1};
  PROT {0 | 1};
  XOFF <offset_argument>;
  XRANG <range_argument>;
  YOFF <yoffset>;
  YRANG <yrange><NL>
```

`<offset_argument>` ::= time from trigger to display reference in seconds. Display reference is left, center, or right (exponential - NR3 format)

`<range_argument>` ::= 5 ns to 50 sec (exponential - NR3 format)

`<yoffset>` ::= offset value in volts (exponential - NR3 format)

`<yrange>` ::= full-scale range value (exponential - NR3 format)

Example

```
DIM Stp$[300]
OUTPUT 707;":WMEMORY1:SETUP?"
ENTER 707;Stp$
PRINT Stp$
```

XOFFset

Command

```
:WMEMory{1 | 2 | 3 | 4}:XOFFset <offset_argument>
```

The :WMEMory{1 | 2 | 3 | 4}:XOFFset command sets the time base delay for the waveform memory selected, from 512 to 32768 point waveforms in real-time mode. This delay is the internal time between the trigger event and the on-screen delay reference point. The delay reference point is the left edge of the display, the right edge of the display, or the center of the display, and is set with the :TIMebase:REFerence command when the waveform was stored.

`<xoffset_argument>` ::= time in seconds from trigger to the on screen delay reference point. The maximum value depends on the :WMEMory:XRANge setting (double exponential - NR3 format).

Example

```
OUTPUT 707;":WMEMORY:XOFFSET 2MS"
```

Query

```
:WMEMory{1 | 2 | 3 | 4}:XOFFset?
```

The XOFFset query returns the current offset value.

Returned Format

```
[:WMEMory{1 | 2 | 3 | 4}:XOFFset] <offset_argument><NL>
```

`<xoffset_argument>` ::= time from trigger to display reference in seconds. Display reference is left, center, or right (double exponential - NR3 format)

Example

```
OUTPUT 707;":WMEM1:XOFF?"
ENTER 707;Xofst
PRINT Xofst
```

XRANge

Command

```
:WMEMory{1 | 2 | 3 | 4}:XRANge <range_argument>
```

The :WMEMory{1 | 2 | 3 | 4}:XRANge command sets the full-scale horizontal time in seconds for the waveform memory selected, from 512 to 32768 point waveforms in real-time mode. The XRANge value is ten times the time per division.

`<range_argument>` ::= 5 ns to 50 s in a 1,2,5 sequence (exponential - NR3 format)

Example

```
OUTPUT 707;":WMEMORY:XRANGE 100MS"
```

Query

```
:WMEMory{1 | 2 | 3 | 4}:XRANge?
```

The XRANge query returns the current full-scale range value.

Returned Format

```
[:WMEMory{1 | 2 | 3 | 4}:XRANge] <range_argument><NL>
```

`<range_argument>` ::= 5 ns to 50 s (exponential - NR3 format)

Example

```
OUTPUT 707;":WMEM:XRAN?"
ENTER 707;Xrng
PRINT Xrng
```

YOFFset?

Query

```
:WMEMory{1 | 2 | 3 | 4}:YOFFset?
```

The :WMEMory{1 | 2 | 3 | 4}:YOFFset query returns the voltage that is represented at center screen for the selected waveform memory.

Returned Format

```
[:WMEMory{1 | 2 | 3 | 4}:YOFFset] <yoffset><NL>
```

<yoffset> ::= offset value in volts (exponential - NR3 format)

Example

```
OUTPUT 707;":WMEMORY2:YOFFSET?"
ENTER 707;Yofst
PRINT Yofst
```

This value is set in memory when the waveform is stored, and cannot be changed.

YRANge?

Query

```
:WMEMory{1 | 2 | 3 | 4}:YRANge?
```

The :WMEMory{1 | 2 | 3 | 4}:YRANge query returns the full-scale vertical axis of the selected waveform memory.

Returned Format

```
[:WMEMory{1 | 2 | 3 | 4}:YRANge] <yrange><NL>
```

<yrange> ::= full-scale range value (exponential - NR3 format)

Example

```
OUTPUT 707;":WMEMORY2:YRANGE?"
ENTER 707;Yrng
PRINT Yrng
```

This value is set in memory when the waveform is stored, and cannot be changed.

23

WAVeform Subsystem

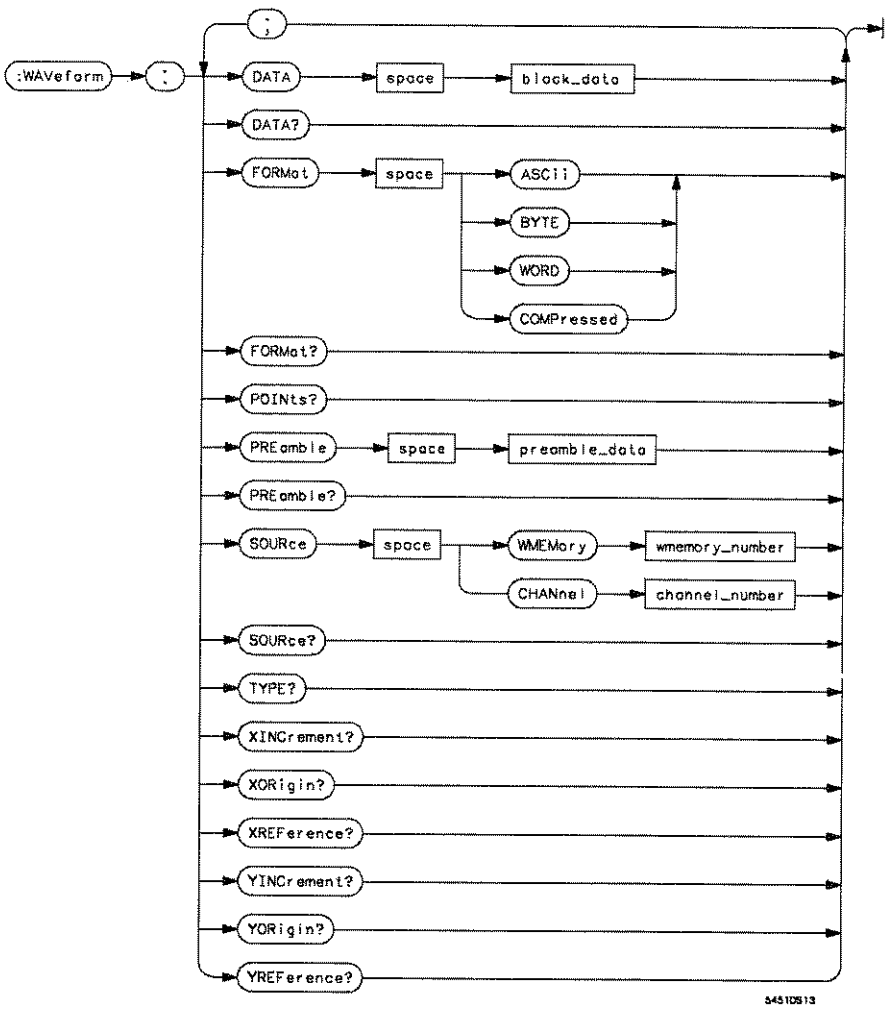
Introduction

The WAVeform subsystem is used to transfer waveform data between a controller and the oscilloscope's waveform memories. This subsystem contains the following commands:

DATA
FORMat
POINts
PREamble
SOURce
TYPE
XINCrement
XORigin
XREFerence
YINCrement
YORigin
YREFerence

Figure 23-1 lists the syntax diagrams for the Waveform subsystem commands.

Figure 23–1



channel_number = an integer, 1 and 2 (HP 54520A/54522A), or 1 through 4 (HP 54540A/54542A).

block_data = definite block data in IEEE 488.2 # format.

preamble_data = refer to the PREAMBLE command.

wmemory_number = an integer, 1 through 4.

Waveform Subsystem Commands Syntax Diagram

Waveform Data and Preamble

The waveform record is actually contained in two portions: the waveform data and the preamble. The waveform data is the actual data acquired for each point in the specified source. The preamble contains the information for interpreting the waveform data, which includes the number of points acquired, format of acquired data, and type of acquired data. The preamble also contains the X and Y increments, origins, and references for the acquired data, so that the raw data can be translated to time and voltage values.

The values set in the preamble are determined when the :DIGitize command is executed or when the front-panel STORE key is pressed.

- When DIGitize is used, the preamble values are based on the settings of variables in the ACQuire subsystem.
- When STORe is used, they are based on the front-panel setup when the STORE key is pressed.

Although the preamble values can be changed with a controller, the way the data is acquired cannot be changed. Changing the preamble values cannot change the type of data that was actually acquired, the number of points actually acquired, etc. Therefore, you must use extreme caution when changing any waveform preamble values to ensure the data is still useful. For example, setting POINts in the preamble to a value different from the actual number of points in the waveform results in inaccurate data.

The waveform data and preamble must be read by the controller or sent to the oscilloscope with two separate commands, DATA and PREamble.

Sending consecutive :DIGitize commands may improve the data throughput. Refer to the :DIGitize command in the "Root Level" chapter for more information.

Data Acquisition Types

There are five types of waveform acquisition that can be selected with the :ACQuire:TYPE command: NORMal, AVERage, ENVelope, PDETect and RAWData. When the data is acquired using the DIGitize command, the data is placed in the channel buffer of the specified source. However, during a RAWData acquisition, data is placed in a special rawdata buffer.

After a DIGitize command, the instrument is stopped. If the instrument is restarted, over the remote interface or the front panel, the data acquired with the DIGitize command is overwritten.

With the exception of the RAWData mode, waveforms in the oscilloscope are:

- 500 points long for repetitive mode.
- 512, 1024, 2048, 4096, 8192, 16384, or 32768 points long for real-time mode.

Real-time Mode

A real-time waveform consists of user specified sample points. If the following conditions are met, all points are transferred over the bus:

- :ACQUIRE:POINTS is set to 512, 1024, 2048, 4096, 8192, 16384, or 32768.
- :WAVEFORM:DATA is requested.
- The specific channel or waveform memory contains real-time data.

During normal operation, the screen only shows 500 points of the total points acquired.

Repetitive Modes

In the repetitive mode, the on-screen time is divided into 500 time buckets and all waveforms contain 500 points.

Normal

Normal data consists of the last data point (hit) in each time bucket. This data is transmitted over the remote interface in a linear fashion starting with time bucket 0 and going through time bucket *n–1*, where *n* is the number returned by the WAVeform:POINts query. Only the magnitude values of each data point are transmitted. The first voltage value corresponds to the first time bucket on the left of the screen and the last value corresponds to the next to last time bucket on the right side of the screen. Time buckets that do not have data in them return –1.

The time values for each data point correspond to the position of the data point in the data array. These values are not transmitted.

Average

Average data consists of the average of the first *n* hits in a time bucket, where *n* is the value returned by the ACQuire:COUNt query. Time buckets that have fewer than *n* hits return the average of what data they do have. If the :ACQuire:COMPlete parameter is set to 100%, then each time bucket must contain the number of data hits specified with the :ACQuire:COUNt command. If a time bucket does not have any data in it, it returns –1. This data is transmitted over the remote interface in a linear fashion, starting with time bucket 0 and proceeding through time bucket *n–1*, where *n* is the number returned by the WAVeform:POINts query. The first value corresponds to a point at the left side of the screen and the last value corresponds to one point away from the right side of the screen.

Envelope

Envelope data consists of two arrays of data: one containing the minimum hits in each time bucket, and the other containing the maximum hits in each time bucket. If a time bucket does not have any hits in it, then –1 is returned for both the minimum and maximum values. The two arrays are transmitted one at a time over the remote interface linearly, starting with time bucket 0 (on the left side of the screen) and proceeding through time bucket *m–1*, where *m* is the value returned by the WAVeform:POINts query. The array with the minimum values is sent first. The first value of each array corresponds to the data point on the left of the screen. The last value is one data point away from the right side of the screen.

The data is transferred from the channel buffer to a controller using the WAVeform:DATA query. Data is transferred into the instrument from a controller using the WAVeform:DATA command. Envelope data can be transferred into Waveform Memories 1 and 2 , if WMEMory1 is specified as the source. If WMEMory3 is specified as the source, the envelope data can be transferred into Waveform Memories 3 and 4.

The data is transferred as two arrays. If Waveform Memory 1 is specified as the source, the first array is transferred into Waveform Memory 1 and the second array is transferred into Waveform Memory 2. If Waveform Memory 3 is specified as the source, the first array is transferred into Waveform Memory 3 and the second array is transferred into Waveform Memory 4. The data type is then changed to normal for each of the waveform memories.

Rawdata

Rawdata is acquired with special routines that are optimized for quick fiso unloading. These routines do not filter the data.

While acquiring data in the Rawdata mode, data is stored as uncalibrated 8-bit Gray code data in a large buffer. When all acquisitions are complete, the buffer is translated into unfiltered, calibrated 16-bit binary data and sent over the bus in the WORD format. The :WAVeform:FORMat command has no effect in the Rawdata mode.

The RAWData command has three parameters: length, acquisitions, and mode. Length specifies the number of points of each acquisition. Acquisitions specify the number of acquisitions to be taken in a single digitize operation. Mode specifies how the data is processed. The data is transferred from the oscilloscope in a single IEEE 488.2 data block. The number of bytes in the block can be determined with the following equation:

$$\text{Number of Bytes} = \text{Acquisitions} \times (\text{Length} \times 2 + 8)$$

The data block consists of two arrays. The first array consists of double precision 64-bit floating point numbers. This array contains the xorigin values of the waveform records to follow. It can read directly into a double precision real array in a controller allocated by the BASIC command ALLOCATE REAL Xorigins(1:Acquisitions). The xorigin values are also available in ASCII format using the :WAVeform:XORigin query. These xorigin values are accurate to within the timing resolution of the oscilloscope. One of the xorigin values is supplied in the preamble for the rawdata record.

The second array consists of 16-bit integer numbers. This data is transmitted in a linear fashion over the remote interface. It starts with sample zero of the first acquisition and continues through sample length-1 of the this acquisition. Then it continues in a similar fashion with sample zero through sample length-1 of the each following acquisition through the last acquisition. This array can be read directly into a two dimensional integer array allocated by the BASIC command ALLOCATE INTEGER Waveforms(1:Acquisitions, 1:Points).

Data Conversion

Data sent from the oscilloscope must be scaled for useful interpretation. The values used to interpret the data are the X and Y references, X and Y origins, and X and Y increments. These values are read from the waveform preamble.

Conversion from Data Value to Voltage

The following formula converts a data value from waveform memories 1 through 4, or channels 1 or 2, to a voltage value:

voltage = [(data value - yreference)*yincrement] + yorigin.

Conversion from Data Value to Time

The time value of a data point can be determined by the position of the data point. As an example, the fourth data point sent with XORigin = 16 ns, XREFerence = 0, and XINCrement = 2 ns can be calculated using the following formula:

time = [(data point number-xreference)*xincrement]+xorigin,

This would result in the following calculation:

time = [(3 - 0) * 2 ns] + 16 ns = 22 ns.

Data Format for Remote Interface Transfer

There are four formats for transferring waveform data over the remote interface: WORD, BYTE, COMPressed, and ASCii.

WORD, BYTE, and COMPressed formatted waveform records are transmitted using the arbitrary block program data format specified in IEEE 488.2. ASCii format block data does not use a block header.

When you use the block data format, the ASCii character string "#8<DD...D>" is sent prior to sending the actual data. The 8 indicates how many D's follow. The D's are ASCII numbers which indicate how many data bytes follow.

For example, if 500 points were acquired and the BYTE format was specified, the block header "#800000500" would be sent. The 8 indicates that eight length bytes follow, and 500 indicates that 500 binary data bytes follow.

WORD Format

In the WORD format, the number of data bytes is twice the number of words (data points). The number of data points is the value returned by the :WAVeform:POINts? query. The number of data bytes is followed by a sequence of bytes representing the data points, with the most significant byte of each word transmitted first. In this format the data is shifted so that the most significant bit after the sign bit contains the most significant bit of the data. If there is a hole in the data, the hole is represented by the 16-bit value of –1. The range of data in the WORD format is from 0 to 32640.

WORD format is useful in applications where the information is read directly into an integer array in a controller.

WORD and ASCii formatted data returns the most accurate data values.

BYTE Format

The BYTE format allows only seven bits to be used to represent the voltage values, with the first bit being the sign bit. If there is a hole in the data, the hole is represented by a value of –1.

The BYTE-formatted data transfers over the remote interface faster than WORD-formatted data, since one byte per point is transferred in BYTE format and two bytes per point are transferred in WORD format. The BYTE-formatted data has less resolution than WORD-formatted data.

COMPressed Format

The number of bytes transmitted when the format is COMPressed is the same as the value returned by the WAVeform:POINts? query.

Eight bits of resolution are retained in the COMPressed format. So that a hole in the data may be represented, a data value of 255 is mapped to 254, and 255 is used to represent a hole. This mode gives greater vertical precision than BYTE-formatted data, with faster transfer times than WORD-formatted data. On the other hand, the COMPressed mode probably requires more time after the transfer for data to be unpacked.

ASCii Format

Waveform records in the ASCii format are transmitted one value at a time, separated by a comma. The data values transmitted are the same as the values sent in the WORD FORMat except that they are converted to an integer ASCii format (six or less characters) before being sent over the remote interface.

ASCii values cannot be used to transfer data into a controller.

DATA

Command

```
:WAVeform:DATA <block_data>
```

The :WAVeform:DATA command transfers a waveform data record to the instrument over the remote interface and stores it in the previously specified waveform memory. The waveform memory is specified with the :WAVeform:SOURce command. Only waveform memories can have waveform data sent to them.

> The oscilloscope only accepts 500 point records in repetitive, or 512, 1024, 2048, 4096, 8192, 16384, or 32768 point records in real-time mode. If you try to send variable length records back into the oscilloscope, an error message appears on the screen.

`<block_data>` ::= definite block data in IEEE 488.2 # format

The format of the data being sent must match the format previously specified by the waveforms preamble for the destination memory. The :WAVeform:DATA command does not accept ASCii data. However, the ASCii format can be specified for data output.

Query

```
:WAVeform:DATA?
```

The DATA query outputs a waveform record to the controller over the remote interface that is stored in a waveform memory or channel buffer previously specified with the :WAVeform:SOURce command.

Returned Format

```
[:WAVeform:DATA] <block_data><NL>
```

> The oscilloscope must be stopped, or the DIGitize command must have been previously executed before reading waveform data using the WAVeform:DATA query.

`<block_data>` ::= definite block data in IEEE 488.2 # format

Example

The following program moves data from the oscilloscope to the controller and then back to the oscilloscope with the :WAVeform:DATA query and command.

```
5 *RST ! Reset the oscilloscope
10 CLEAR 707 ! Initialize instrument interface
20 OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE"    ! Select sample mode
30 ! Setup the Acquire subsystem
40 OUTPUT 707;":ACQUIRE:TYPE NORMAL"     ! Display mode to NORMAL
60 OUTPUT 707;":ACQUIRE:POINTS 500"
70 OUTPUT 707;":DIGITIZE CHANNEL1"       ! Acquire data
80 OUTPUT 707;":SYSTEM:HEADER OFF"       ! Headers off
90 OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"  ! Channel 1 as waveform source
100 OUTPUT 707;":WAVEFORM:FORMAT WORD"   ! WORD format
110 OUTPUT 707;":WAVEFORM:DATA?"
120 ENTER 707 USING "#,2A,8D";Headers$,Bytes   ! Read length byte
130 Length=Bytes
140 Length=Length/2
150 ALLOCATE INTEGER Waveform(1:Length)
160 ENTER 707 USING "#,W";Waveform(*)    ! Enter waveform data to integer array
170 ENTER 707 USING "-K,B";End$
180 DIM Preamble$[200]     ! Dimension variable
190 OUTPUT 707;":WAVEFORM:PREAMBLE?"     ! Output waveform preamble to controller
200 ENTER 707 USING "-K";Preamble$
210 OUTPUT 707;":WAVEFORM:SOURCE WMEMORY4"     ! Waveform memory 4 as source
220 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Preamble$  ! Send preamble
221           ! from controller to WMEMORY 4
230 OUTPUT 707 USING "#,K";":WAVEFORM:DATA #800001000"      ! Send header
240 OUTPUT 707 USING "W";Waveform(*)     ! Send waveform data to WMEMORY 4
250 OUTPUT 707;":BLANK CHANNEL1"  ! Turn channel 1 off
260 OUTPUT 707;":VIEW WMEMORY4"   ! Turn WMEMORY 4 on
270 END
```

In program line 220, the space after :WAVeform:PREamble and before the quotation mark is required.

FORMat

Command

```
:WAVeform:FORMat {ASCii | WORD | BYTE |
COMPressed}
```

The :WAVeform:FORMat command sets the data transmission mode for waveform data output. This command controls how the data is formatted on the Remote Interface when sent from the oscilloscope.

When the ASCii mode is selected, the data consists of ASCii digits with each data value separated by a comma. The ASCii mode cannot be used to transfer data into a controller.

WORD formatted data transfers as 16-bit binary integers in two bytes, with the most significant byte of each word sent first.

BYTE and COMPressed formatted data is transferred as 8-bit bytes. COMPressed formatted data (recommended) has full 8-bit resolution, while BYTE is truncated to 7 bits.

> The :WAVeform:FORMat command has no effect in the Rawdata mode. In this mode, data is always transferred in the WORD format.

Example

```
OUTPUT 707;":WAVEFORM:FORMAT WORD"
```

Query

```
:WAVeform:FORMat?
```

The FORMat query returns the current output format for the transfer of waveform data.

Returned Format

```
[:WAVeform:FORMat] {ASCii | WORD | BYTE | COMPressed}<NL>
```

Example

```
DIM Frmt$[30]
OUTPUT 707;":WAV:FORM?"
ENTER 707;Frmt$
PRINT Frmt$
```

POINts?

Query

```
:WAVeform:POINts?
```

The :WAVeform:POINts query returns the points value in the currently selected waveform preamble. The points value is the number of time buckets contained in the waveform selected with the WAVeform:SOURce command.

Returned Format

```
[:WAVeform:POINts] {500 | 512 | 1024 | 2048 | 4096 | 8192 |
16384 | 32768}<NL>
```

Example

```
OUTPUT 707;":WAVEFORM:POINTS?"
ENTER 707;Pts
PRINT Pts
```

PREamble

Command

```
:WAVeform:PREamble <preamble_data>
```

The :WAVeform:PREamble command sends a waveform preamble to a previously selected waveform memory in the instrument.

In the Rawdata mode, the format is always 2 for WORD, and the number of acquisitions is returned by the count parameter. The count parameter returns 1 in all other modes.

`<preamble_data>` ::= <format NR1>,<type NR1>,<points NR1>,<count NR1>,<xincrement NR3>,<xorigin double NR3>,<xreference NR3>,<yincrement NR3>,<yorigin NR3>,<yreference NR3>

Query

```
:WAVeform:PREamble?
```

The PREamble query sends a waveform preamble to the controller from the waveform source.

Returned Format

```
[:WAVeform:PREamble] <preamble block><NL>
```

`<preamble_data>` ::= <format>,<type>,<points>,<count>,<xincrement >, <xorigin>,<xreference>,<yincrement>,<yorigin>,<yreference>

`<format>` ::= 0 for ASCii format
1 for BYTE format
2 for WORD format
4 for COMPressed format (integer - NR1 format)

`<type>` ::= 0 for INVALID type
1 for REPetitive NORMal type or REALtime
2 for REPetitive AVERage type
3 for REPetitive ENVelope type
4 for RAWData type
5 for PDETect type (integer - NR1 format)

See individual commands for <points>, <count>, <xincrement >, <xorigin>, <xreference>, <yincrement>, <yorigin>, and <yreference> arguments.

The oscilloscope must be stopped, or the DIGitize command must have been previously executed before reading waveform preamble using the WAVeform:PREamble query.

This example program uses both the command and query form of the PREamble command. First the preamble is queried (output to the controller). Then, the preamble is returned to a previously selected waveform memory.

Example

```
10 OUTPUT 707;":WAVEFORM:SOURCE WMEM1"
20 DIM Pre$[120]
30 OUTPUT 707;":SYSTEM:HEADER OFF"
40 OUTPUT 707;":WAVEFORM:PREAMBLE?"
50 ENTER 707 USING "-K";Pre$
60 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Pre$
70 END
```

In line 60 of the program example, a space is inserted between the word "PREamble " and the closed quotation mark. This space must be inside the quotation mark because in this format (#,K) the data is packed together. Failure to add the space produces a word that is not a proper command word.

The following program example places the preamble in a numeric array.

Example

```
10 OUTPUT 707;":WAVEFORM:SOURCE WMEM1"
20 DIM Preamble(1:10)
30 OUTPUT 707;":SYSTEM:HEADER OFF"
40 OUTPUT 707;":WAVEFORM:PREAMBLE?"
50 ENTER 707 ;Preamble(*)
60 OUTPUT 707;":WAVEFORM:PREAMBLE ";Preamble(*)
70 END
```

SOURce

Command

```
:WAVeform:SOURce {CHANnel<n> |
WMEMory{1 | 2 | 3 | 4}}
```

The :WAVeform:SOURce command selects the channel or waveform memory to be used as the source for the waveform commands. This command also selects the destination for data being transferred over the bus.

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
OUTPUT 707;":WAVEFORM:SOURCE WMEMORY3"
```

Query

```
:WAVeform:SOURce?
```

The SOURce query returns the currently selected source for the waveform commands.

Returned Format

```
[:WAVeform:SOURce] {CHANnel<n> | WMEMory{1 | 2 | 3 | 4}}<NL>
```

<n> ::= 1 or 2 (HP 54520A/54522A)
::= 1 through 4 (HP 54540A/54542A) (integer - NR1 format)

Example

```
DIM Src$[30]
OUTPUT 707;":WAV:SOUR?"
ENTER 707;Src$
PRINT Src$
```

TYPE?

Query

```
:WAVeform:TYPE?
```

The :WAVeform:TYPE query returns the data type for the previously specified waveform source.

Returned Format

```
[:WAVeform:TYPE] {INValid | AVERage | ENVelope | NORMal |
PDETect | RAWData}<NL>
```

Example

```
DIM Typ$[30]
OUTPUT 707;":WAVEFORM:TYPE?"
ENTER 707;Typ$
PRINT Typ$
```

XINCrement?

Query

```
:WAVeform:XINCrement?
```

The :WAVeform:XINCrement query returns the current x-increment value in the preamble for the current specified source. This value is the time difference between consecutive data points for NORMal, AVERage, ENVelope, PDETect, or RAWData.

For a 500 point waveform, the x-increment pixel width is in seconds. For 512, 1024, 2048, 4096, 8192, 16384, or 32768 point waveforms and RAWData waveforms, the xincrement value is the sample period. Therefore, these waveforms never contain the interpolated data points.

Returned Format

```
[:WAVeform:XINCrement] <value><NL>
```

<value> ::= x-increment in the current preamble (exponential - NR3 format)

Example

```
OUTPUT 707;":WAVEFORM:XINCREMENT?"
ENTER 707;Xin
PRINT Xin
```

XORigin?

Query

```
:WAVeform:XORigin?
```

The :WAVeform:XORigin query returns the current x-origin value in the preamble for the current specified source. For a 500 point record, the x-origin value is the time of the first data point in the memory with respect to the trigger point. For a 512, 1024, 2048, 4096, 8192, 16384, or 32768 point record, the x-origin value is the point referenced by x-reference.

In the RAWData mode, the :WAVeform:XORigin query may return multiple values depending on the number of acquisitions specified. In this mode, the x-origin value is the time of the first data point in each acquisition.

> In the rawdata mode, the x-origin value returned by the preamble is the minimum time of the first point in memory that occurs in any acquired data record.

Returned Format

```
[:WAVeform:XORigin] <value>[,<value>]...<NL>
```

<value> ::= x-origin value in the current preamble (exponential - NR3 format)

Example

```
OUTPUT 707;":WAVEFORM:XORIGIN?"
ENTER 707;Xr
PRINT Xr
```

XREFerence?

Query

```
:WAVeform:XREFerence?
```

The :WAVeform:XREFerence query returns the current x-reference value in the preamble for the current specified source. This value specifies the number of the first point on the screen. For 500 point waveforms and the rawdata mode, xreference is zero. For 512, 1024, 2048, 4096, 8192, 16384, or 32768 point waveforms, xreference is the number of the first data point appearing on screen. This x-reference value is between zero and 1 less than the value selected. The exact number depends on the sweep speed; left, center, or right reference; and the pan and zoom if the waveform is stopped.

Returned Format

```
[:WAVeform:XREFerence] <value><NL>
```

<value> ::= x-reference value in the current preamble (integer - NR1 format)

Example

```
OUTPUT 707;":WAVEFORM:XREFERENCE?"
ENTER 707;Xrf
PRINT Xrf
```

YINCrement?

Query

```
:WAVeform:YINCrement?
```

The :WAVeform:YINCrement query returns the current y-increment value in the preamble for the current specified source. This value is the voltage difference between consecutive data points.

Returned Format

```
[:WAVeform:YINCrement] <value><NL>
```

<value> ::= y-increment value in the current preamble (exponential - NR3 format)

Example

```
OUTPUT 707;":WAVEFORM:YINCREMENT?"
ENTER 707;Yin
PRINT Yin
```

YORigin?

Query

```
:WAVeform:YORigin?
```

The :WAVeform:YORigin query returns the current y-origin value in the preamble for the current specified source. This value is the voltage at the center of the screen.

Returned Format

```
[:WAVeform:YORigin] <value><NL>
```

<value> ::= y-origin in the current preamble (exponential - NR3 format)

Example

```
OUTPUT 707;":WAVEFORM:YORIGIN?"
ENTER 707;Yr
PRINT Yr
```

YREFerence?

Query

```
:WAVeform:YREFerence?
```

The :WAVeform:YREFerence query returns the current y-reference value in the preamble for the current specified source. This value specifies the data point where the y-origin occurs.

Returned Format

```
[:WAVeform:YREFerence] <value><NL>
```

<value> ::= y-reference value in the current preamble (integer - NR1 format)

Example

```
OUTPUT 707;":WAVEFORM:YREFERENCE?"
ENTER 707;Yrf
PRINT Yrf
```

24

EXTernal Trigger Subsystem

Introduction

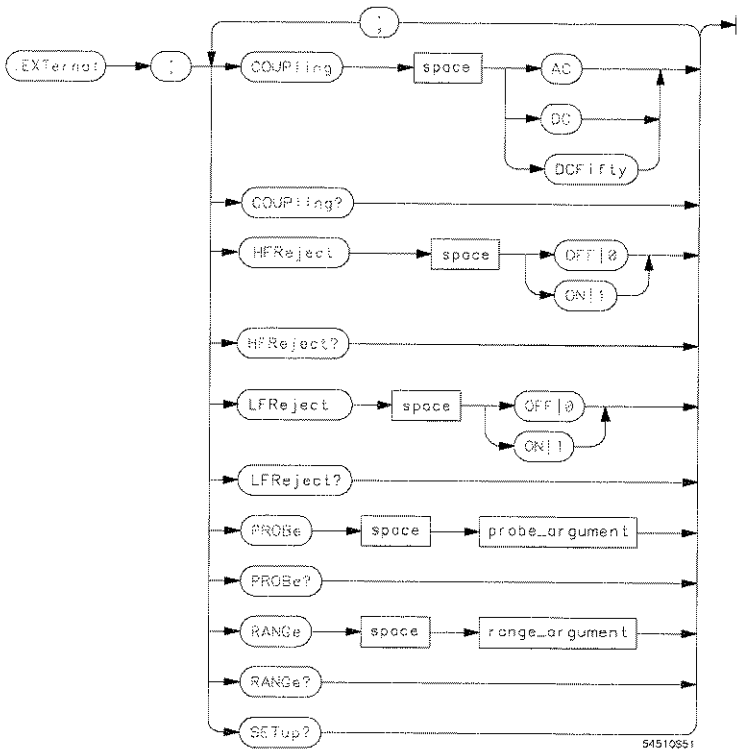
The External Trigger subsystem contains the following commands:

COUPling
HFReject
LFReject
PROBe
RANGe
SETup

The commands in this subsystem are for the external trigger on the HP 54520 and HP 54522 oscilloscopes. If any of these commands are sent to an HP 54540 or HP 54542, the error, "Undefined header", is issued.

Figure 24-1 lists the syntax diagrams for the External Trigger subsystem commands.

Figure 24–1



probe_argument = a real number from 0.9 to 1000.0 specifying the probe attenuation with respect to 1.

range_argument = a real number specifying the size of the acquisition window in volts either 1.0, 5.0, or 25.0.

External Trigger Subsystem Commands Syntax Diagram

COUPling

Command

```
:EXTernal:COUPling {AC | DC | DCFifty}
```

The :EXTernal:COUPling command selects the coupling for the external trigger. The coupling can be set to AC, DC, or DCFifty. DCFifty places an internal 50 Ω load on the external trigger. AC and DC automatically set the external trigger impedance to 1M ohm.

Example

```
OUTPUT 707;":EXTERNAL:COUPLING DC"
```

Query

```
:EXTernal:COUPling?
```

The COUPling query returns the current coupling for the external trigger.

Returned Format

```
[:EXTernal:COUPling] {AC | DC | DCFifty}
```

Example

```
DIM Ch$[50]
OUTPUT 707;":EXTernal:COUPling?"
ENTER 707;Ch$
PRINT Ch$
```

HFReject

Command

```
:EXTernal:HFReject {{OFF | 0} | {ON | 1}}
```

The :EXTernal:HFReject command controls an internal low-pass filter. When the filter is ON, the bandwidth of the external trigger is limited to approximately 20 MHz. The bandwidth limit filter may be used when either AC, DC, or DCFifty coupling is used.

Example

```
OUTPUT 707;":EXTERNAL:HFREJECT ON"
```

Query

```
:EXTernal:HFReject?
```

The HFReject query returns the current setting of the command.

Returned Format

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

Example

```
OUTPUT 707;":EXTernal:HFReject?"
ENTER 707;Hf
PRINT Hf
```

LFReject

Command

```
:EXTernal:LFReject {{OFF | 0} | {ON | 1}}
```

The :EXTernal:LFReject command controls an internal high-pass filter. When the filter is ON, the bandwidth of the external trigger is limited to approximately 400 Hz. The bandwidth limit filter may be used only when AC coupling is used. Otherwise, the bandwidth limit filter is automatically turned off.

Example

```
OUTPUT 707;":EXTERNAL:COUPLING AC"     !select AC coupling
OUTPUT 707;":EXTERNAL:LFREJECT ON"
```

Query

```
:EXTernal:LFReject?
```

The LFReject query returns the current setting of the command.

Returned Format

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

Example

```
OUTPUT 707;":EXT:LFR?"
ENTER 707;Lf
PRINT Lf
```

PROBe

Command

```
:EXTernal:PROBe <probe_argument>
```

The :EXTernal:PROBe command specifies the probe attenuation factor for the external trigger. The range of the probe attenuation factor is from 0.9 to 1000.0. This command does not change the actual input sensitivity of the oscilloscope. It changes the reference constants for scaling the display factors, for making automatic measurements, for setting trigger levels, etc. Because of the coupling that occurs between the vertical and trigger level controls, EXTernal:PROBe must be specified prior to changing EXTernal:RANGe, or TRIGger:LEVel.

`<probe_argument>` ::= 0.9 to 1000.0 (exponential - NR3 format)

Example

```
OUTPUT 707;":EXTERNAL:PROBE 24"
```

Query

```
:EXTernal:PROBe?
```

The PROBe query returns the current probe attenuation factor for the external trigger.

Returned Format

```
[:EXTernal:PROBe] <probe_argument><NL>
```

`<probe_argument>` ::= full-scale range value (exponential - NR3 format)

Example

```
OUTPUT 707;":EXTernal:PROBe?"
ENTER 707;Prb
PRINT Prb
```

RANGe

Command

```
:EXTernal:RANGe <range_argument>
```

The :EXTernal:RANGe command defines the full-scale vertical axis of the external trigger. The RANGe can be set to 1.0, 5.0, or 25.0 when using 1:1 probe attenuation. If the probe attenuation is changed, the range value is multiplied by the probe attenuation factor.

`<range_argument>` ::= a real number specifying the size of the acquisition window in volts either 1.0, 5.0, or 25.0 (exponential - NR3 format)

Examples

```
:OUTPUT 707;":EXTERNAL:RANGE 25.0"
:OUTPUT 707;":EXTERNAL:RANGE 1.0 V"
```

Query

```
:EXTernal:RANGe?
```

The RANGe query returns the current full-scale range setting for the external trigger.

Returned Format

```
[:EXTernal:RANGe] <range_argument><NL>
```

`<range_argument>` ::=a real number specifying the size of the acquisition window in volts either full-scale range value (exponential - NR3 format)

Example

```
OUTPUT 707;":EXT:RANG?"
ENTER 707;Rng
PRINT Rng
```

SETup?

Query

```
:EXTernal:SETup?
```

The :EXTernal:SETup query returns the current settings for the EXTernal Subsystem commands.

Returned Format

```
[:EXTernal:SETup] COUP{AC|DC|DCF};
  HFR {0 | 1};
  LFR {0 | 1};
  PROB <probe_argument>;
  RANG <range_argument><NL>
```

`<probe_argument>` ::= 0.9 to 1000.0 (exponential - NR3 format)

`<range_argument>` ::=a real number specifying the size of the acquisition window in volts either 1.0, 5.0, or 25.0. (exponential - NR3 format)

Example

```
DIM Stp$[300]
OUTPUT 707;":EXTERNAL:SETUP?"
ENTER 707;Stp$
PRINT Stp$
```

25

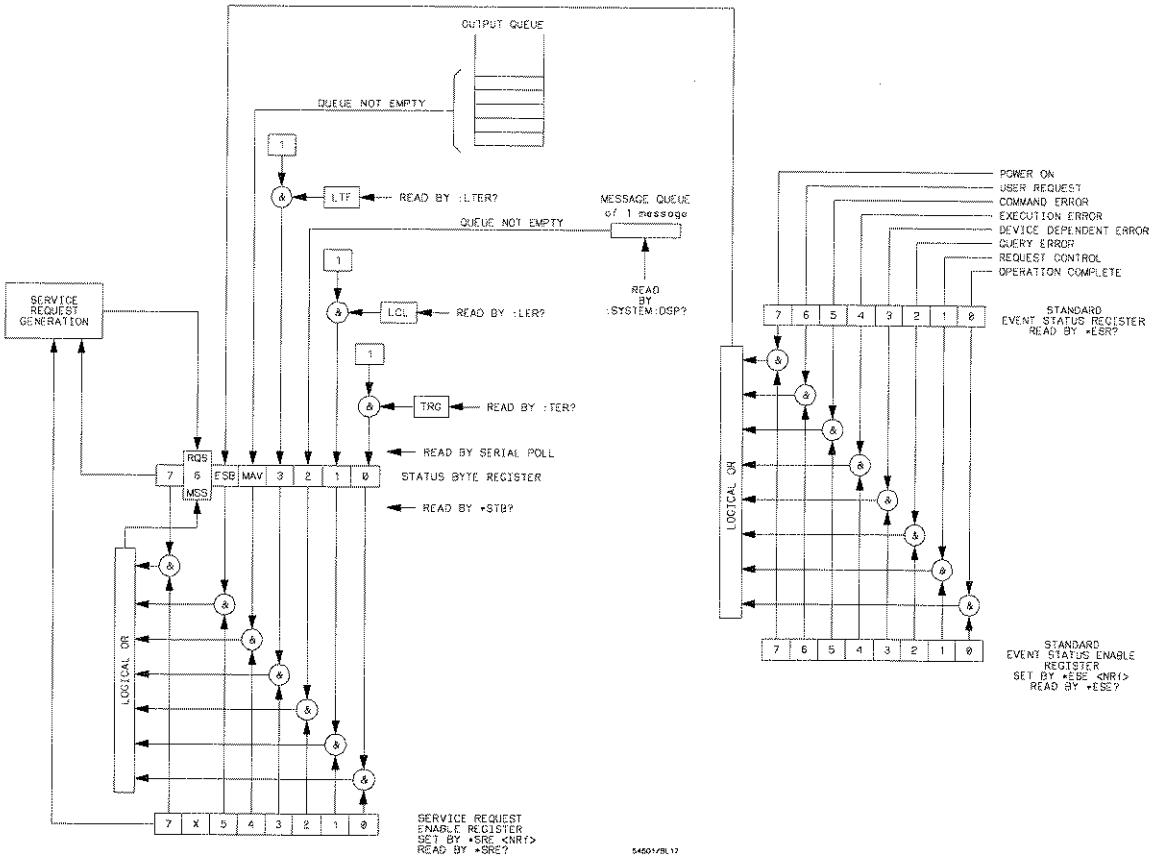
Status Reporting

Introduction

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 25–1



Status Reporting Data Structures

Bit Definitions

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

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

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

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

MSG Message. Indicates whether there is a message in the message queue.

PON Power on. Always 0 in the HP 545XXA.

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

CME Command error. Indicates whether the parser detected an error.

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

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

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

RQC Request control. Indicates whether the device is requesting control. The HP 545XXA never requests control.

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

LCL Local. Indicates whether a remote-to-local transition has occurred.

TRG Trigger. Indicates whether a trigger has been received.

LTF Limit test failure. Indicates whether a limit test or compare test failure has occurred.

Operation Complete (*OPC)

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

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

Trigger Bit (TRG)

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

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

```
OUTPUT 707;"*SRE 1"    ! enables a trigger service request.
                       ! the next trigger will generate an SRQ.
OUTPUT 707;":TER?"     ! queries the TRG event register, thus
ENTER 707;A$           ! clearing it.
                       ! the next trigger can now generate an
                       ! SRQ
```

Status Byte

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

Serial Poll

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

Using Serial Poll

This example shows how to use the service request by conducting a serial poll of all instruments on the bus. In this example, assume that there are two instruments on the bus: an oscilloscope at address 7 and a printer at address 1. These address assumptions are made throughout this manual. It is also assumed that you are operating on Interface Select Code 7.

The program command for serial poll using HP BASIC 5.0 is Stat=SPOLL(707). The address 707 is the address of the oscilloscope in this example. The command for checking the printer is Stat=SPOLL(701) because the address of that instrument is 01 on bus address 7. This command reads the contents of the remote interface Status Register into the variable called Stat. At that time bit 6 of the variable Stat can be tested to see if it is set (bit 6=1).

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

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

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

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

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

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

26

Error Messages

Introduction

This chapter lists the error messages that are returned by the parser on the oscilloscope.

Error Number	Description
00	No error
11	Questionable horizontal scaling
12	Edges required not found
13	Not a 545XXA command
70	RAM write protected
–100	Command error
–101	Invalid character
–102	Syntax error
–103	Invalid separator
–104	Data type error
–105	GET not allowed
–108	Parameter not allowed
–109	Missing parameter
–112	Program mnemonic too long
–113	Undefined header
–121	Invalid character in number
–123	Numeric overflow
–124	Too many digits
–128	Numeric data not allowed
–130	Suffix error
–131	Invalid suffix
–138	Suffix not allowed
–140	Character data error
–141	Invalid character data
–144	Character data too long
–148	Character data not allowed
–150	String data error
–151	Invalid string data
–158	String data not allowed
–160	Block data error
–161	Invalid block data

Error Number	Description
–168	Block data not allowed
–170	Expression error
–171	Invalid expression
–178	Expression data not allowed
–180	Macro error
–181	Invalid outside macro define
–183	Invalid inside macro define
–200	Execution error
–211	Trigger ignored
–221	Settings conflict
–222	Data out of range
–223	Too much data
–250	Mass Storage error
–251	Missing mass storage
–252	Missing media
–253	Corrupt media
–254	Media full
–255	Directory full
–256	File name not found
–257	File name error
–258	Media protected
–270	Macro error
–272	Macro execution error
–273	Illegal macro label
–276	Macro recursion error
–277	Macro redefined not allowed
–310	System error
–350	Too many errors
–400	Query error
–410	Query INTERRUPTED
–420	Query UNTERMINATED
–430	Query DEADLOCKED
–440	Query UNTERMINATED after indefinite response

27

Algorithms

Introduction

One of the primary features of the oscilloscope is its ability to make automatic measurements on displayed waveforms. This chapter provides details on how automatic measurements are calculated and offers some tips on how to improve results.

Measurement Setup

Measurements typically should be made at the fastest possible sweep speed for the most accurate measurement results. The entire portion of the waveform that is to be measured must be displayed on the oscilloscope:

- At least one complete cycle must be displayed for period or frequency measurements.
- The entire pulse must be displayed for width measurements.
- The leading edge of the waveform must be displayed for rise time measurements and all other edge measurements.
- The trailing edge of the waveform must be displayed for fall time measurements and all other edge measurements.

Making Measurements

If more than one waveform, edge, or pulse is displayed, the measurements are made on the first (leftmost) portion of the displayed waveform that can be used. If there are not enough data points, the oscilloscope displays ≤ with the measurement results. This is to remind you that the results may not be as accurate as possible. It is recommended that you re-scale the displayed waveform and make your measurement again.

When any of the standard measurements are requested, the oscilloscope first determines the top-base voltage levels at 100%-0%. From this information, it can determine the other important voltage values (10%, 90%, and 50%) needed to make the measurements. The 10% and 90% voltage values are used in the rise time and fall time measurements, as well as in all other edge measurements. The 10% and 90% values are also used to determine the 50% value. The 50% voltage value is used for measuring frequency, period, pulse width, and duty cycle.

Top-Base

Top-Base is the heart of most automatic measurements. It is used to determine V_{top} and V_{base}, the 0% and 100% voltage levels at the top and bottom of the waveform. From this information the oscilloscope can determine the 10%, 50%, and 90% points, which are also used in most measurements. The top or base of the waveform is not necessarily the maximum or minimum voltage present on the waveform. Consider a pulse that has slight overshoot. It would be wrong to select the highest point of the waveform as the top since the waveform normally rests below the perturbation.

Top-Base performs a histogram on the waveform and finds the most prevalent point above and below the waveform midpoint. The most prevalent point is one that represents greater than approximately 5% of the total display points (501) and is considered to be either the top or base. If no point accounts for more than 5% of the total, then the top is chosen as the absolute maximum and the base is chosen as the absolute minimum.

Edge Definition

Both rising and falling edges are defined as transitional edges that must cross three thresholds.

A rising edge must cross the lower threshold in a positive direction (defining it as a rising edge), cross the mid threshold (any number of crossings, both positive and negative are permissible) and then cross the upper threshold without any additional crossing of the lower threshold.

A falling edge must cross the upper threshold in a negative direction, cross the mid threshold (any number of times), and then cross the lower threshold without any additional crossing of the upper threshold.

Most time measurements are made based on the position of the first crossing of the middle threshold.

Algorithm Definitions

This section lists the definitions that the measurements in the oscilloscope are based on.

delay

There are three types of delay measurements:

- Jitter.
- Standard.
- User-defined.

Jitter occurs only under the following circumstances:

- The standard/user-defined key is set to standard.
- Two delay parameters are the same.
- The display mode is set to envelope.

if

first edge on minimum waveform is rising

then

delay = mid-threshold of first rising edge of max waveform minus mid-threshold of first rising edge on min waveform

else

delay = mid-threshold of first falling edge on min waveform minus mid-threshold of first falling edge on max waveform

The standard delay measurement occurs in the standard mode (not user-defined) and is not a jitter measurement.

standard delay = mid-threshold of the first edge of the second parameter minus the mid-threshold of the first edge of the first parameter

Negative delay is possible.

User defined delay = second channel edge minus first channel edge

+ width The + width algorithm has standard and user-defined considerations.

if

first edge is rising

then

+ width = mid-threshold crossing of first falling edge minus mid-threshold crossing of first rising edge

else

+ width = mid-threshold crossing of second falling edge minus mid-threshold crossing of first rising edge

The user-defined definition is the same as the standard definition except for the user-defined threshold.

– width The – width algorithm has standard and user-defined considerations:

if

first edge is rising

then

– width = second rising edge minus first falling edge

else

– width = first rising edge minus first falling edge

Period

if

first edge is rising

then

period = second rising edge minus first rising edge

else

period = second falling edge minus first falling edge

Frequency frequency = 1/period

Duty Cycle duty cycle = (+ width/period) * 100

> The + width for duty cycle is always calculated using the mid-threshold.

Rise time Rise time = time at upper threshold minus time at lower threshold

Fall time Fall time = time at lower threshold minus time at upper threshold

V_{max} Vmax = voltage of the maximum point on the screen

V_{min} Vmin = voltage of the minimum point on the screen

V_{p-p} Vp-p = Vmax minus Vmin

V_{top} Vtop = most prevalent point above waveform midpoint

V_{base} Vbase = most prevalent point below waveform midpoint

V_{amp} Vamp = Vtop minus Vbase

V_{avg} The average voltage of the first cycle of the displayed signal is measured. If a complete cycle is not present, the oscilloscope averages all data points.

V_{rms} (ac) The ac rms voltage of the first cycle of the displayed signal is measured. If a complete cycle is not present, the measurement computes ac rms on all data points.

$$V_{rms(ac)} = \left[\frac{1}{n} \sum_{j=0}^{n-1} V_j^{\,2} - \left(\frac{1}{n} \sum_{j=0}^{n-1} V_j \right)^{2} \right]^{\frac{1}{2}}$$

V_{rms} (dc) The true rms voltage of the first cycle of the displayed signal is measured.

$$V_{rms(dc)} = \left[\frac{1}{n} \sum_{j=0}^{n-1} V_j^{\,2} \right]^{\frac{1}{2}}$$

Integrate The equation is the integral of the channel, where I represents the integral and C represents the channel. The integral is calculated by adding the voltage points multiplied by the time bucket width, Δt.

$$I_C = \sum_{j=0}^{n-1} V_j \, \Delta t$$

Differentiate

$$d_1 = 0$$

$$d_n = \frac{c(n) - c(n-1)}{\Delta t}$$

The equation is the differential waveform of the channel, where d represents the differential and c represents the channel. The differential is the voltage differences between consecutive points in time divided by the time bucket width, Δt.

Index

A
Acquire Subsystem, 8–2
  COUNt Command/Query, 8–9
  POINts Command/Query, 8–10, 19–16 to 19–17
  Syntax Diagram, 8–2
  TYPE Command/Query, 8–12
ADD Command, 13–6
add to memory, 6–15
advisory line, 7–6
Algorithms, 27–1, 27–3 to 27–8
ALL? Query, 16–15
alpha argument, 3–3
Arguments, 1–6
AUTO time base mode, 20–5
automatic measurements, 27–2
Automatic Top-Base, 27–4
Autoscale
  undo, 5–20
AUToscale Command, 6–6
AVERAGE Type, 23–5
Averaging Mode, 8–4

B
BASIC, 1–3
BLANk command, 6–26
Block data, 1–5, 2–14
BNC Command/Query, 6–9
BYTE format, 23–10

C
Calibrate Subsystem, 9–2
  SETup? Query, 9–4, 10–14, 24–9
  Syntax Diagram, 9–2
  TNULl Command/Query, 9–5
carriage return, 3–7
CDIRectory, 11–5
CENTered Command, 21–15
Channel Subsystem, 10–2, 17–2, 18–2
  ECL Command, 10–7
  Syntax Diagram, 10–3, 15–3 to 15–4, 17–3, 18–3, 19–4, 22–4
  TTL Command, 10–15
Character data, 1–11, 2–11
Character program data, 1–11, 2–11
CLEar Command, 18–3
CLEAR DISPLAY, 6–11
*CLS (clear status) Command, 5–6 to 5–8, 5–19
CME - command error, 25–4
Combining commands, 1–8
Command, 1–5, 2–10
  :MEASure:SCRatch, 16–47
  :MEASure:VFIFty, 16–66
  :MMEMory:STORe, 17–7
  :PMEMory:CLEar, 18–3
  :PMEMory:MERGe, 18–5
  :SEQuential:TTAGs?, 19–16
  :TRIGger:CENTered, 21–15
  :TRIGger:GLITch:CENTered, 21–24
  Acquire Subsystem, 8–2
  ACQuire:COMPlete, 8–7
  ACQuire:COUNt, 8–9
  ACQuire:POINts, 8–10
  ACQuire:TYPE, 8–12
  AUToscale, 6–6
  BEEPer, 6–7
  BLANk, 6–8
  BNC, 6–9
  Calibrate Subsystem, 9–2
  CALibrate:TNULl, 9–5
  CDIRectory, 11–5
  Channel Subsystem, 10–2, 15–2, 17–2, 18–2, 19–2, 22–2
  CHANnel :COUPling, 10–5
  CHANnel :DISPlay, 10–6
  CHANnel :HFReject, 10–8
  CHANnel :LFReject, 10–9
  CHANnel :OFFSet, 10–10
  CHANnel :PROBe, 10–11
  CHANnel :RANGe, 10–13
  CHANnel :TTL, 10–15
  CHANnelN:ECL, 10–7
  *CLS, 5–5 to 5–6
  Common Commands, 3–4, 5–2
  DATA:ASCii, 9–3
  DELete, 11–5 to 11–6
  DIGitize, 2–7, 6–10
  Display Subsystem, 12–2
  DISPlay:COLumn, 12–6
  DISPlay:CONNect, 12–7
  DISPlay:DATA, 12–8
  DISPlay:FORMat, 12–10
  DISPlay:GRATicule, 12–11
  DISPlay:INVerse, 12–12
  DISPlay:LINE, 12–13
  DISPlay:MARKer|TMARker|VMARker, 12–24
  DISPlay:MASK, 12–14
  DISPlay:PERSistence, 12–16
  DISPlay:ROW, 12–18
  DISPlay:SCReen, 12–19
  DISPlay:SOURce, 12–21
  DISPlay:STRing, 12–22
  DISPlay:TEXT, 12–23
  *DMC, 5–7
  *EMC, 5–8
  ERASe, 6–11
  *ESE, 5–9
  *ESR, 5–11
  EXTernal:HFReject, 24–5
  EXTernal:LFReject, 24–6
  EXTernal :COUPling, 24–4
  EXTernal :PROBe, 24–7
  EXTernal :RANGe, 24–8
  FORmat, 11–8
  Function Subsystem, 13–2
  FUNCtion{1|2|3|4}:SUBTract, 13–28
  FUNCtion{1|2|3|4}:VERSus, 13–29
  FUNCtionN:ADD, 13–6
  FUNCtionN:DIFF, 13–7
  FUNCtionN:INTegrate, 13–11
  FUNCtionN:INVert, 13–12
  FUNCtionN:ONLY, 13–18
  FUNCtionN:RANGe, 13–24
  Hardcopy Subsystem, 14–2
  HARDcopy:PLOT:AREA, 14–9
  HEADer, 2–10, 7–10
  *IDN, 5–13 to 5–14
  LER, 6–12
  *LMC, 5–15
  LOAD, 11–9
  LONGform, 2–10, 7–13
  *LRN, 5–16
  LTER, 6–13
  MDIRectory, 11–10
  Measure Subsystem, 16–2
  MEASure:COMPare, 16–16
  MEASure:DEFine, 16–19
    MEASure:DUTycycle, 16–24
    MEASure:FREQuency, 16–32
    MEASure:LIMittest, 16–33
    MEASure:OVERshoot, 16–39
    MEASure:PREShoot, 16–42
    MEASure:PWIDth, 16–43
    MEASure:RISetime, 16–46
    MEASure:SCRatch, 16–47

MEASure:VACRms, 16–60
MEASure:VAMPlitude, 16–61
MEASure:VAVerage, 16–62
MEASure:VBASe, 16–63
MEASure:VDCRms, 16–64
MEASure:VMAX, 16–67
MEASure:VMIN, 16–68
MEASure:VPP, 16–69
MEASure:VTOP, 16–75
MENU, 6–14, 6–16
MERGe, 6–15
MWINdow, 16–37
*OPC, 5–17
*OPT, 5–18
PLOT, 6–17
*PMC, 5–19
POWerup, 6–18
PRINt, 6–19
PWD?, 11–11
*RCL, 5–20
Root Level Commands, 3–4, 6–2
*RST, 5–21
RUN, 6–20
*SAV, 5–23
SERial, 6–21
SETup?, 9–4, 10–14, 24–9
SIMage, 11–12
*SRE, 5–24
STATus, 6–22
*STB, 5–26
STOP, 6–23
STORe, 6–24, 11–10, 11–12 to 11–13
Subsystem Commands, 3–7
SYSTem:DATE, 7–5
SYSTem:DSP, 6–18, 7–6, 7–14
SYSTem:ERRor, 7–7
SYSTem:HEADer, 7–10
SYSTem:KEY, 7–11
SYSTem:LONGform, 7–13
SYSTem:SETup, 7–15
SYSTem:TIME, 7–16
TER, 6–25
*TRG, 5–28
Trigger Mode, 21–8
Trigger Subsystem, 21–2
TRIGGER:CONDITION, 21–10 to 21–11, 21–14
TRIGGER:DELAY, 21–12
TRIGGER:FIELD, 21–13
TRIGGER:HOLDOFF, 21–10, 21–14
TRIGGER:LEVEL, 21–8 to 21–10
TRIGGER:LOGIC, 21–11
TRIGGER:OCCURRENCE, 21–12 to 21–13
TRIGGER:OCCURRENCE:SLOPE, 21–12 to 21–13
TRIGGER:OCCURRENCE:SOURCE , 21–12
TRIGGER:PATH, 21–10 to 21–11
TRIGGER:POLARITY, 21–13
TRIGGER:QUALIFY, 21–13
TRIGGER:SLOPE, 21–9, 21–48
TRIGGER:SOURCE, 21–9, 21–13, 21–49
TRIGGER:STANDARD, 21–13, 21–50
TRIGGER:TIME, 21–12
*TST, 5–29
UTILity[GMARkers|LABels| FACTor], 7–17
VIEW, 6–26
*WAI, 5–30
Command structure, 2–6
command tree, 3–2, 3–4 to 3–8
Command Types, 3–4
Command/Query
:MEASure:FREQuency, 16–31 to 16–32
:MEASure:NWIDth, 16–38
:MEASure:OVERshoot, 16–39
:MEASure:PERiod, 16–40
:MEASure:POSTfailure, 16–41
:MEASure:PREShoot, 16–42
:MEASure:PWIDth , 16–43
:MEASure:RISetime, 16–46
:MEASure:SOURce, 16–48
:MEASure:STATistics, 16–49
:MEASure:STATistics:MODE, 16–50
:MEASure:TSTArt, 16–54
:MEASure:TSTOp, 16–55
:MEASure:UNITs, 16–57
:MEASure:UPPer, 16–58
:MEASure:VACRms, 16–60
:MEASure:VAMPlitude, 16–61
:MEASure:VAVerage, 16–62
:MEASure:VBASe, 16–63
:MEASure:VDCRms, 16–64
:MEASure:VMAX, 16–67
:MEASure:VMIN, 16–68
:MEASure:VPP, 16–69
:MEASure:VRELative, 16–70
:MEASure:VSTArt, 16–72
:MEASure:VSTOp, 16–73
:MEASure:VTOP , 16–75
:MEASure:WCOMpare:ALLowance, 16–77, 16–83
:MEASure:WCOMpare:COMPare, 16–78
:MEASure:WCOMpare:DESTination , 16–79
:MEASure:WCOMpare: HALLowance, 16–81
:MEASure:WCOMpare:POSTfailure , 16–82
:MEASure:WCOMpare: VALLowance, 16–77, 16–83
:MEASure:WCOMpare:WTESt?, 16–84
:MMEMory:DISPlay, 17–4
:MMEMory:FNUMber, 17–5
:MMEMory:SOURce, 17–6
:PMEMory:DISPlay, 18–4
:SEQuential:DISPlay, 19–5
:SEQuential:EXCLude, 19–6
:SEQuential:INCLude, 19–8
:SEQuential:NPOINts, 19–10
:SEQuential:NSEGments, 19–11
:SEQuential:SNUMber, 19–14
:SEQuential:SOURce , 19–15
:TIMebase:DELay, 20–4
:TIMebase:MODE, 20–5
:TIMebase:RANGe, 20–6
:TIMebase:REFerence, 20–7
:TIMebase:RLENgth, 20–8
:TIMebase:SAMPle:CLOCk, 20–11
:TRIGger:CONDition, 21–16
:TRIGger:COUPling, 21–18
:TRIGger:DELay, 21–19
:TRIGger:DELay:SLOPe, 21–20
:TRIGger:DELay:SOURce, 21–21
:TRIGger:FIELd, 21–22

:TRIGger:GLITch:HOLDoff, 21–25
:TRIGger:GLITch:LEVel, 21–26
:TRIGger:GLITch:SOURce, 21–27
:TRIGger:GLITch:WIDth, 21–28
:TRIGger:HOLDoff, 21–29
:TRIGger:LEVel, 21–30
:TRIGger:LINE, 21–31
:TRIGger:LOGic, 21–32
:TRIGger:MODE, 21–33
:TRIGger:NREJect, 21–37
:TRIGger:OCCurrence, 21–38
:TRIGger:OCCurrence:SLOPe, 21–39
:TRIGger:OCCurrence:SOURce, 21–40
:TRIGger:PATH, 21–41
:TRIGger:POLarity, 21–42
:TRIGger:QUALify, 21–43
:WAVeform:DATA, 23–12
:WAVeform:FORMat, 23–14
:WAVeform:PREamble, 23–16
:WAVeform:SOURce, 23–18
:WMEMory:{1|2|3|4}:XOFFset, 22–8
:WMEMory{1|2|3|4}:DISPLay, 22–4
:WMEMory{1|2|3|4}:GET, 22–5
:WMEMory{1|2|3|4}:PROTect, 22–6
:WMEMory{1|2|3|4}:XRANge, 22–9
ADD, 13–6
CONNect, 12–7
DESTination, 16–22
DIFF, 13–7
DISPlay, 15–5
HARDcopy:LENGth, 14–6
HARDcopy:MODE, 14–7
HARDcopy:PAGE, 14–8
LEVel, 13–13
MAGNify, 13–14
MEASure:DELay, 16–21
MEASure:EANalysis, 16–29
MEASure:ESTArt, 16–25
MEASure:ESTOp, 16–27
MEASure:LOWer, 16–34
MEASure:MODE, 16–36
MODE, 15–6
OFFSet, 13–17
PAGE, 14–8
PEAK, 13–19
PLOT:{PEN|COLor}, 14–11
PLOT:AREA, 14–9
PLOT:INITialize, 14–10
SPAN, 13–27
WINDow, 13–30
X1Position, 15–8
X1Y1source, 15–10
X2Position, 15–9
X2Y2source, 15–11
Y1Position, 15–13
Y2Position, 15–14
Command/Query
DUTycycle, 16–24
Common command header, 1–7
common commands, 3–4, 5–2
*CLS, 5–6
*DMC, 5–7
*EMC, 5–8
*ESE, 5–9
*ESR Query, 5–11
*GMC Query, 5–13
*IDN Query, 5–14
*LMC Query, 5–15
*LRN Query, 5–16
*OPC Command/Query, 5–17
*OPT Query, 5–18
*PMC, 5–19
*RCL Command, 5–20
*RST Command, 5–21
*SAV Command, 5–23
*SRE Command/Query, 5–24
*STB , 5–26
*TRG Command, 5–28
*TST, 5–29
*WAI Command, 5–30
Communication, 1–3
COMPare Command, 16–16
Compound command header, 1–7
compound header, 3–7
COMPRESSED format, 23–10
CONDition Command/Query, 21–16
Controllers, 1–3
COUNt Command/Query, 8–9
COUPling Command/Query, 10–5, 21–18, 24–4
CURSor? Query, 16–18

D

Data Acquisition Types, 23–5 to 23–8
DATA Command/Query, 12–8, 23–12
DATE Command/Query, 7–5
DC RMS, 16–64
DDE - device specific error, 25–4
DEFine Command, 16–19
Definite-length block response data, 2–14
DELay Command/Query, 16–21, 20–4, 21–19
Delay Trigger Mode, 21–12, 21–16
DELay:SLOPe Command/Query, 21–20
DELay:SOURce Command/Query, 21–21
DELete, 11–6
DELete Command, 11–6
deleting files, 11–6
DESKJET, 14–7
DESTination Command/Query, 16–22
Device address, 1–5
DIFF Command, 13–7
differentiate, 13–7
DIGitize Command, 2–7, 6–10
DIRectory?, 11–7
DIRectory? Query, 11–7
Disk Subsystem
DIRectory? Query, 11–7
Disk Subsystem Command
DELete, 11–6
FORMat Command, 11–8
LOAD, 11–9
STORe, 11–12 to 11–13
DISK:CDIRectory, 11–5
DISK:MDIRectory, 11–10
DISK:PWD?, 11–11
DISPlay
Command\Query, 12–6
DISPlay Command/Query, 13–8, 15–5 to 15–6, 17–4, 18–4, 19–5, 22–4
display mode, 8–4
Display Subsystem, 12–2
COLumn Command\Query, 12–6
CONNect Command/Query, 12–7
DATA Command/Query, 12–8
FORMat Command/Query, 12–10
GRATicule, 12–11
INVerse Command/Query, 12–12
LINE Command, 12–13
MARKer|TMARker|VMARker

Command/Query, 12–24
MASK Command/Query, 12–14
PERSistence Command/Query, 12–16
ROW Command/Query, 12–18
SCReen Command/Query, 12–19
SETup? Query, 12–20
SOURce Command/Query, 12–21
STRing command, 12–22
Syntax Diagram, 12–3
TEXT Command, 12–23
DISPlay:SCReen, 12–23
DOS file compatibilty, 11–2
DSP Command/Query, 7–6
Duplicate mnemonics, 1–8
duty cycle, 27–7
DUTycycle Command/Query, 16–24

E
EANalysis Command/Query, 16–29
ECL
undo, 5–20
ECL Command, 10–7
Edge Definition, 27–4
Embedded strings, 1–3, 1–5, 1–11
Enter statement, 1–3
Envelope Mode, 8–5
ENVELOPE Type, 23–5
EOI, 1–12
ERASe Command, 6–11
ERASE PMEMORY0, 6–11
error number, 7–7
ERRor query, 7–7
error queue, 5–6
Errors
messages, 26–2
ESB - event status bit, 25–4
*ESE Command
Command/Query, 5–9
*ESR Query, 5–11
ESR, event summary, 5–9
ESTArt Command/Query, 16–25
ESTOp Command/Query, 16–27
Event Status Register, 5–11
Example Program, 2–6, 4–2
Channel Setup, 4–4, 4–6, 4–10, 4–13, 4–19, 4–23, 4–26, 4–32, 4–36, 4–38
EXCLude Command/Query, 19–6
EXE - execution error, 25–4
Exponents, 1–11
External Trigger Subsystem
Syntax Diagram, 24–3

F
FACTor, 7–17
fall time, 27–7
fall time measurement, 16–12
fall time measurements, 27–3
FFT Command, 13–9
FIELd Command/Query, 21–22
file
format, 11–14 to 11–15
filenames, 11–2
FNUMber Command/Query, 17–5
FORmat, 11–8
FORMat Command, 11–8
FORMat Command/Query, 12–10, 23–14
formatting disks, 11–8
Fractional values, 1–11
frequency, 27–6
FREQuency Command/Query, 13–10, 16–32
frequency measurement, 16–12
Function Subsystem, 13–2
ADD Command, 13–6
DIFF Command, 13–7
DISPlay Command/Query, 13–8
FFT Command, 13–9
FREQuency Command/Query, 13–10
INTegrate, 13–11
INTegrate Command, 13–11
INVert Command, 13–12
LEVel Command/Query, 13–13
MAGNify Command/Query, 13–14
MODe? Query, 13–15
MULTiply Command, 13–16
OFFSet Command/Query, 13–17
ONLY Command, 13–18
PEAK Command/Query, 13–19
POINts Command, 13–21
RANGe Command/Query, 13–24
SETup? Query, 13–25
SPAN Command/Query, 13–27
SUBTract Command, 13–28
Syntax Diagram, 13–4
VERSus, 13–29
VERSus Command, 13–29
WINDow Command/Query, 13–30

G
GET Command/Query, 22–5
GLITch Command/Query, 21–23
GLITch:CENTered Command, 21–24
GLITch:HOLDoff Command/Query, 21–25
GLITch:LEVel Command/Query, 21–26
GLITch:SOURce Command/Query, 21–27
GLITch:WIDth Command/Query, 21–28
GMARkers, 7–17
*GMC Query, 5–13
GT (Greater Than), 21–10

H
Hardcopy
Plot, 6–17
Print, 6–19
Hardcopy Subsystem, 14–2
LENGth Command/Query, 14–6
MODE Command/Query, 14–7
PAGE Command/Query, 14–8
PLOT:{PEN|COLor} Command/Query, 14–11
PLOT:AREA Command/Query, 14–9
PLOT:INITialize Command/Query, 14–10
Syntax Diagram, 14–3 to 14–4
HEADer command, 2–10
HEADer Command/Query, 7–10
Headers, 1–5 to 1–6, 3–3
HOLDoff Command/Query, 21–29
Host language, 1–5

I
*IDN Query, 5–14, 6–21
IEEE 488.2, 3–4
Standard, 1–2
INCLude Command/Query, 19–8
Infinity Representation, 3–9
Initialization, 2–3
input
vertical, 6–6
Instruction headers, 1–5
Instruction syntax, 1–4

Instructions, 1–4
integrate, 13–11
INTegrate Command, 13–11
INVert Command, 13–12

K
key codes, 7–11
KEY Command/Query, 7–11

L
LABels, 7–17
LASERJET, 14–7
LCL - local, 25–4
leading colon, 3–7 to 3–8
LENGth Command/Query, 14–6
LER Query, 6–12
level
  trigger, 6–6
LEVel Command/Query, 13–13, 21–30
Limit Test Event Register, 6–13
LIMittest Command, 16–33
LINE Command, 12–13
LINE Command/Query, 21–31
linefeed, 3–4
linefeed (CRLF), 3–7
*LMC Query, 5–15
LOAD, 11–9
LOAD Command, 11–9
LOGic Command/Query, 21–32
Long form, 1–10
LONGform command, 2–10
LONGform Command/Query, 7–13
LOWer Command/Query, 16–34
Lowercase, 1–10
*LRN Query, 5–16
LT (Less Than), 21–10
LTER Query, 6–13
LTF - limit test failure, 25–4

M
MAGNify Command, 13–14
MAGNify Command/Query, 13–14
Making Measurements, 16–13 to 16–14, 27–3
Marker Subsystem, 15–2
  DISPlay Command/Query, 15–5 to 15–6
  SETup? Query, 15–7
  X1Position Command/Query, 15–8
  X1Y1source Command/Query, 15–10
  X2Position Command/Query, 15–9
  X2Y2source Command/Query, 15–11
  XDELta? Query, 15–12
  Y1Position Command/Query, 15–13
  Y2Position Command/Query, 15–14
  YDELta? Query, 15–15
MARKer|TMARker|VMARker Command/Query, 12–24
MASK Command/Query, 12–14
MAV - message available, 25–4
MDIRectory, 11–10
Measure Subsystem, 16–2
  :MEASure:COMPare Command, 16–16
  :MEASure:CURSor? Query, 16–18
  :MEASure:DEFine Command, 16–19
  :MEASure:DELay Command/Query, 16–21
  :MEASure:DESTination Command/Query, 16–22
  :MEASure:DUTycycle Command/Query, 16–24
  :MEASure:EANalysis Command/Query, 16–29
  :MEASure:ESTArt Command/Query, 16–25
  :MEASure:ESTOp Command/Query, 16–27
  :MEASure:FALLtime Command/Query, 16–31
  :MEASure:FREQuency Command/Query, 16–31 to 16–32
  :MEASure:LOWer Command/Query, 16–34
  :MEASure:NWIDth Command/Query, 16–38
  :MEASure:OVERshoot Command/Query, 16–39
  :MEASure:PERiod Command/Query, 16–40
  :MEASure:POSTfailure Command/Query, 16–41
  :MEASure:PRESHoot Command/Query, 16–42
  :MEASure:PWIDth Command/Query, 16–43
  :MEASure:RESults? Query, 16–44
  :MEASure:RISetime Command/Query, 16–46
  :MEASure:SOURce Command/Query, 16–48
  :MEASure:STATistics Command/Query, 16–49
  :MEASure:STATistics:MODE Command/Query, 16–50
  :MEASure:TDELta? Query, 16–51
  :MEASure:TMAX? Query, 16–52
  :MEASure:TMIN? Query, 16–53
  :MEASure:TSTArt Command/Query, 16–54
  :MEASure:TSTOp Command/Query, 16–55
  :MEASure:TVOLt? Query, 16–56
  :MEASure:UNITs Command/Query, 16–57
  :MEASure:UPPer Command/Query, 16–58
  :MEASure:VACRms Command/Query, 16–60
  :MEASure:VAMPlitude Command/Query, 16–61
  :MEASure:VAVerage Command/Query, 16–62
  :MEASure:VBASe Command/Query, 16–63
  :MEASure:VDCRms Command/Query, 16–64
  :MEASure:VDELta? Query, 16–65
  :MEASure:VFIFty Command , 16–66
  :MEASure:VMAX Command/Query, 16–67
  :MEASure:VMIN Command/Query, 16–68
  :MEASure:VPP Command/Query, 16–69
  :MEASure:VRELative Command/Query, 16–70
  :MEASure:VSTArt Command/Query, 16–72
  :MEASure:VSTOp Command/Query, 16–73
  :MEASure:VTIMe? Query, 16–74
  :MEASure:VTOP Command/Query, 16–75
  :MEASure:WCOMpare:ALLowance Command/Query, 16–77
  :MEASure:WCOMpare:COMPare Command/Query, 16–78
  :MEASure:WCOMpare:DESTination Command/Query, 16–79
  :MEASure:WCOMpare:HALLowance Command/Query, 16–81
  :MEASure:WCOMpare:POSTfailure Command/Query, 16–82
  :MEASure:WCOMpare:VALLowance Command/Query, 16–77

:MEASure:WCOMpare:WTESt Command/Query, 16–84
ALL? Query, 16–15
LIMittest Command, 16–33
MODE Command/Query, 16–36
SCRatch Command, 16–47
VMAX Command/Query, 16–67
VMIN Command/Query, 16–68
VPP Command/Query, 16–69
VTOP Command/Query, 16–75
Measurement Error, 16–12
Measurement Setup, 16–12, 27–3
MENU Command/Query, 6–14
MERGe Command, 6–15, 18–5
- width, 27–6
mnemonic, 3–3
Mode
Delay Trigger, 21–12
Pattern Trigger, 21–10
State Trigger, 21–11, 21–14
TV Trigger, 21–13
MODE Command/Query, 14–7, 16–36, 20–5, 21–33
MODe? Query, 13–15
MSG - message, 25–4
MSS (Master Summary Status), 5–26
MSS - master summary status, 25–4
Multiple Memory (MMEMory) Subsystem
:MMEMory:DISPlay Command/Query, 17–4
:MMEMory:FNUMber Command/Query, 17–5
:MMEMory:SOURce Command/Query, 17–6
:MMEMory:STORe Command, 17–7
Multiple numeric variables, 2–14
Multiple program commands, 1–12
Multiple program data, 1–10
Multiple queries, 2–14
Multiple subsystems, 1–12
MULTiply Command, 13–16
MWINdow Command/Query, 16–37

N

- width, 27–6
NL, 1–12, 3–4
Normal Persistence mode, 8–4
NORMAL Type, 23–5
Notation Conventions and Definitions, 3–10
NPOints Command/Query, 19–10
NREJect Command/Query, 21–37
NSEGments Command/Query, 19–11
number of averages, 8–4
Numeric data, 1–11
Numeric program data, 1–11
Numeric variables, 2–13
NWIDth Command/Query, 16–38

O

OCCurrence Command/Query, 21–38
OCCurrence:SLOPe Command/Query, 21–39
OCCurrence:SOURce Command/Query, 21–40
offset
vertical, 6–6
OFFSet Command/Query, 10–10, 13–17
ONLY Command, 13–18
*OPC Command/Query, 5–17
OPC - operation complete, 25–4
operating the disk, 11–2
Operation Complete, 25–4
*OPT Query, 5–18
Order of commands, 2–4
Oscilloscope
errors, 26–2
Output command, 1–4
output queue, 5–17
OUTPUT statement, 1–3
Overlapped Commands, 3–9
OVERshoot Command/Query, 16–39

P

PAGE Command/Query, 14–8
PAINTJET, 14–7
Parallel Poll, 25–6
Parameters, 1–6
Parser, 2–3
PATH Command/Query, 21–41
Pattern Trigger Mode, 21–10, 21–16
PDETECT Type, 23–5
PEAK Command, 13–19
PEAK Command/Query, 13–19
period, 27–6
PERiod Command/Query, 16–40
period measurement, 16–12
PERSistence Commnand/Query, 12–16
PIMacro Command, 7–14
Pixel Memory (PMEMory) Subsystem
:PMEMory:CLEar Command, 18–3
:PMEMory:DISPlay Command/Query, 18–4
:PMEMory:MERGe Command, 18–5
:PMEMory:SETup? Query, 18–6
pixel memory, storing, 11–13
PLOT Query, 6–17
PLOT:{PEN|COLor} Command/Query, 14–11
PLOT:AREA Command/Query, 14–9
PLOT:COLor Command/Query, 14–11
PLOT:INITialize Command/Query, 14–10
+ width, 27–6
POINts Command, 13–21
POINts Command/Query, 8–10, 19–16 to 19–17
POINts? Query, 23–15
POLarity Command/Query, 21–42
PON - power on, 25–4
POSTfailure Command/Query, 16–41
POWerup Command/Query, 6–18
PREamble Command/Query, 23–16
PREShoot Command/Query, 16–42
PRINt Query, 6–19
Program data, 1–6, 1–10
Program example, 2–6, 4–2
Channel Setup, 4–4, 4–6, 4–10, 4–13, 4–19, 4–23, 4–26, 4–32, 4–36, 4–38
program message, 3–8
Program message syntax, 1–4
Program message terminator, 1–12, 3–7
program message unit separator, 3–8
Program syntax, 1–4
PROTect Command/Query, 22–6
pulse width measurement, 16–12
PWD? Query, 11–11
PWIDth Command/Query, 16–43

Q

QUALify Command/Query, 21–43
Query, 1–5, 1–9, 2–10
:MEASure:RESults?, 16–44
:MEASure:TDELta?, 16–51
:MEASure:TMAX?, 16–52
:MEASure:TMIN?, 16–53

:MEASure:TVOLt?, 16–56
:MEASure:VDELta?, 16–65
:MEASure:VTIMe?, 16–74
:PMEMory:SETup?, 18–6
:SEQuential:SETup?, 19–13
:TIMebase:SETup?, 20–12
:TRIGger:SETup, 21–44
:WAVeform:POINts?, 23–15
:WAVeform:TYPE?, 23–19
:WAVeform:XINCrement?, 23–20
:WAVeform:XORigin?, 23–21
:WAVeform:XREFerence?, 23–22
:WAVeform:YINCrement?, 23–23
:WAVeform:YORigin?, 23–24
:WAVeform:YREFerence?, 23–25
:WMEMory :SETup?, 22–7
:WMEMory{1|2|3|4}:YOFFset?, 22–10
:WMEMory{1|2|3|4}:YRANGe?, 22–11
DIRectory?, 11–7
MEASure:ALL?, 16–15
MEASure:CURSor?, 16–18
MODe?, 13–15
SETup?, 12–20, 13–25, 15–7
XDELta?, 15–12
YDELta?, 15–15
Query command, 1–9
Query response, 2–9
query responses, 3–9
Question mark, 1–9
QYE - query error, 25–4

R

RANGE, 21–10
RANGe Command/Query, 13–24, 20–6
Rawdata Mode, 8–5, 23–7, 23–16
*RCL Command, 5–20
Realtime, 20–10
Recall
undo, 5–20
REFerence Command/Query, 20–7
Repetitive, 20–10
Response data, 2–14
Response Generation, 3–9
RESults? Query, 16–44
retrieval and storage, 11–2
rise time, 27–7
rise time measurement, 16–12
rise time measurements, 27–3
RISetime Command/Query, 16–46
RLENgth Command/Query, 20–8
Root Level Command
PCFRequency, 6–16
Root Level commands, 3–4, 6–2
AUToscale Command, 6–6
BNC Command/Query, 6–9
DIGitize Command, 6–10
ERASe Command, 6–11
LER Query, 6–12
LTER Query, 6–13
MENU Command/Query, 6–14
MERGe Command, 6–15
PLOT Query, 6–17
PRINt Query, 6–19
RUN Command, 6–20
SERial (Serial Number) Command, 6–21
STATus Query, 6–22
STOP Command, 6–23
STORe Command, 6–24
Syntax Diagram, 6–2
TER Query, 6–25
VIEW Command, 6–26
ROW Command/Query, 12–18
RQC - request control, 25–4
RQS - request service, 25–4
*RST Command, 5–21
RUN Command, 6–20

S

SAMPle Command/Query, 20–10
SAMPle:CLOCk Command/Query, 20–11
*SAV Command, 5–23
save register, 5–23
save/recall register, 5–20
SCRatch Command, 16–47
SCReen Command/Query, 12–19
seconds/division, 20–6, 22–9
sensitivity
vertical, 6–6
Separator, 1–6
SEQuential, 19–2
Sequential commands, 3–9
SEQuential Subsystem, 19–2
:SEQuential:DISPlay Command/Query, 19–5
:SEQuential:EXCLude Command/Query, 19–6
:SEQuential:INCLude Command/Query, 19–8
:SEQuential:NPOints Command/Query, 19–10
:SEQuential:NSEGments Command/Query, 19–11
:SEQuential:SETup? Query, 19–13
:SEQuential:SNUMber Command/Query, 19–14
:SEQuential:SOURce Command/Query, 19–15
:SEQuential:TTAGs? Command, 19–16
:SEQuential:TTDifference? Command, 19–17
serial number, 6–21
Serial Poll, 25–6 to 25–7
Service, 5–24
SETup Command/Query, 7–15
SETup Query, 21–44
setup, storing, 11–13
SETup? Query, 9–4, 10–14, 12–20, 13–25, 15–7, 18–6, 19–13, 20–12, 22–7, 24–9
707 (device address), 2–12
Short form, 1–10
SIMage, 11–12
Simple command header, 1–6
SINGLE time base mode, 20–5
SLOPe Command/Query, 21–48
SNUMber Command/Query, 19–14
SOURce Command/Query, 12–21, 16–48, 17–6, 19–15, 21–49, 23–18
Spaces, 1–6
SPAN Command/Query, 13–27
*SRE Command/Query, 5–24
standard + width, 27–6
standard - width, 27–6
STANdard Command/Query, 21–50
Standard Event Status Enable Register, 5–9
Standard Event Status Register, 5–9, 5–17
State Trigger Mode, 21–11, 21–14, 21–16
STATistics Command/Query, 16–49
STATistics:MODE Command/Query, 16–50
Status, 2–15
Status Byte, 5–24, 25–5
status data structures, 5–6
STATus Query, 6–22

status register, 5–5
Status registers, 2–15
Status Reporting, 25–2
*STB Command, 5–26
STOP Command, 6–23
storage and retrieval, 11–2
STORe, 11–14
STORe Command, 17–7
STRing command, 12–22
String variables, 2–12
Subsystem
  Acquire, 8–2
  Calibrate, 9–2
  Display, 12–2
  Function, 13–2
  HARDcopy, 14–2
  Measure, 16–2
  Trigger, 21–2
Subsystem commands, 3–4
Subsystem Channel, 10–2, 15–2, 17–2, 18–2
SUBTract Command, 13–28
sweep speed, 6–6
Syntax Diagram
  Acquire Subsystem, 8–2
  Calibrate Subsystem, 9–2
  Channel Subsystem, 10–3, 15–3 to 15–4, 17–3, 18–3, 19–4, 22–4
  Display Subsystem, 12–3
  External Trigger Subsystem, 24–3
  Function Subsystem, 13–4
  Hardcopy Subsystem, 14–3 to 14–4
  Root Level Commands, 6–2
  System Subsystem, 7–2
  Waveform Subsystem, 23–3
Syntax Diagram:Trigger Subsystem, 21–3
System Subsystem
  DATE Command/Query, 7–5
  DSP Command/Query, 7–6
  ERRor Query, 7–7
  HEADer Command/Query, 7–10
  KEY Command/Query, 7–11
  LONGform Command/Query, 7–13
  PMacro Command, 6–18, 7–14
  PPWerup Command, 6–18
  SETup Command/Query, 7–15
  Syntax Diagram, 7–2
  TIME Command/Query, 7–16

T

Talking to the instrument, 1–3
TDELta? Query, 16–51
TER Query, 6–25
Terminator, 1–12
TEXT Command, 12–23
THINKJET, 14–7
time bucket, 11–14
TIME Command/Query, 7–16
Timebase Subsystem
  :TIMebase:DELay Command/Query, 20–4
  :TIMebase:MODE Command/Query, 20–5
  :TIMebase:RANGe Command/Query, 20–6
  :TIMebase:REFerence Command/Query, 20–7
  :TIMebase:RLENgth Command/Query, 20–8
  :TIMebase:SAMPle Command/Query, 20–10
  :TIMebase:SAMPle:CLOCk Command/Query, 20–11
  :TIMebase:SETup? Query, 20–12
TMAX? Query, 16–52
TMIN Query, 16–53
TNULl Command/Query, 9–5
*TRG Command, 5–28
TRG - trigger, 25–4
Trigger Bit, 25–5
Trigger Condition command, 21–10 to 21–11, 21–14
Trigger Delay command, 21–12
Trigger Delay Source command, 21–12
Trigger Event, 21–12
Trigger Event command, 21–12
Trigger Field command, 21–13
Trigger Holdoff, 21–10, 21–14
trigger level, 6–6
Trigger Level command, 21–8 to 21–9
Trigger Logic command, 21–10 to 21–11
Trigger Mode, 21–8
  EDGE, 21–8 to 21–9
  TV, 21–8
  Valid Commands, 21–8
Trigger Occurrence command, 21–12 to 21–13
Trigger Occurrence Slope command, 21–13
Trigger Path command, 21–10 to 21–11
Trigger Polarity command, 21–13
Trigger Qualify command, 21–12 to 21–13
Trigger Slope command, 21–9
Trigger Source command, 21–9, 21–13
Trigger Standard command, 21–13
Trigger Subsystem, 21–2
  :TRIGger:CENTered Command, 21–15
  :TRIGger:CONDition Command/Query, 21–16
  :TRIGger:COUPling Command/Query, 21–18
  :TRIGger:DELay Command/Query, 21–19
  :TRIGger:DELay:SLOPe Command/Query, 21–20
  :TRIGger:DELay:SOURce Command/Query, 21–21
  :TRIGger:FIELd Command/Query, 21–22
  :TRIGger:GLITch Command/Query, 21–23
  :TRIGger:GLITch:CENTered Command, 21–24
  :TRIGger:GLITch:HOLDoff Command/Query, 21–25
  :TRIGger:GLITch:SOURce Command/Query, 21–27
  :TRIGger:GLITch:WIDth Command/Query, 21–28
  :TRIGger:HOLDoff Command/Query, 21–29
  :TRIGger:LEVel Command/Query, 21–26, 21–30
  :TRIGger:LINE Command/Query, 21–31
  :TRIGger:LOGic Command/Query, 21–32
  :TRIGger:MODE Command/Query, 21–33
  :TRIGger:NREJect Command/Query, 21–37
  :TRIGger:OCCurrence Command/Query, 21–38
  :TRIGger:OCCurrence:SLOPe Command/Query, 21–39
  :TRIGger:OCCurrence:SOURce Command/Query, 21–40
  :TRIGger:PATH Command/Query, 21–41
  :TRIGger:POLarity Command/Query, 21–42
  :TRIGger:QUALify Command/Query, 21–43
  :TRIGger:SETup Query, 21–44
  SLOPe Command/Query, 21–48
  SOURce Command/Query, 21–49
  STANdard Command/Query, 21–50

Syntax Diagram, 21–3
Trigger Time command, 21–12
TRIGGERED time base mode, 20–5
True RMS, 16–64
Truncation Rules, 3–3
*TST Query, 5–29
TSTArt Command/Query, 16–54
TSTOp Command/Query, 16–55
TTAGs? Command, 19–16
TTDifference? Command, 19–17
TTL
undo, 5–20
TTL Command, 10–15
TV Trigger Mode, 21–8, 21–13, 21–16
TVOLt? Query, 16–56
TYPE Command/Query, 8–12
TYPE Query, 23–19

U
UNITs Command/Query, 16–57
UPPer Command/Query, 16–58
Uppercase, 1–10
URQ - user request, 25–4
URQ, user request, 5–9
User-Defined Measurements, 16–12
UTILity{GMARkers|LABels|FACTor}, 7–17

V
VACRms Command/Query, 16–60
Vamp, 27–7
VAMPlitude Command/Query, 16–61
VAVerage Command/Query, 16–62
Vavg, 27–7
Vbase, 27–7
VBASe Command/Query, 16–63
VDCRms Command/Query, 16–64
VDELta? Query, 16–65
VERSus Command, 13–29
vertical input, 6–6
vertical offset, 6–6
vertical sensitivity, 6–6
Vertical Setup, 4–4, 4–6, 4–10, 4–13, 4–19, 4–23, 4–26, 4–32, 4–36, 4–38
VFIFty Command, 16–66
VIEW command, 6–8, 6–26
Vmax, 27–7
VMAX Command/Query, 16–67
Vmin, 27–7
VMIN Command/Query, 16–68
Voltage measurements, 16–13
Vp-p, 27–7
VPP Command/Query, 16–69
VRELative Command/Query, 16–70
Vrms, 27–8
VSTArt Command/Query, 16–72
VSTOp Command/Query, 16–73
VTIMe? Query, 16–74
Vtop, 27–7
VTOP Command/Query, 16–75

W
*WAI Command, 5–30
Waveform Data Conversion, 23–9
Waveform Memory (WMEMory) Subsystem, 22–2
:WMEMory:DISPlay Command/Query, 22–4
:WMEMory:GET Command/Query, 22–5
:WMEMory :SETup? Query, 22–7
:WMEMory{1|2|3|4}:PROTect, 22–6
:WMEMory{1|2|3|4}:XOFFset Command/Query, 22–8
:WMEMory{1|2|3|4}:XRANge Command/Query, 22–9
:WMEMory{1|2|3|4}:YOFFset? Query, 22–10
:WMEMory{1|2|3|4}:YRANge? Query, 22–11
Waveform Subsystem
:WAVeform:DATA Command/Query, 23–12
:WAVeform:FORMat Command/Query, 23–14
:WAVeform:POINts? Query, 23–15
:WAVeform:PREamble Command/Query, 23–16
:WAVeform:SOURce Command/Query, 23–18
:WAVeform:TYPE? Query, 23–19
:WAVeform:XINCrement? Query, 23–20
:WAVeform:XORigin? Query, 23–21
:WAVeform:XREFerence? Query, 23–22
:WAVeform:YINCrement? Query, 23–23
:WAVeform:YORigin? Query, 23–24
:WAVeform:YREFerence? Query, 23–25
Syntax Diagram, 23–3
waveform, storing, 11–13
WCOMpare:ALLowance Command/Query, 16–77
WCOMpare:COMPare Command/Query, 16–78
WCOMpare:DESTination Command/Query, 16–79
WCOMpare:HALLowance Command/Query, 16–81
WCOMpare:POSTfailure Command/Query, 16–82
WCOMpare:WTESt? Command/Query, 16–84
White space, 1–6
+ width, 27–6
WINDow Command/Query, 13–30
WORD format, 23–10

X
X1Position Command/Query, 15–8
X1Y1source Command/Query, 15–10
X2Position Command/Query, 15–9
X2Y2source Command/Query, 15–11
XDELta? Query, 15–12
XINCrement Query, 23–20
XOFFset Command/Query, 22–8
XORigin? Query, 23–21
XRANge Command/Query, 22–9
XREFerence? Query, 23–22

Y
Y1Position Command/Query, 15–13
Y2Position Command/Query, 15–14
YDELta? Query, 15–15
YINCrement? Query, 23–23
YOFFset? Query, 22–10
YORigin? Query, 23–24
YRANge? Query, 22–11
YREFerence? Query, 23–25

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.

Hewlett-Packard
P.O. Box 2197
1900 Garden of the Gods Road
Colorado Springs, CO 80901

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 54540 and 54520 Series Oscilloscope Programmer's Reference.*

Publication number
54542-97038

Printed in USA.

Edition dates are as follows:

First edition, November 1995

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