Programming Reference

HP 54501A
100 MHz Digitizing Oscilloscope

Publication 54501-90907

Printed in the U.S.A. June 1989

Printing History

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 the customer. The dates on the title page change only when a new edition is published.

A software code may be printed before the date; this indicates the version of the software product at the time the manual or update was issued. Many product updates and fixes 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.

Edition 1 June 1989 54501-90907

List of Effective Pages

The List of Effective Pages gives the date of the current edition and of any pages changed in updates to that edition. Within the manual, any page changed since the last edition is indicated by printing the date the changes were made on the bottom of the page. If an update is incorporated when a new edition of the manual is printed, the change dates are removed from the bottom of the pages and the new edition date is listed in the Printing History and on the title page.

Pages	Effective Date

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 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. However, warranty service for products installed by Hewlett-Packard and certain other products designated by Hewlett-Packard will be performed at the Buyer's facility at no charge within the Hewlett-Packard service travel area. Outside Hewlett-Packard service travel areas, warranty service will be performed at the Buyer's facility only upon Hewlett-Packard's prior agreement and the Buyer shall pay Hewlett-Packard's round trip travel expenses.

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 AND FITNESS FOR A PARTICULAR PURPOSE.

Exclusive Remedies

THE REMEDIES PROVIDED HEREIN ARE 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 assistance, contact your nearest Hewlett-Packard Sales and Service 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 Bureau of Standards, to the extent allowed by the Bureau's calibration facility, and to the calibration facilities of other International Standards Organization members.

Safety

This product has been designed and tested according to International Safety Requirements. To ensure safe operation and to keep the product safe, the information, cautions, and warnings in this manual must be heeded.

Introduction

This manual explains how to program the HP 54501A Digitizing Oscilloscope and lists the commands and queries associated with this instrument. It is divided into 17 chapters and 2 appendices.

Chapter 1 introduces you to the programming syntax required to program this instrument and provides some basic programming concepts to get you started programming..

Chapter 2 describes the interface functions and some general concepts of HP-IB.

Chapter 3 describes the operation of instruments that operate in compliance with the **IEEE 488.2** standard. This chapter also describes the **status reporting** features that are available over the HP-IB.

Chapter 4 covers the conventions which are used to program the instrument as well as conventions used in the remainder of this manual. This chapter also includes a complete **command tree** and **alphabetic command cross-reference.**

Chapter 5 lists the **Common Commands** which are the commands defined by IEEE 488.2. These commands control some functions that are common to all IEEE 488.2 instruments.

Chapter 6 lists the **Root Level Commands** which control many of the basic functions of the instrument.

Chapter 7 lists the **System Subsystem Commands** which control some basic functions of the oscilloscope.

Chapter 8 lists the **Acquire Subsystem Commands** which set the parameters for acquiring and storing data.

Chapter 9 lists the **Calibrate Subsystem Commands** which set time nulls (channel-to-channel skew).

Chapter 10 lists the **Channel Subsystem Commands** which control all Y-axis oscilloscope functions.

Chapter 11 lists the **Display Subsystem Commands** which control how waveforms, voltage and time markers, graticule, and text are displayed and written on the screen.

Chapter 12 lists the **Function Subsystem Commands** which control the waveform math functions of the oscilloscope.

Chapter 13 lists the **Hardcopy Subsystem Commands** which control the parameters used during the printing of waveforms.

Chapter 14 lists the **Measure Subsystem Commands** which select the automatic measurements to be made.

Chapter 15 lists the **Timebase Subsystem Commands** which control all X-axis oscilloscope functions.

Chapter 16 lists the **Trigger Subsystem Commands** which control the trigger modes and parameters for each trigger mode.

Chapter 17 lists the **Waveform Subsystem Commands** which provide access to waveform data, including active data from channels and functions as well as static data from waveform memories.

Appendix A provides details on how automatic measurements are calculated and offers some tips on how to improve results.

Appendix B contains **example programs** using the command set from the HP 54501A.

At the end of the manual is **Quick Reference Guide** that lists the commands and queries with their corresponding arguments and returned formats. Also, at the end of the manual is a complete **index** for easy reference of commands and functions.

Contents

Introduction

Chapter 1: Introduction to Programming an Oscilloscope

Introduction 1-1
Programming Syntax 1-2
Talking to the Instrument 1-2
Addressing the Instrument 1-3
Program Message Syntax 1-4
Separator 1-5
Command Syntax 1-5
Query Command 1-7
Program Header Options 1-8
Program Data 1-8
Program Message Terminator 1-9
Selecting Multiple Subsystems 1-10
Summary 1-10
Programming an Oscilloscope 1-11
Initialization 1-11
Autoscale 1-12
Setting Up the Instrument 1-12
Receiving Information from the Instrument 1-13
Response Header Options 1-14
Response Data Formats 1-15
String Variables 1-16
Numeric Variables 1-17
Definite-Length Block Response Data 1-18
Multiple Queries 1-19
Instrument Status 1-19
Digitize Command 1-20

Chapter 2: Interface Functions

Introduction 2-1
Interface Capabilities 2-1
Command and Data Concepts 2-1
Addressing 2-1
Remote, Local and Local Lockout 2-2
Bus Commands 2-3
  Device Clear 2-3
  Group Execute Trigger (GET) 2-3
  Interface Clear (IFC) 2-3
Status Annunciators 2-3

Chapter 3: Message Communication and System Functions

Protocols 3-1
  Functional Elements 3-1
  Protocol Overview 3-2
  Protocol Operation 3-2
  Protocol Exceptions 3-3
Syntax Diagrams 3-5
Syntax Overview 3-5
  Device Listening Syntax 3-8
  Device Talking Syntax 3-21
Common Commands 3-27
Status Reporting 3-28
  Bit Definitions 3-30
  Key Features 3-31
  Serial Poll 3-32
  Parallel Poll 3-34
  Polling HP-IB Devices. 3-36
  Configuring Parallel Poll Responses 3-36
  Conducting a Parallel Poll 3-37
  Disabling Parallel Poll Responses 3-37
  HP-IB Commands 3-37

Chapter 4: Programming and Documentation Conventions

Introduction 4-1
Truncation Rules 4-1
The Command Tree 4-4
  Command Types 4-4
  Tree Traversal Rules 4-5
  Examples 4-5
Infinity Representation 4-6
Sequential and Overlapped Commands. 4-7
Response Generation 4-7
Notation Conventions and Definitions 4-7
Syntax Diagrams 4-8
Command Structure 4-9
  Common Commands 4-9
  Root Level Commands 4-9
  Subsystem Commands 4-9
Program Examples 4-11
Command Set Organization 4-12

Chapter 5: Common Commands

*CLS 5-4
*ESE 5-5
*ESR 5-7
*IDN 5-9
*IST 5-10
*LRN 5-11
*OPC 5-12
*OPT 5-13
*PRE 5-14
*RCL 5-15
*RST 5-16
*SAV 5-18
*SRE 5-19
*STB 5-21
*TRG 5-23
*TST 5-24
*WAI 5-25

Chapter 6: Root Level Commands

Introduction 6-1
AUToscale 6-4
BEEPer 6-5
BLANk 6-6
DIGitize 6-7
EOI 6-8
ERASe 6-9
LER 6-10
LTER 6-11
MENU 6-12
MERGe 6-13
PRINt 6-14
RUN 6-15
SERial 6-16
STOP 6-17
STORe 6-18
TER 6-19
VIEW 6-20

Chapter 7: System Subsystem

Introduction 7-1
DSP 7-3
ERRor 7-4
HEADer 7-6
KEY 7-7
LONGform 7-9
SETup 7-10

Chapter 8: Acquire Subsystem

Introduction 8-1
(Normal) Persistence Mode 8-1
Averaging Mode 8-2
Envelope Mode 8-2
COMPlete 8-4
COUNt 8-5
POINts 8-6
TYPE 8-7

Chapter 9: Calibrate Subsystem

Introduction 9-1
TNULl 9-2

Chapter 10: Channel Subsystem

Introduction 10-1
COUPling 10-3
ECL 10-4
HFReject 10-5
OFFSet 10-6
PROBe 10-7
RANGe 10-8
TTL 10-9

Chapter 11: Display Subsystem

Introduction 11-1
COLumn 11-4
CONNect 11-5
DATA 11-6
FORMat 11-8
GRATicule 11-9
INVerse 11-10
LINE 11-11
MASK 11-12
PERSistence 11-14

ROW 11-15
SCReen 11-16
SOURce 11-17
STRing 11-18
TEXT 11-19
TMARker 11-20
VMARker 11-21

Chapter 12: Function Subsystem

Introduction 12-1
ADD 12-4
INVert 12-5
MULTiply 12-6
OFFSet 12-7
ONLY 12-8
RANGe 12-9
SUBTract 12-10
VERSus 12-11

Chapter 13: Hardcopy Subsystem

Introduction 13-1
LENGth 13-2
PAGE 13-3

Chapter 14: Measure Subsystem

Introduction 14-1
Measurement Setup 14-1
User-Defined Measurements 14-1
Measurement Error 14-2
Making Measurements 14-2
ALL 14-11
COMPare 14-12
CURSor 14-14
DEFine 14-15
DELay 14-17
DESTination 14-18
DUTycycle 14-19
ESTArt 14-20

ESTOp 14-22
FALLtime 14-24
FREQuency 14-25
LIMittest 14-26
LOWer 14-27
MODE 14-28
NWIDth 14-29
OVERshoot 14-30
PERiod 14-31
POSTfailure 14-32
PRECision 14-33
PREShoot 14-34
PWIDth 14-35
RESults 14-36
RISetime 14-37
SCRatch 14-38
SOURce 14-39
STATistics 14-40
TDELta 14-41
TMAX 14-42
TMIN 14-43
TSTArt 14-44
TSTOp 14-45
TVOLt 14-46
UNITs 14-48
UPPer 14-49
VAMPlitude 14-50
VAVerage 14-51
VBASe 14-52
VDELta 14-53
VFIFty 14-54
VMAX 14-55
VMIN 14-56
VPP 14-57
VRELative 14-58
VRMS 14-60
VSTArt 14-61
VSTOp 14-62
VTIMe 14-63
VTOP 14-64

Chapter 15: Timebase Subsystem

Introduction 15-1
DELay 15-3
MODE 15-4
RANGe 15-5
REFerence 15-6
WINDow 15-7
WINDow:DELay 15-8
WINDow:RANGe 15-9

Chapter 16: Trigger Subsystem

Introduction 16-1
The EDGE Trigger Mode 16-3
The Pattern Trigger Mode 16-4
The State Trigger Mode 16-5
The Delay Trigger Mode 16-6
The TV Trigger Mode 16-7
CONDition 16-13
DELay 16-16
DELay:SLOPe 16-17
DELay:SOURce 16-18
FIELd 16-19
HOLDoff 16-20
LEVel 16-21
LINE 16-22
LOGic 16-23
MODE 16-24
OCCurrence 16-25
OCCurrence:SLOPe 16-26
OCCurrence:SOURce 16-27
PATH 16-28
POLarity 16-29
QUALify 16-30
SLOPe 16-32
SOURce 16-33
STANdard 16-34

Chapter 17: Waveform Subsystem

Introduction 17-1
Data Acquisition Types 17-2
  Normal 17-2
  Average 17-3
  Envelope 17-3
Data Conversion 17-4
  Conversion from Data Value to Voltage 17-4
  Conversion from Data Value to Time 17-4
Data Format for HP-IB Transfer 17-5
  WORD Format 17-5
  BYTE Format 17-6
  COMPRESSED Format 17-6
  ASCII Format 17-6
COUNt 17-9
DATA 17-10
FORMat 17-12
POINts 17-13
PREamble 17-14
SOURce 17-16
TYPE 17-17
XINCrement 17-18
XORigin 17-19
XREFerence 17-20
YINCrement 17-21
YORigin 17-22
YREFerence 17-23

Appendix A: Algorithms

Introduction A-1
Measurement Setup A-1
Making Measurements A-1
Automatic Top-Base A-2
Edge Definition A-2
Algorithm Definitions A-3
delay A-3
+ width A-4
- width A-5
Period A-5
Frequency A-5
Duty Cycle A-5
Risetime A-5
Falltime A-5
Vmax A-5
Vmin A-5
Vp-p A-5
Vtop A-5
Vbase A-6
Vamp A-6
Vavg A-6
Vrms A-6

Appendix B: Example Programs

Introduction B-1
Vertical Channel Setup Program B-2
Timebase Program B-4
Measurement Setup Program B-6
Digitize Program B-9
Hardcopy Program (Service Request using OPC) B-11
Waveform Template Program B-12

Quick Reference Guide

Index

Introduction to Programming an Oscilloscope 1

Introduction

This chapter introduces you to the basic concepts of HP-IB communication and provides information and examples to get you started programming. The exact mnemonics for the commands are listed in chapters 5 through 17.

There are four basic operations that can be done with a controller and an oscilloscope via HP-IB. You can:

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

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

This chapter deals mainly with how to set up the instrument, how to retrieve setup information and measurement results, how to digitize a waveform, and how to pass data to the controller. The chapter is divided into two sections. The first section (page 1-2) concentrates on program syntax, and the second section (page 1-11) discusses programming an oscilloscope. Refer to the chapter "Measure Subsystem" for information on sending measurement data to the instrument.

Note

The programming examples in this manual are written in HP Basic 4.0 for an HP 9000 Series 200/300 Controller.

Programming Syntax

Talking to the Instrument

In general, computers acting as controllers communicate with the instrument by passing messages over a remote interface using the I/O statements provided in the instruction set of the controller's host language. Hence, the HPSL 1.0 messages for programming the HP 54501A, described in this manual, will normally appear as ASCII character strings imbedded inside the I/O statements of your controller's program. For example, the HP 9000 Series 200/300 BASIC and PASCAL language systems use the OUTPUT statement for sending program messages to the HP 54501A, and the ENTER statement for receiving response messages from the HP 54501A.

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 command turns the command headers on:

OUTPUT < device address > ;":SYSTEM:HEADER ON" < terminator >

< device address > represents the address of the device being programmed.

Note

The programming examples in this manual are written in HP Basic 4.0 for an HP 9000 Series 200/300 Controller.

Note

The actual OUTPUT command you use when programming is dependent on the controller and the programming language you are using.

Angular brackets "< >," in this manual, enclose words or characters that symbolize a program code parameter or a bus command.

Information that is displayed in quotes represents the actual message that is sent across the bus. The message terminator (NL or EOI) is the only additional information that is also sent across the bus.

For HP 9000 Series 200/300 controllers, it is not necessary to type in the actual <terminator> at the end of the program message. These controllers automatically terminate the program message internally when the return key is pressed.

Addressing the Instrument

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

Interface Select Code (Selects Interface). Each interface card has a unique interface select code. This code is used by the controller to direct commands and communications to the proper interface. The default is typically "7" for HP-IB controllers.

Instrument Address (Selects Instrument). Each instrument on an HP-IB bus must have a unique instrument address between decimal 0 and 30. The device address passed with the program message must include not only the correct instrument address, but also the correct interface select code.

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

For example, if the instrument address for the HP 54501A is 4 and the interface select code is 7, when the program message is passed, the routine performs its function on the instrument at device address 704.

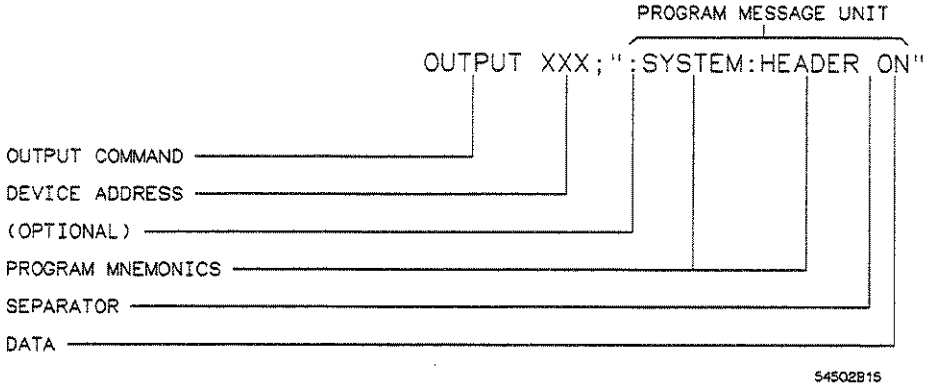
For the HP 54501A, the instrument address is typically set to "7" at the factory. This address can be changed in the HP-IB menu of the Utility menu.

Note

The program examples in this manual assume the HP 54501A is at device address 707.

Program Message Syntax

To program the instrument over the bus, you must have an understanding of the command format and structure expected by the instrument. The instrument is remotely programmed with program messages. These are composed of sequences of program message units, with each unit representing a program command or query. A program command or query is composed of a sequence of functional elements that include separators, headers, program data, and terminators. These are sent to the instrument over the system interface as a sequence of ASCII data messages. For example:



Separator

The <separator> shown in the program message refers to a blank space which is required to separate the program mnemonic from the program data.

Command Syntax

A command is composed of a header, any associated data, and a terminator. The header is the mnemonic or mnemonics that represent the operation to be performed by the instrument. The different types of headers are discussed in the following paragraphs.

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

<program mnemonic> <terminator>

When program data must be included with the simple command header (for example, :DIGITIZE CHAN1), a separator is added. 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 appear between the subsystem mnemonic and the function mnemonic when there are additional levels within the subsystem that must be transversed. The mnemonics within the compound message are separated by colons. For example:

To execute a single function within a subsystem, use the following:

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

(For example :SYSTEM:LONGFORM ON)

To transverse 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 CHAN1)

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)

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 timebase to 1 second full scale.

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

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.

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). The query :TIMEBASE:RANGE? places the current timebase setting in the output queue. 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 risetime of your waveform and place the result in the output queue.

Note

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 the program statement ENTER Value_risetime to read the result of the query and place the result in a variable (Value_risetime).

Sending another command before reading the result of the query will cause the output buffer to be cleared and the current response to be lost. This will also generate 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.

Both program command and query headers may be sent in either longform (complete spelling), shortform (abbreviated spelling), or any combination of longform and shortform. Either of the following examples turn the headers on.

:SYSTEM:HEADER ON - longform

:SYST:HEAD ON - shortform

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

Note

The rules for shortform syntax are shown in the chapter "Programming and Documentation Conventions."

Program Data

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 data parameters a comma separates sequential program data.

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

For example, :TRIGGER:DELAY TIME,1.23E-01 has two data parameters: TIME and 1.23E-01.

Character Program Data. Character program data is used to convey parameter information as alpha or alphanumeric strings. For example, the timebase command MODE 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 timebase mode to single.

Numeric Program Data. Some command headers require program data to be a number. For example, :TIMEBASE:RANGE requires the desired full scale range to be expressed numerically. The instrument recognizes integers, real numbers, and scientific notation. For more information see the chapter "Message Communication and System Functions."

Program Message Terminator

The program codes 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. All three ways are equivalent with the exact encodings for the program terminators listed in the chapter "Message Communication and System Functions." 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).

Note

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

Selecting Multiple Subsystems

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

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

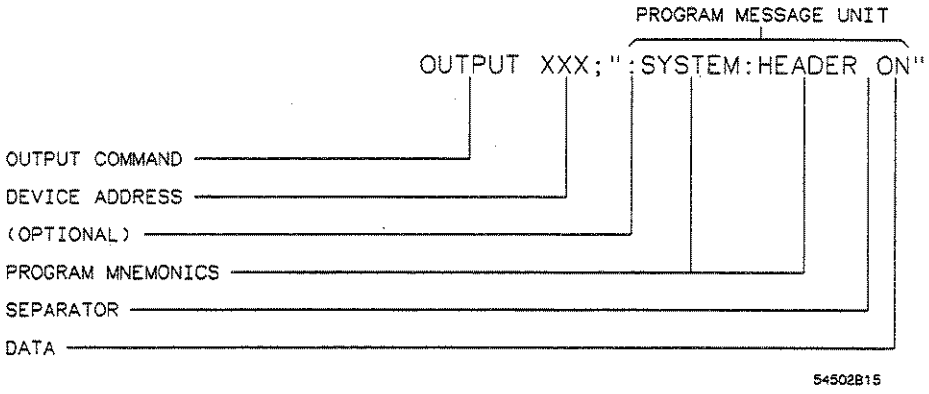
:CHANNEL1:RANGE 0.4;:TIMEBASE RANGE 1

Note

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

Summary

The following illustration summarizes the syntax for programming over the bus.



Programming an Oscilloscope

Initialization

To make sure the bus and all appropriate interfaces are in a known state, begin every program with an initialization statement. For example:

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

Then initialize the instrument to a preset state. For example:

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

Note

The actual commands and syntax for initializing the instrument are discussed in the chapter "Common Commands."

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

Autoscale

The AUTOSCALE feature of Hewlett-Packard digitizing oscilloscopes performs a very useful function on unknown waveforms by setting up the vertical channel, timebase, 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 1 ms full-scale with 20 ns delay. The timebase mode is set to triggered, and the trigger circuit is programmed to trigger at 0.25 volts on a positive slope.

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 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$.

Note

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 the program statement ENTER Risetime$ to read the result of the query and place the result in a variable (Risetime$).

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

Executing an ENTER statement before sending a query will cause the controller to wait indefinitely.

The actual ENTER program statement you use when programming is dependent on the programming language you are using.

The format specification for handling the 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 longform is OFF , the header will be in its shortform and the header will vary in length depending on the particular query. The following would be returned from 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**)

Note

A command or query may be sent in either longform or shortform, or in any combination of longform and shortform. The HEADER and LONGFORM commands only control the format of the returned data and have no effect on the way commands are sent. Common commands never return a header.

Refer to the chapter "System Subsystem" for information on turning the HEADER and LONGFORM commands on and off.

Response Data Formats

Most data will be returned as exponential or integer numbers. However, query data of instrument setups may be returned as character data. Interrogating the trigger SLOPE? will return one of the following:

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

Note

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 observe 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$.

Note

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.

Note

For the example programs, assume that the device being programmed is at device address 707. The actual address will vary 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 off:

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

After running this program, the controller displays:

```
+1.00000E-1
```

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.

Note

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

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

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

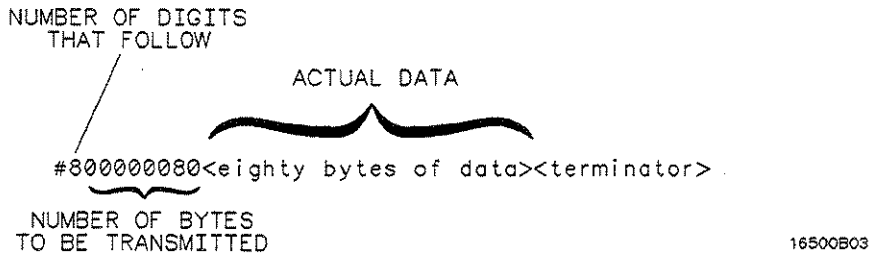
After running this program, the controller displays:

```
.1
```

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:



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 :TIMEBASE:RANGE?;DELAY? with HEADER and LONGFORM **on** would be:

```
:TIMEBASE:RANGE <range_setting>;:TIMEBASE:DELAY <delay_setting>
```

If you do not need to see the headers when the numeric values are returned, then you could use following program message to read the query :TIMEBASE:RANGE?;DELAY? into multiple numeric variables:

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

Note

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

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 chapter "Message Communication and System Functions" explains how to check the status of the instrument.

Digitize Command

The ACQUIRE and WAVEFORM subsystems are subsystems that affect the DIGITIZE command. The DIGITIZE command is used to capture a waveform in a known format which is specified by the ACQUIRE subsystem. When the DIGITIZE command is sent to an instrument, the specified channel signal is digitized with the current ACQUIRE parameters. To obtain waveform data, you must specify the WAVEFORM parameters for the waveform data prior to sending the :WAVEFORM:DATA? query.

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 acquistion, and number of averages used by the DIGITIZE command. This allows you to specify exactly what the digitized information will contain. A typical setup is:

```
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 average mode with four averages and defines the data record to be 500 points. This means that when the DIGITIZE command is received, the waveform will not be stored 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.**

The easiest method of entering a digitized waveform from the instrument is to use the ASCII format and place the information in an integer array. The data point is represented by signed six-digit integers whose values range from 0 to 32,640. You must scale the integers to determine the voltage value of each point. These integers are passed starting with the leftmost point on the instrument's display. For more information, refer to the chapter "Waveform Subsystem."

Interface Functions 2

Introduction

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

Interface Capabilities

The interface capabilities of the HP 54501A, as defined by IEEE 488.1 are SH1, AH1, T5, L4, SR1, RL1, PP1, DC1, DT1, C0, and E2.

Command and Data Concepts

The HP-IB has two modes of operation: command mode and data mode. The bus is in command mode when the ATN line is true. The command mode is used to send talk and listen addresses and various bus commands, such as a group execute trigger (GET). The bus is in the data mode when the ATN line is false. The data mode is used to convey device-dependent messages across the bus. The device-dependent messages include all of the instrument command and responses found in chapters 5 through 17 of this manual.

Addressing

By using the front-panel controls, the instrument can be placed in either talk-only mode or addressed (talk/listen) mode (see your front-panel reference). Talk-only mode should be used when you want the instrument to talk directly to a printer without the aid of a controller. Addressed mode is used when the instrument will operate in conjunction with a controller. When the instrument is in the addressed mode, the following is true:

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

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

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

Remote, Local and Local Lockout

The local, remote, and remote with local lockout modes may be used for various degrees of front-panel control while a program is running. The instrument will accept and execute bus commands while in local mode, and the front panel will also be entirely active. If the HP 54501A is in remote mode, all controls (except the power switch and the LOCAL key) are entirely locked out. Local control can only be restored by the controller or pressing the front-panel LOCAL key.

Note

Cycling the power will also restore local control, but this will also reset certain HP-IB states.

The instrument is placed in remote mode by setting the REN bus control line true, and then addressing the instrument to listen. The instrument can be placed in local lockout mode by sending the local lockout command (LLO). The instrument can be returned to local mode by either setting the REN line false, or sending the instrument the go-to-local command (GTL), or simulating a front-panel LOCAL key press using the SYSTEM:KEY command.

Bus Commands

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

Device Clear

The device clear (DCL) or selected device clear (SDC) commands clear the input and output buffers, reset the parser, and clear any pending commands.

Group Execute Trigger (GET)

The group execute trigger (GET) command arms the trigger which is the same action produced by sending the RUN command.

Interface Clear (IFC)

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

Status Annunciators

The HP 54501A will display the HP-IB status on the CRT. The message will indicate whether the instrument is in the remote mode, whether talk or listen is addressed, and whether the instrument has requested service. When the instrument is in the local mode only the SRQ annunciator may be displayed.

Message Communication and System Functions

3

This chapter describes the operation of instruments that operate in compliance with the IEEE 488.2 standard. Instruments that are compatible with IEEE 488.2 must also be compatible with IEEE 488.1, however IEEE 488.1 compatible instruments may or may not conform to the IEEE 488.2 standard. The IEEE 488.2 standard defines the message exchange protocols by which the instrument and the controller will communicate. It also defines some common capabilities, which are found in all IEEE 488.2 instruments. This chapter also contains a few items which are not specifically defined by IEEE 488.2, but deal with message communication or system functions.

Protocols

The protocols of IEEE 488.2 define the overall scheme used by the controller and the instrument to communicate. This includes defining when it is appropriate for devices to talk or listen, and what happens when the protocol is not followed.

Functional Elements

Before proceeding with the description of the protocol, a few system components should be understood.

Input Buffer. The input buffer of the instrument is the memory area where commands and queries are stored prior to being parsed and executed. It allows a controller to send a string of commands to the instrument which could take some time to execute, and then proceed to talk to another instrument while the first is parsing and executing commands. The HP 54501A's input buffer will hold 300 characters, or bytes of data.

Output Queue. The output queue of the instrument is the memory area where all output data (<response messages>) are stored until read by the controller. The HP 54501A's output queue will hold 300 characters, however the instrument will handle block data of greater than 300 characters where appropriate.

Parser. The instrument's parser is the component that interprets the commands sent to the instrument and decides what actions should be taken. "Parsing" refers to the action taken by the parser to achieve this goal. Parsing and executing of commands begins when either the instrument sees a <program message terminator> (defined later in this chapter) or the input buffer becomes full. If you wish to send a long sequence of commands to be executed and then talk to another instrument while they are executing, you should send all the commands before sending the <program message terminator>.

Protocol Overview

The instrument and controller communicate using <program message>s and <response message>s. These messages serve as the containers into which sets of program commands or instrument responses are placed. <program message>s are sent by the controller to the instrument, and <response message>s are sent from the instrument to the controller in response to a query message. A "query message" is defined as being a <program message> which contains one or more queries. The instrument will only talk when it has received a valid query message, and therefore has something to say. The controller should only attempt to read a response after sending a complete query message, but before sending another <program message>. The basic rule to remember is that the instrument will only talk when prompted to, and it then expects to talk before being told to do something else.

Protocol Operation

When the instrument is turned on or when it receives a device clear command, the input buffer and output queue are cleared, and the parser is reset to the root level of the command tree.

The instrument and the controller communicate by exchanging complete <program message>s and <response message>s. This means that the controller should always terminate a <program message> before attempting to read a response. The instrument will terminate <response message>s except during a hardcopy output.

If a query message is sent, the next message passing over the bus should be the <response message>. The controller should always read the complete <response message> associated with a query message before sending another <program message> to the same instrument.

The instrument allows the controller to send multiple queries in one query message. This is referred to as sending a "compound query." As will be noted later in this chapter, multiple queries in a query message are separated by semicolons. The responses to each of the queries in a compound query will also be separated by semicolons.

Commands are executed in the order they are received. This also applies to the reception of the group execute trigger (GET) bus command. The group execute trigger command should not be sent in the middle of a <program message>.

Protocol Exceptions

If an error occurs during the information exchange, the exchange may not be completed in a normal manner. Some of the protocol exceptions are shown below.

Addressed to talk with nothing to say. If the instrument is addressed to talk before it receives a query, it will indicate a query error and will not send any bytes over the bus. If the instrument has nothing to say because queries requested were unable to be executed because of some error, the device will not indicate a query error, but will simply wait to receive the next message from the controller.

Addressed to talk with no listeners on the bus. If the instrument is addressed to talk and there are no listeners on the bus, the instrument will wait for a listener to listen, or for the controller to take control.

Command Error. A command error will be reported if the instrument detects a syntax error or an unrecognized command header.

Execution Error. An execution error will be reported if a parameter is found to be out of range, or if the current settings do not allow execution of a requested command or query.

Device-specific Error. A device-specific error will be reported if the instrument is unable to execute a command for a strictly device dependent reason.

Query Error. A query error will be reported if the proper protocol for reading a query is not followed. This includes the interrupted and unterminated conditions described below.

Unterminated Condition. If the controller attempts to read a <response message> before terminating the <program message>, a query error will be generated. The parser will reset itself, and the response will be cleared from the output queue of the instrument without being sent over the bus.

Interrupted Condition. If the controller does not read the entire <response message> generated by a query message and then attempts to send another <program message>, the device will generate a query error. The unread portion of the response will then be discarded by the instrument. The interrupting <program message> will not be affected.

Buffer Deadlock. The instrument may become deadlocked if the input buffer and output queue both become full. This condition can occur if a very long <program message> is sent containing queries that generate a great deal of response data. The instrument cannot accept any more bytes, and the controller cannot read any of the response data until it has completed sending the entire <program message>. Under this condition the instrument will break the deadlock by clearing the output queue, and continuing to discard responses until it comes to the end of the current <program message>. The query error bit will also be set.

Syntax Diagrams

The syntax diagrams in this chapter are similar to the syntax diagrams in the IEEE 488.2 specification. Commands and queries are sent to the instrument as a sequence of data bytes. The allowable byte sequence for each functional element is defined by the syntax diagram that is shown with the element description.

The allowable byte sequence can be determined by following a path in the syntax diagram. The proper path through the syntax diagram is any path that follows the direction of the arrows. If there is a path around an element, that element is optional. If there is a path from right to left around one or more elements, that element or those elements may be repeated as many times as desired.

Syntax Overview

This overview is intended to give a quick glance at the syntax defined by IEEE 488.2. It should allow you to understand many of the things about the syntax you need to know. This chapter also contains the details of the IEEE 488.2 defined syntax.

IEEE 488.2 defines the blocks used to build messages which are sent to the instrument. A whole string of commands can therefore be broken up into individual components.

Figure 3-1 shows a breakdown of an example <program message>. There are a few key items to notice:

1. A semicolon separates commands from one another. Each <program message unit> serves as a container for one command. The <program message unit>s are separated by a semicolon.

2. A <program message> is terminated by a <NL> (new line), a <NL> with EOI asserted, or EOI being asserted on the last byte of the message. The recognition of the <program message terminator>, or <PMT>, by the parser serves as a signal for the parser to begin execution of commands. The <PMT> also affects command tree traversal (see the Programming and Documentation Conventions chapter).

3. Multiple data parameters are separated by a comma.

4. The first data parameter is separated from the header with one or more spaces.

5. The header MEAS:SOURCE is a compound header. It places the parser in the measure subsystem until the <NL> is encountered.

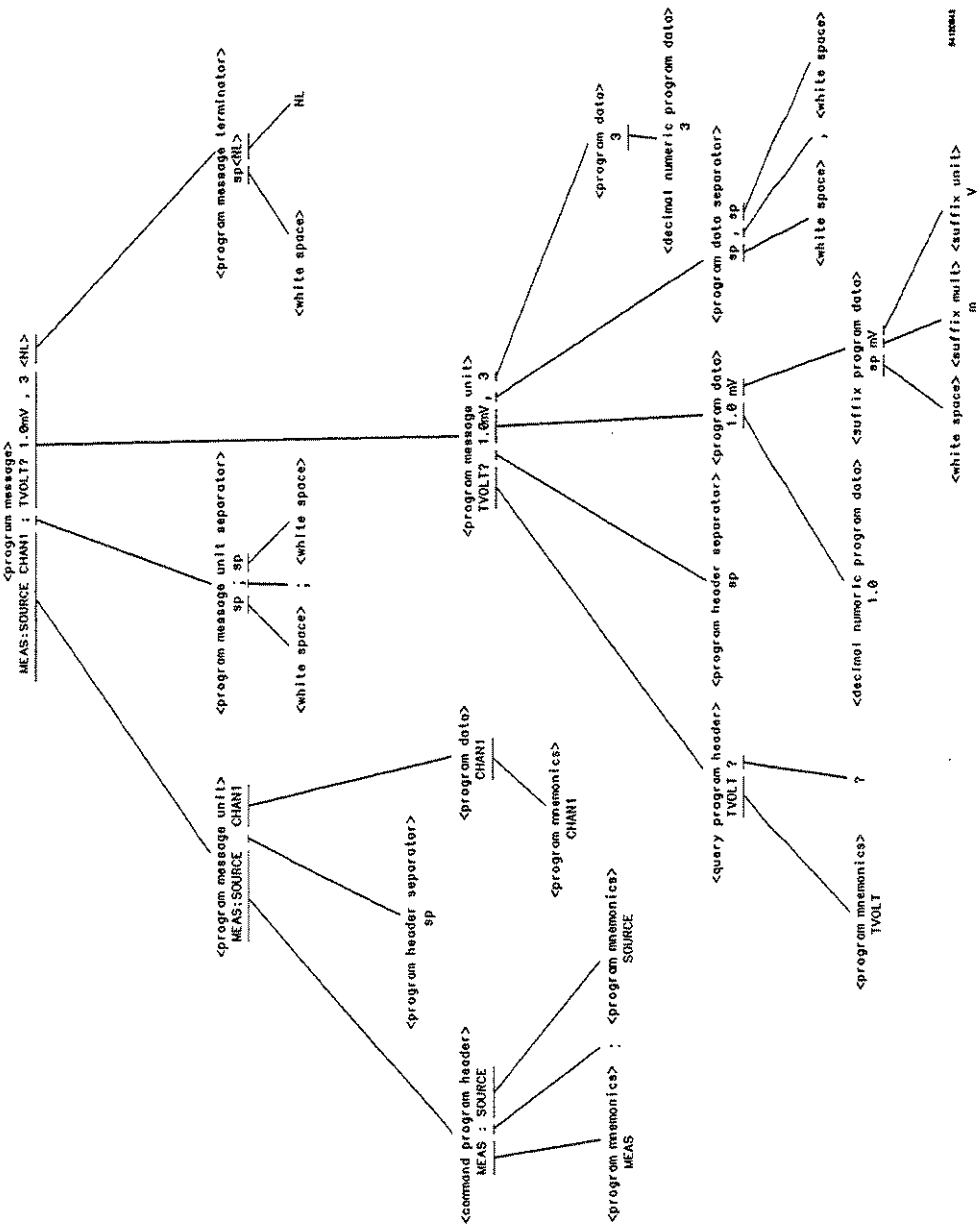


Figure 3-1. <program message> Parse Tree

Device Listening Syntax

The listening syntax of IEEE 488.2 is designed to be more forgiving than the talking syntax. This allows greater flexibility in writing programs, as well as allowing them to be easier to read.

Upper/Lower Case Equivalence. Upper and lower case letters are equivalent. The mnemonic **RANGE** has the same semantic meaning as the mnemonic **range**.

<white space>. <white space> is defined to be one or more characters from the ASCII set of 0 - 32 decimal, excluding 10 decimal (NL). <white space> is used by several instrument listening components of the syntax.

It is usually optional, and can be used to increase the readability of a program.

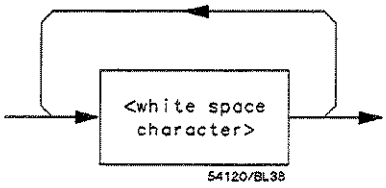


Figure 3-2. <white space>

<program message>. The <program message> is a complete message to be sent to the instrument. The instrument will begin executing commands once it has a complete <program message>, or when the input buffer becomes full. The parser is also repositioned to the root of the command tree after executing a complete <program message>. Refer to the Tree Traversal Rules in the Programming and Documentation Conventions chapter for more details.

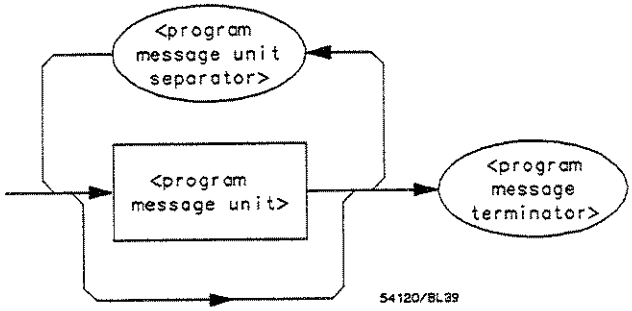


Figure 3-3. <program message>

<program message unit>. The <program message unit> is the container for individual commands within a <program message>.

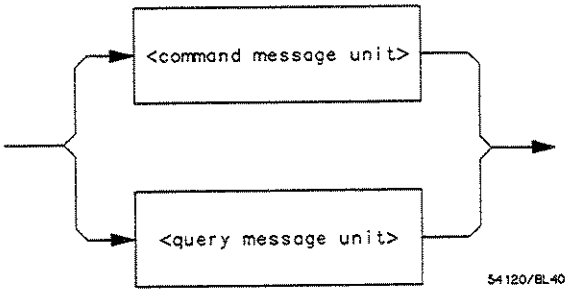


Figure 3-4. <program message unit>

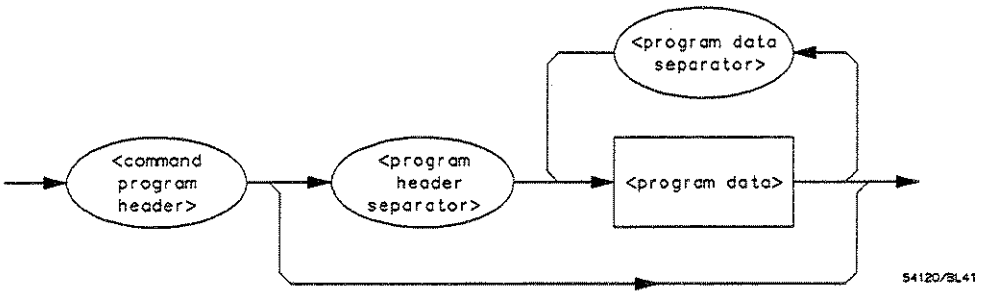


Figure 3-5. <command message unit>

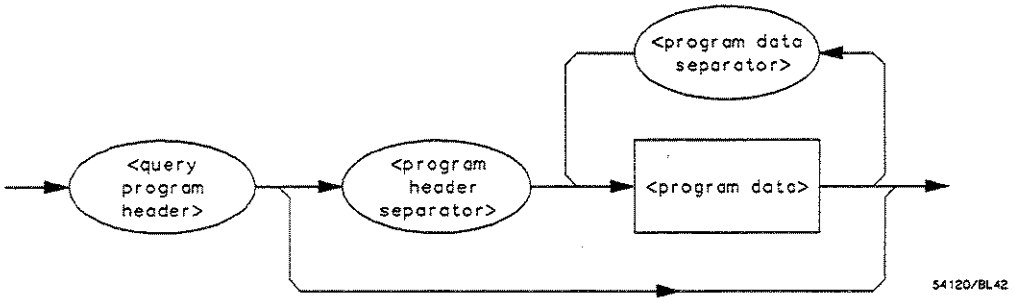


Figure 3-6. <query message unit>

<program message unit separator>. A semicolon separates <program message unit>s, or individual commands.

<command program header>/<query program header>. These elements serve as the headers of commands or queries. They represent the action to be taken.

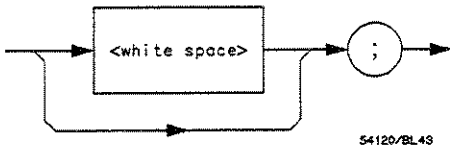


Figure 3-7. <program message unit separator>

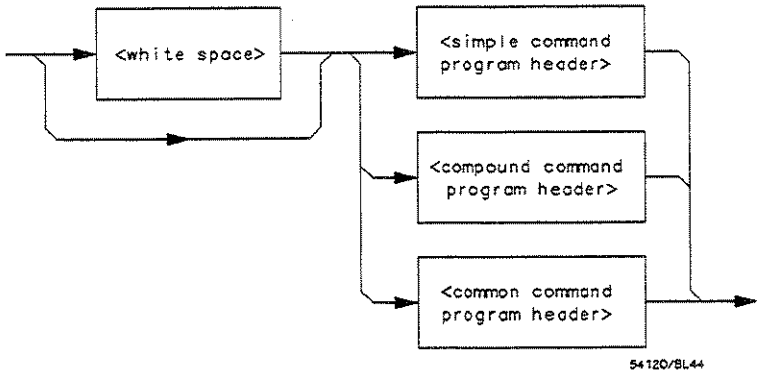
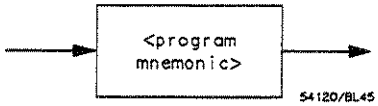
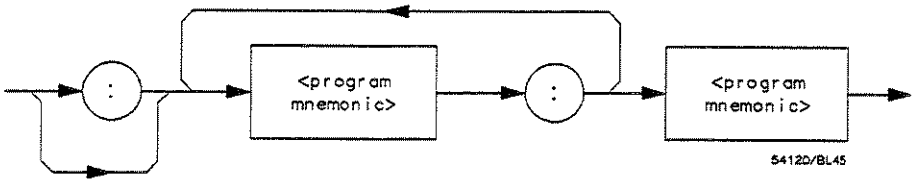


Figure 3-8. <command program header>

Where <simple command program header> is defined as

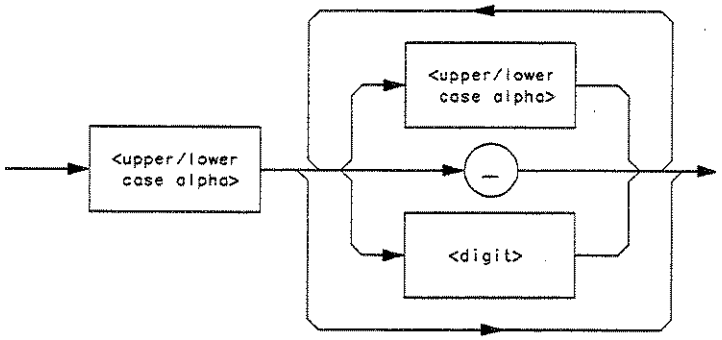


Where <compound command program header> is defined as



Where <common command program header> is defined as

Where <program mnemonic> is defined as

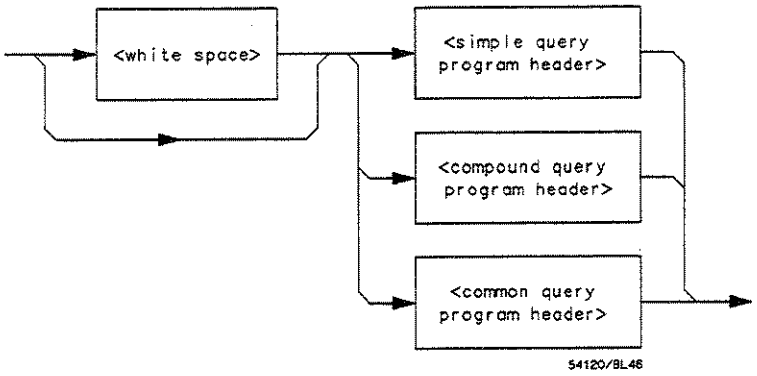


Where <upper/lower case alpha> is defined as a single ASCII encoded byte in the range 41 - 5A, 61 - 7A (65 - 90, 97 - 122 decimal).

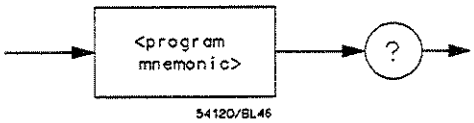
Where <digit> is defined as a single ASCII encoded byte in the range 30 - 39 (48 - 57 decimal).

Where (_) represents an "underscore", a single ASCII-encoded byte with the value 5F (95 decimal).

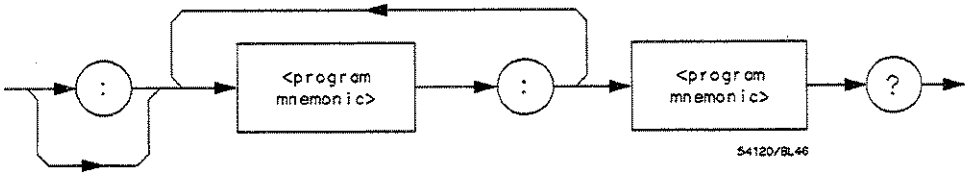
Figure 3-8. <command program header> (continued)



Where <simple query program header> is defined as



Where <compound query program header> is defined as



Where <common query program header> is defined as

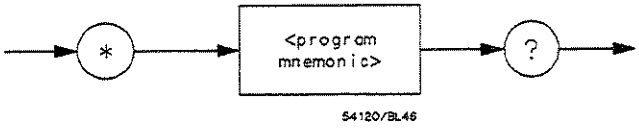


Figure 3-9. <query program header>

<program data>. The <program data> element represents the possible types of data which may be sent to the instrument. The HP 54501A will accept the following data types: <character program data>, <decimal numeric program data>, <suffix program data>, <string program data>, and <arbitrary block program data>.

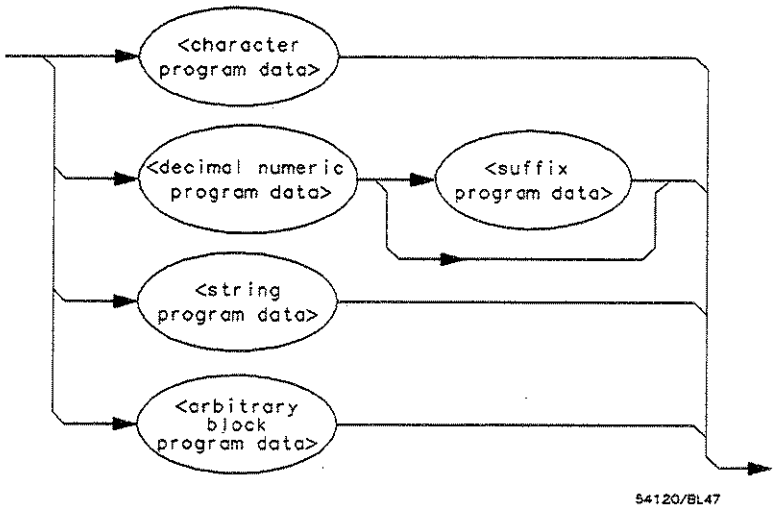


Figure 3-10. <program data>

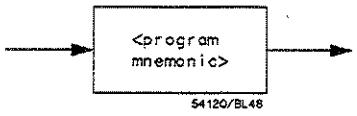
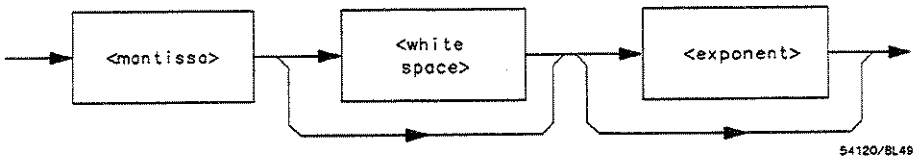
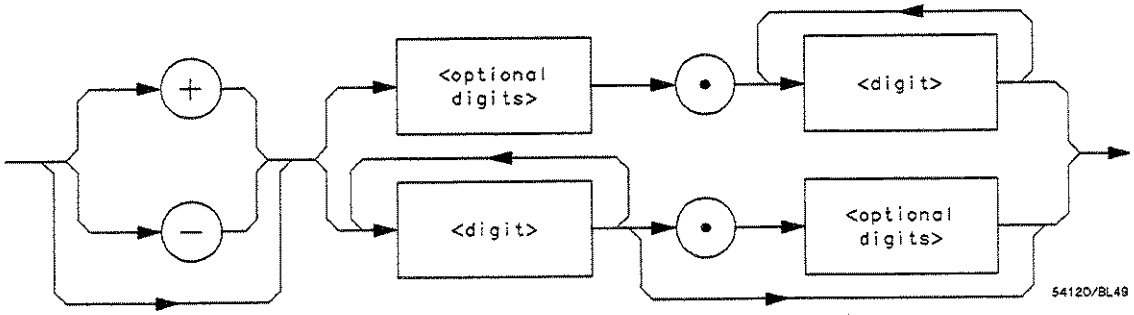


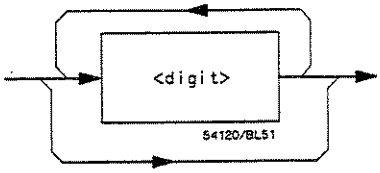
Figure 3-11. <character program data>



Where <mantissa> is defined as



Where <optional digits> is defined as



Where <exponent> is defined as

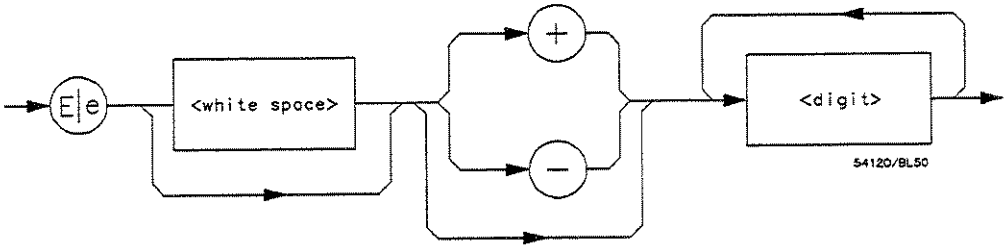


Figure 3-12. <decimal numeric program data>

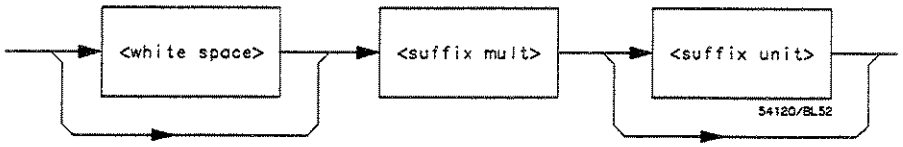


Figure 3-13. <suffix program data>

Suffix Multiplier. The suffix multipliers that the instrument will accept are shown in table 3-1.

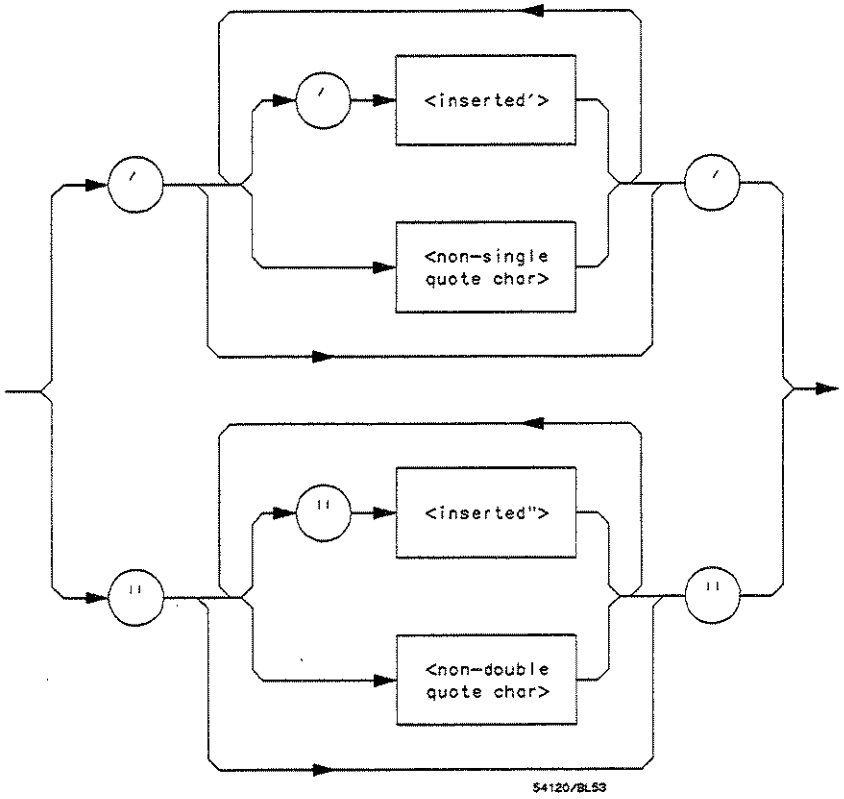
Table 3-1. <suffix mult>

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

Suffix Unit. The suffix units that the instrument will accept are shown in table 3-2.

Table 3-2. <suffix unit>

Suffix	Referenced Unit
V	Volt
S	Second



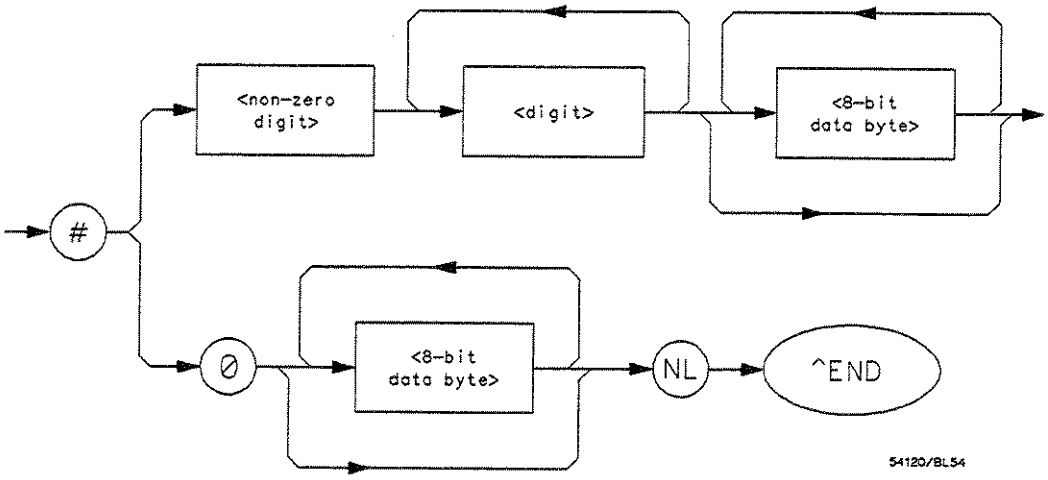
Where <inserted'> is defined as a single ASCII character with the value 27 (39 decimal).

Where <non-single quote char> is defined as a single ASCII character of any value except 27 (39 decimal).

Where <inserted"> is defined as a single ASCII character with the value 22 (34 decimal).

Where <non-double quote char> is defined as a single ASCII character of any value except 22 (34 decimal).

Figure 3-14. <string program data>



Where <non-zero digit> is defined as a single ASCII encoded byte in the range 31 - 39 (49 - 57 decimal).

Where <8-bit byte> is defined as an 8-bit byte in the range 00 -ff (0 - 255 decimal).

Figure 3-15. <arbitrary block program data>

<program data separator>. A comma separates multiple data parameters of a command from one another.

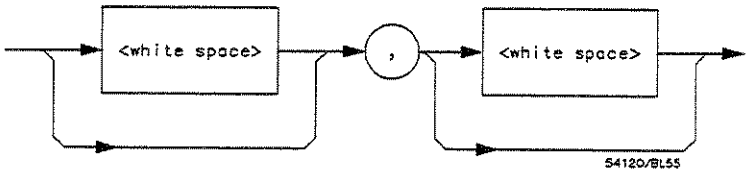


Figure 3-16. <program data separator>

< program header separator >. A space (ASCII decimal 32) separates the header from the first or only parameter of the command.

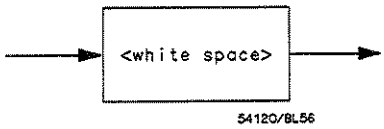
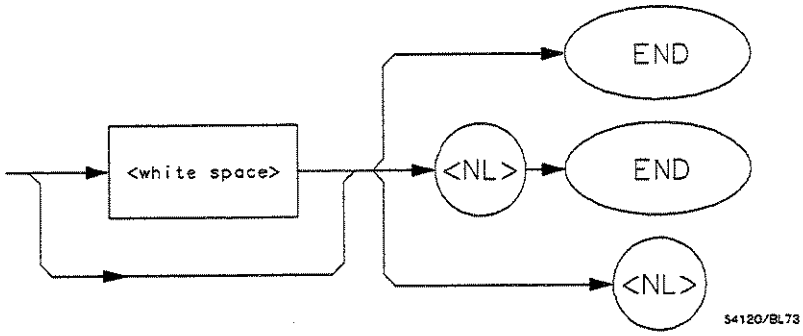


Figure 3-17. <program header separator>

< program message terminator >. The < program message terminator > or < PMT > serves as the terminator to a complete < program message >. When the parser sees a complete < program message > it will begin execution of the commands within that message. The < PMT > also resets the parser to the root of the command tree.



While < NL > is defined as a single ASCII-encoded byte 0A (10 decimal).

Figure 3-18. <program message terminator>

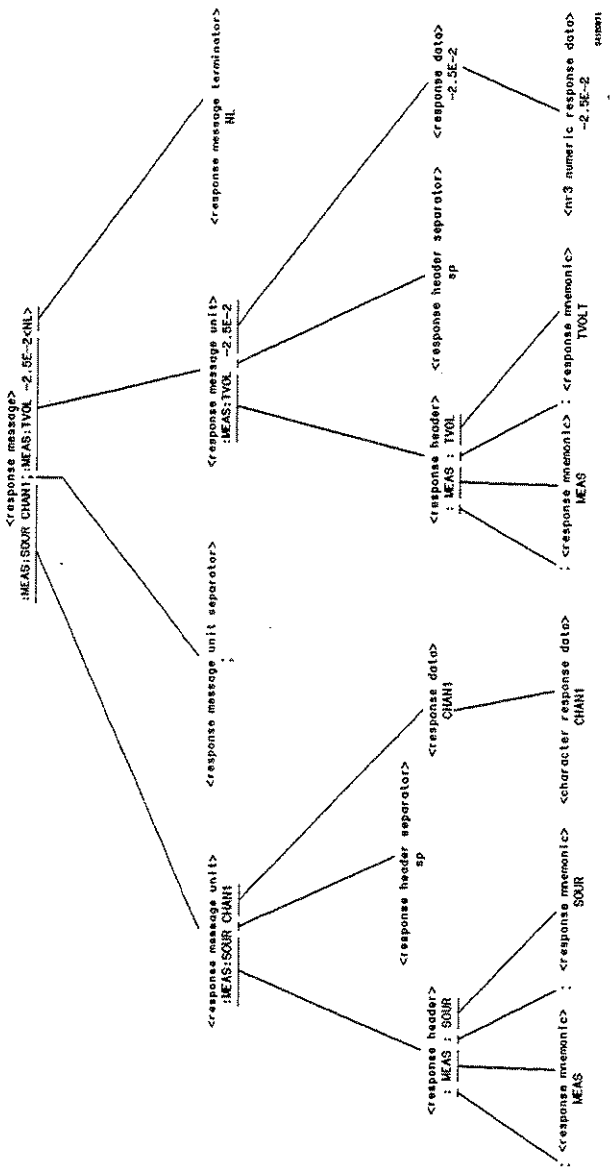


Figure 3-19. <response message> Tree

Device Talking Syntax

The talking syntax of IEEE 488.2 is designed to be more precise than the listening syntax. This allows the programmer to write routines which can more easily interpret and use the data the instrument is sending. One of the implications of this is the absence of < white space > in the talking formats. The instrument will not pad messages which are being sent to the controller with spaces.

< response message >. This element serves as a complete response from the instrument. It is the result of the instrument executing and buffering the results from a complete < program message >. The complete < response message > should be read before sending another < program message > to the instrument.

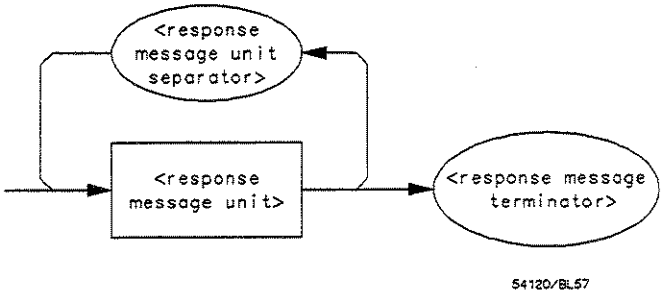
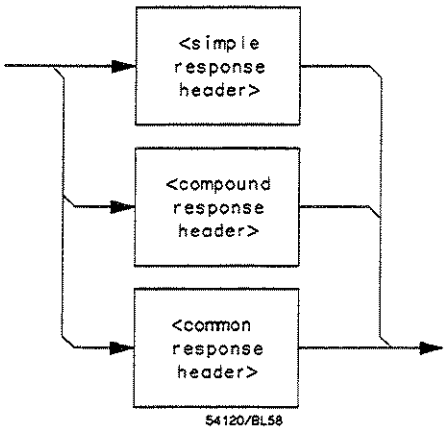


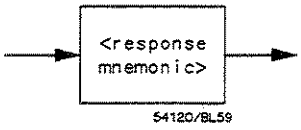
Figure 3-20. < response message >

< response message unit >. This element serves as the container of individual pieces of a response. Typically a < query message unit > will generate one < response message unit >, although a < query message unit > may generate multiple < response message unit > s.

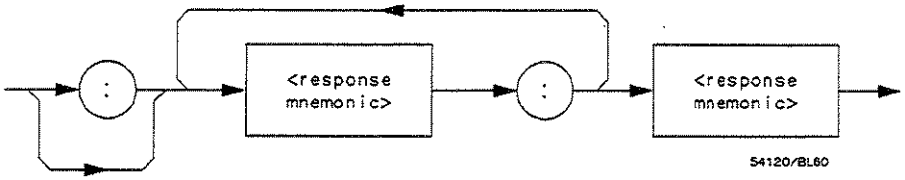
< response header >. The < response header >, when returned, indicates what the response data represents.



Where <simple response mnenomic> is defined as



Where <compound response header> is defined as



Where <common response header> is defined as

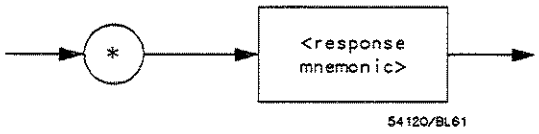
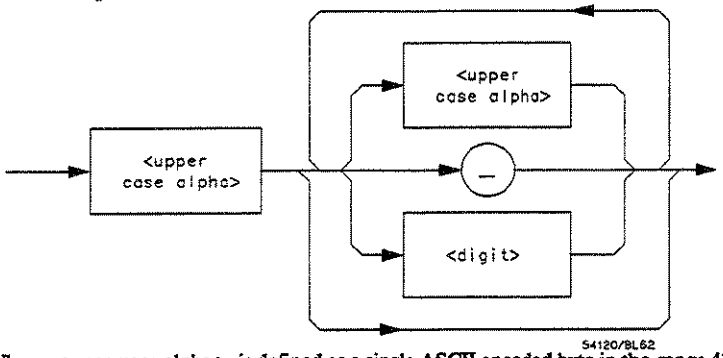


Figure 3-21. <response message unit>

Where <response mnemonic> is defined as



Where <uppercase alpha> is defined as a single ASCII encoded byte in the range 41 - 5A (65 - 90 decimal).

Where (_) represents an "underscore", a single ASCII encoded byte with the value 5F (95 decimal).

Figure 3 - 21. <response message unit> (continued)

<response data>. The <response data> element represents the various types of data which the instrument may return. These types include: <character response data>, <nr1 numeric response data> (integer), <nr3 numeric response data> (exponential), <string response data>, <definite length arbitrary block response data>, and <arbitrary ASCII response data>.

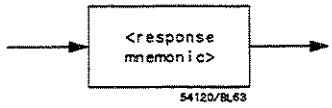


Figure 3-22. <character response data>

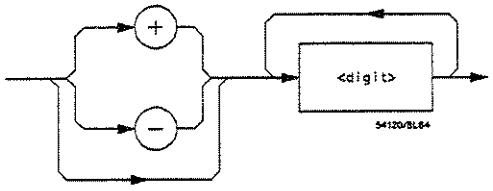


Figure 3-23. <nr1 numeric response data>

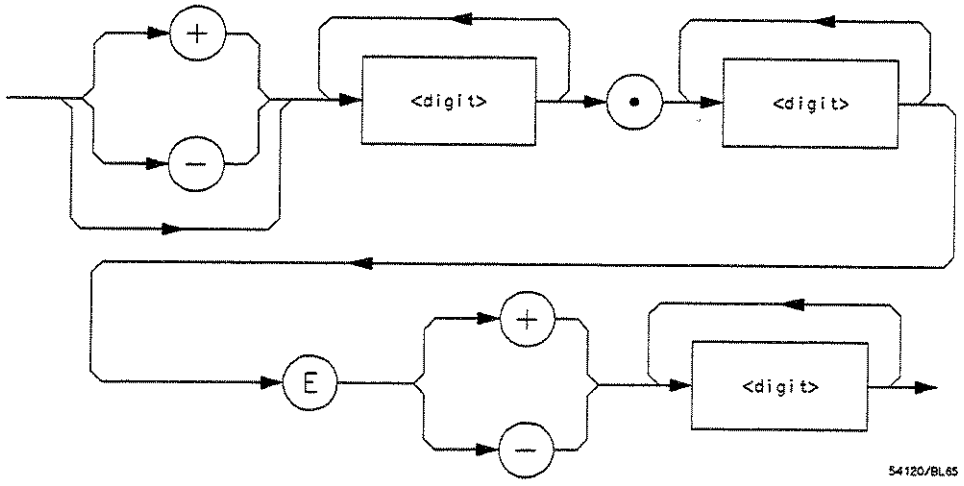


Figure 3-24. <nr3 numeric response data>

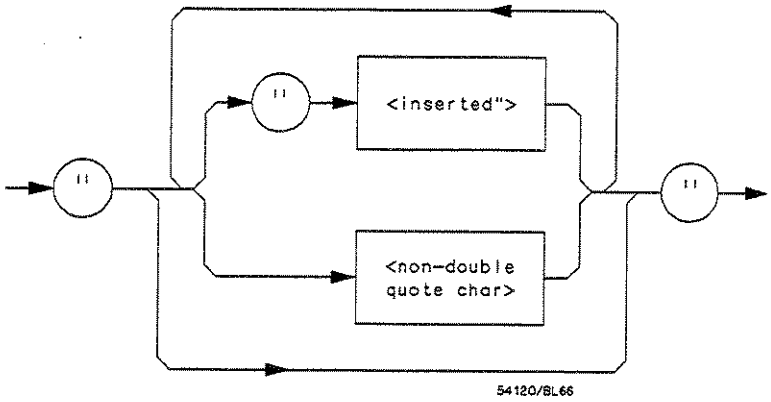


Figure 3-25. <string response data>

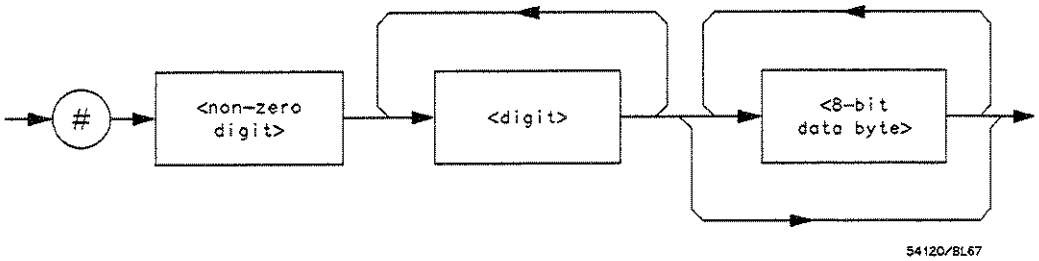
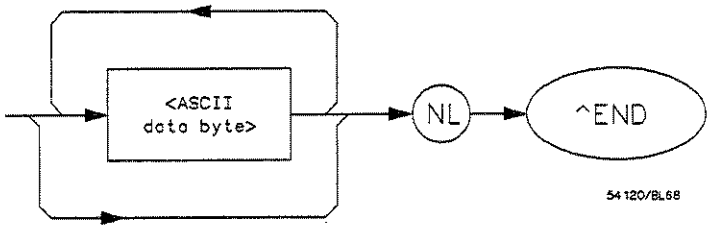


Figure 3-26. <definite length arbitrary block>



Where <ASCII data type> represents any ASCII-encoded byte except <NL> (0A, 10 decimal).

1. The END message provides an umambiguous termination to an element that contains arbitrary ASCII characters.

2. The IEEE 488.1 END message serves the dual function for terminating this element as well as terminating the <RESPONSE MESSAGE>. It is only sent once with the last byte of the indefinite block data. The NL is presented for consistency with the <RESPONSE MESSAGE TERMINATOR>.

Figure 3-27. <arbitrary ASCII response data>

<response data separator>. A comma separates multiple pieces of response data within a single <response message unit>.

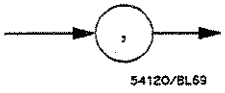


Figure 3-28. <response data separator>

<response header separator>. A space (ASCII decimal 32) delimits the response header, if returned, from the first or only piece of data.

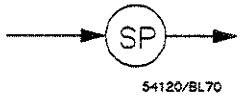


Figure 3-29. <response header separator>

<response message unit separator>. A semicolon delimits the <response message unit>s if multiple responses are returned.

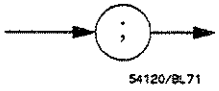


Figure 3-30. <response message unit separator>

<response message terminator>. A <response message terminator> (NL) terminates a complete <response message>. It should be read from the instrument along with the response itself.

Note

If you do not read the <response message terminator> the HP 54501A will produce an interrupted error on the next message.

Common Commands

IEEE 488.2 defines a set of common commands. These commands perform functions which are common to any type of instrument. They can therefore be implemented in a standard way across a wide variety of instrumentation. All the common commands of IEEE 488.2 begin with an asterisk. There is one key difference between the IEEE 488.2 common commands and the rest of the commands found in this instrument. The IEEE 488.2 common commands do not affect the parser's position within the command tree. More information about the command tree and tree traversal can be found in the Programming and Documentation Conventions chapter.

Table 3-3. HP 54501A's IEEE 488.2 Common Commands

Command	Command Name
*CLS	Clear Status Command
*ESE	Event Status Enable Command
*ESE?	Event Status Enable Query
*ESR?	Event Status Register Query
*IDN?	Identification Query
*IST?	Individual Status Query
*LRN?	Learn Device Setup Query
*OPC	Operation Complete Command
*OPC?	Operation Complete Query
*OPT?	Option Identification Query
*PRE	Parallel Poll Enable Register Enable Command
*PRE?	Parallel Poll Enable Register Enable Query
*RCL	Recall Command
*RST	Reset Command
*SAV	Save Command
*SRE	Service Request Enable Command
*SRE?	Service Request Enable Query
*STB?	Read Status Byte Query
*TRG	Trigger Command
*TST?	Self-Test Query
*WAI	Wait-to-Continue Command

Status Reporting

The status reporting features which are available over the HP-IB include the serial and parallel polls. IEEE 488.2 defines data structures, commands, and common bit definitions for each. 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 via 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 will also be cleared.

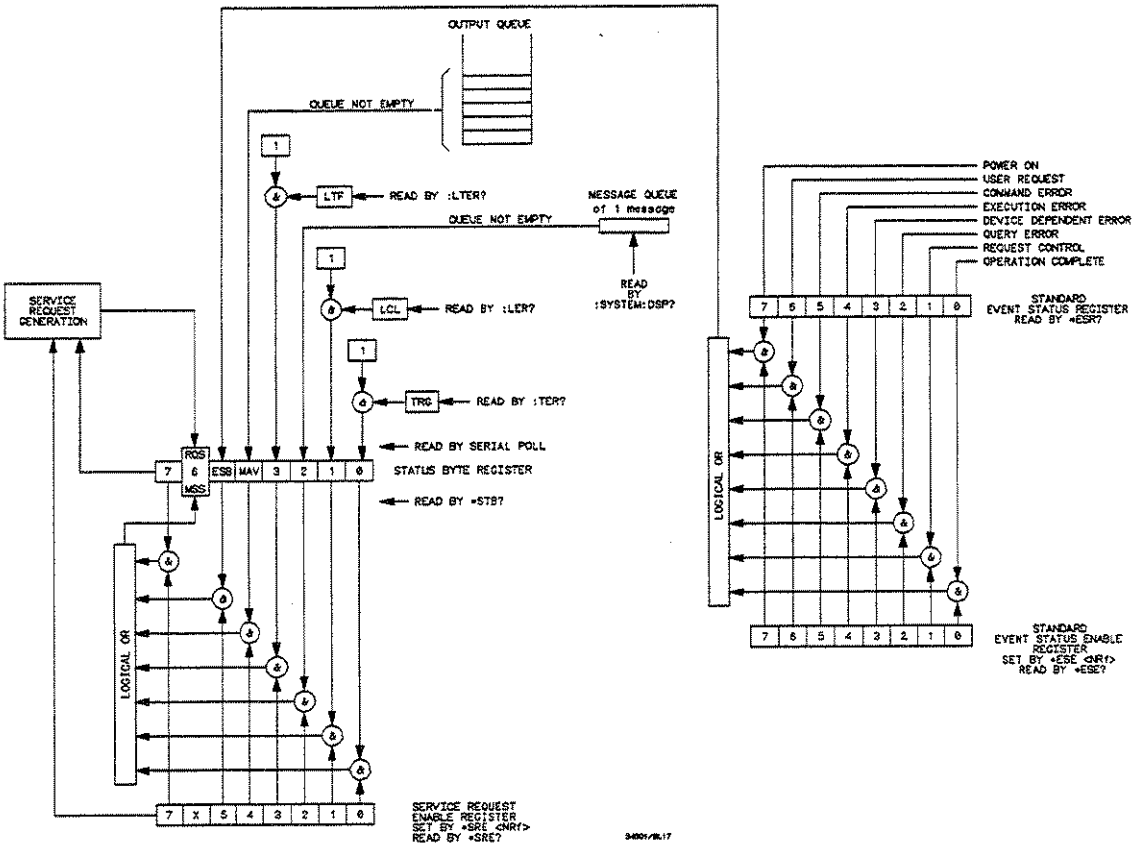


Figure 3-31. Status Reporting Data Structures

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 will be set to 0 after being 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 54501A.

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 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 54501A will never request 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 failure has occurred.

Key Features

A few of the most important features of Status Reporting are shown below.

Operation Complete. 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, will set the OPC bit in the Standard Event Status Register. If the OPC bit and the RQS bit have been enabled, a service request will be generated.

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

The Trigger Bit. The 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?) will not be cleared by reading it. The status byte is not cleared when read, except for the RQS bit.

Serial Poll

The HP 54501A 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 will show 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, and it is also assumed that we are operating on Interface Select Code 7.

The program command for serial poll using HP BASIC 4.0 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 HP-IB 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 instruments 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 HP 54501A Status Byte Register will be reset if it was set. Once a bit in the Status Byte Register is set, it will remain 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.

Parallel Poll Parallel poll is a controller initiated operation which is used to obtain information from several devices simultaneously. When a controller initiates a Parallel Poll, each device returns a Status Bit via one of the DIO data lines. Device DIO assignments are made by the controller using the PPC (Parallel Poll Configure) sequence. Devices respond either individually, each on a separate DIO line; collectively on a single DIO line; or any combination of these two ways. When responding collectively, the result is a logical AND (True High) or a logical OR (True Low) of the groups of the status bits.

Figure 3-32 shows the Parallel Poll Data Structure. The summary bit is sent in response to a Parallel Poll. This summary bit is the "ist" (Individual Status) local message.

The Parallel Poll Enable Register determines which events are summarized in the individual status local register. The *PRE (Parallel Poll Register Enable) command is used to write to the enable register and the *PRE? query is used to read the register. The *IST? query can be used to read the "ist" without doing a parallel poll.

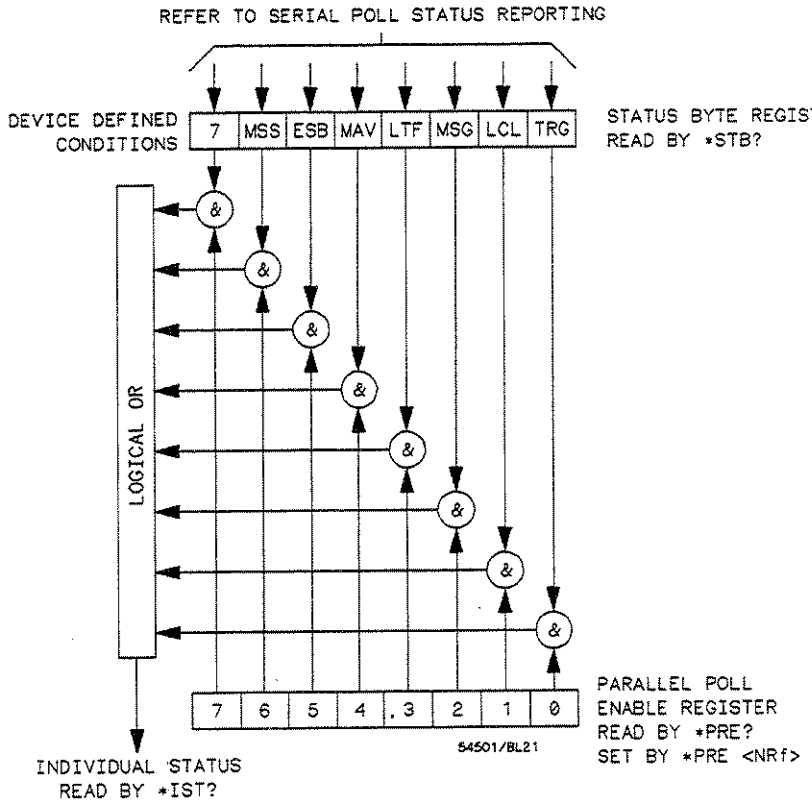


Figure 3-32. Parallel Poll Data Structure.

Polling HP-IB Devices. Parallel Poll is the fastest means of gathering device status when several devices are connected to the bus. Each device (with this capability) can be programmed to respond with one bit of status when parallel polled. This makes it possible to obtain the status of several devices in one operation. If a device responds affirmatively to a parallel poll, more information about its specific status can be obtained by conducting a serial poll of the device.

Configuring Parallel Poll Responses Certain devices, including the HP 54501A, can be remotely programmed by a controller to respond to a parallel poll. A device which is currently programmed for a parallel poll responds to the poll by placing its current status on one of the bus data lines. The response and the data-bit number can be programmed by the PPC (Parallel Poll Configure) statement. Multiple listeners cannot be specified in this statement. If more than one device is to respond on a single bit, each device must be configured with a separate PPC statement.

Example:
```
ASSIGN @Device to 707
PPOLL CONFIGURE @Device;Mask
```

The value of Mask (any numeric expression can be specified) is first rounded and then used to configure the device's parallel response. The least significant 3 bits (bits 0 through 2) of the expression are used to determine which data line the device is to respond on (place its status on). Bit 3 specifies the "true" state of the parallel poll response bit of the device. A value of 0 implies that the device's response is 0 when its status bit message is true.

Example: The following statement configures the device at address 07 on the interface select 7 to respond by placing a 0 on DIO4 when its status response is "true."

```
PPOLL CONFIGURE 707;4
```

Conducting a Parallel Poll

The PPOLL (Parallel Poll) function returns a single byte containing up to 8 status bit messages for all devices on the bus capable of responding to the poll. Each bit returned by the function corresponds to the status bit of the devices(s) configured to respond to the parallel poll (one or more devices can respond on a single line). The PPOLL function can only be executed by the controller. It is initiated by the simultaneous assertion of ATN and EOI.

Example:

```
Response = PPOLL(7)
```

Disabling Parallel Poll Responses

The PPU (Parallel Poll Unconfigure) statement gives the controller the capability of disabling the parallel poll response of one or more devices on the bus.

Examples: The following statement disables device five only:

```
PPOLL UNCONFIGURE 705
```

This statement disables all devices on interface select code eight from responding to a parallel poll:

```
PPOLL UNCONFIGURE 8
```

If no primary address is specified, all bus devices are disabled from responding to a parallel poll. If a primary address is specified, only the specified devices (which have the parallel poll configure capability) are disabled.

HP-IB Commands

The following paragraphs describe actual HP-IB command which can be used to perform the functions of the Basic commands shown in the previous examples.

Parallel Poll Unconfigure Command. The parallel poll unconfigure command (PPU) resets all parallel poll devices to the idle state (unable to respond to a parallel poll).

Parallel Poll Configure Command. The parallel poll configure command (PPC) causes the addressed listener to be configured according to the parallel poll enable secondary command PPE.

Parallel Poll Enable Command. The parallel poll enable secondary command (PPE) configures the devices which have received the PPC command to respond to a parallel poll on a particular HP-IB DIO line with a particular level.

Parallel Poll Disable Command. The Parallel Poll disable secondary command (PPD) disables the devices which have received the PPC command from responding to parallel poll.

Table 3-4. Parallel Poll Commands

Command	Mnemonic	Decimal Code	ASCII/ISO Character
Parallel Poll Unconfigure (Multiline Command)	PPU	21	NAK
Parallel Poll Configure (Secondary Command)	PPC	05	ENQ
Parallel Poll Enable (Secondary Command)	PPE	96-111	I-O
Parallel Poll Disable (Secondary Command)	PPD	112	P

Programming and Documentation Conventions

4

Introduction

This section 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. For more information on command syntax refer to the chapter "Message Communication and System Functions."

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:

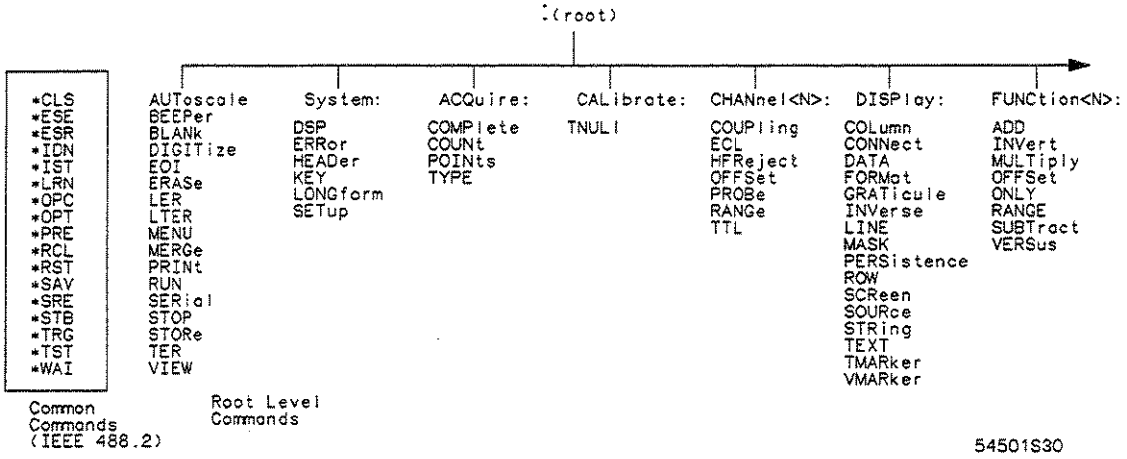
> The fourth character is a vowel, then the mnemonic is the first three characters of the keyword.

This rule will not be 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 4-1.

Table 4-1. Mnemonic Truncation

Longform	Shortform
RANGE	RANG
PATTERN	PATT
TIME	TIME
DELAY	DEL



This instrument contains four identical channel subsystems and two identical function subsystems. The "N" in the Channel header must be 1 through 4, and the Function header must be 1 or 2.

Figure 4-1. The HP 54501A Command Tree

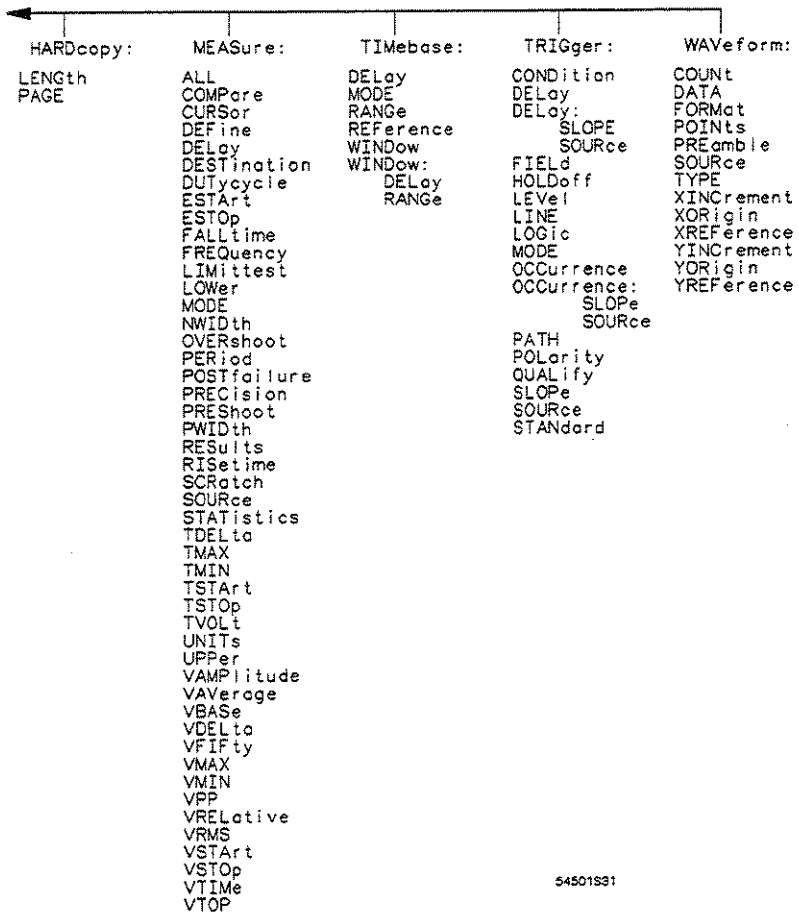


Figure 4-1. The HP 54501A Command Tree (continued)

The Command Tree

The command tree (figure 4-1) shows all commands in the HP 54501A 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. After a <NL> (linefeed - ASCII decimal 10) has been sent to the instrument, the parser will be set to the "root" of the command tree.

Command Types

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

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

Example: ***RST**.

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

Example: **:AUTOSCALE**

Subsystem commands. Subsystem commands are grouped together under a common node of the tree, such as the TIMEBASE commands.

Tree Traversal Rules

Command headers are created by traversing down the command tree. A legal command header from the command tree in figure 4-1 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 a <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 4-1, use the last mnemonic in the compound header as a reference point (for example, RANGE). Then find the last colon above that mnemonic (CHANNEL1:), and that is where the parser will be. Any command below that point can be sent within the current program message without sending the mnemonic(s) which appear above them (OFFSET).

Examples

The OUTPUT statements are written using HP BASIC 4.0 on a HP 9000 Series 200/300 Controller. The quoted string is placed on the bus, followed by a carriage return and linefeed (CRLF).

Example 1: OUTPUT 707;":CHANNEL1:RANGE 0.5 ;OFFSET 0"

Comments: The colon between CHANNEL1 and RANGE is necessary, 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 set 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, because the <program message terminator> will place 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 example 2.

Example 3: OUTPUT 707;":TIM:REF CENTER ; :CHAN1:OFFSET 0"

Comments: The leading colon before CHAN1 tells the parser to go back to the root of the command tree. The parser can then see the CHAN1: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, and therefore the command following an overlapped command may be started before the overlapped command is completed. All the commands of the HP 54501A are **sequential.**

Response Generation

IEEE 488.2 defines two times at which query responses may be buffered. The first is when the query is parsed by the instrument, the second is when the controller addresses the instrument to talk so that it may read the response. The HP 54501A will buffer responses to a query when the query is parsed.

Notation Conventions and Definitions

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

< > Angular brackets enclose words or characters that are used to 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) is used to 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 must be selected.

The following definitions are used:

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

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

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

<sp> ::= <white space>

<white space> ::= 0 through 32 (decimal) except linefeed (decimal 10).

Syntax Diagrams

At the beginning of each of the following chapters are 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.

Command Structure

The HP 54501A programming commands are divided into three types: common commands, root level commands, and subsystem commands. A programming command tree is shown in figure 4-1.

Common Commands

The common commands are the commands defined by IEEE 488.2. These commands control some functions that are common to all IEEE 488.2 instruments. Sending the common commands do not take the instrument out of a selected subsystem.

Root Level Commands

The root level commands control many of the basic functions of the instrument.

Subsystem Commands

There are several subsystems in this instrument. Only one subsystem may be selected at any given time. At power on, the command parser is set to the root of the command tree, therefore no subsystem is selected.

Note

When a program message terminator or a leading colon (:) is sent in a program message, the command parser is returned to the root of the command tree

The 11 subsystems in the HP 54501A are:

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).

Channel - controls all Y-axis oscilloscope 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 printing of waveforms.

Measure - selects the automatic measurements to be made.

Timebase - controls all X-axis oscilloscope functions.

Trigger - controls the trigger modes and parameters for each trigger mode.

Waveform - provides access to waveform data, including active data from channels and functions as well as static data from waveform memories.

Program Examples

The program examples given for each command in the following chapters and appendices were written on an HP 9000 Series 200/300 controller using the HP BASIC 4.0 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, special attention should be paid to the ways in which the command/query can be sent. The way the instrument is set up to respond to a command/query has no bearing on how you send the command/query. That is, the command/query can be sent using the longform or shortform if one exists for that command. You can send the command/query using upper case (capital) letters or lower case (small) letters, both work the same. Also, the data can be sent using almost any form you wish. If you were 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 longform and using the decimal format.

  OUTPUT 707;":CHANNEL1:RANGE .1"

- commands in shortform and using an exponential format.

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

- commands using lower case letters, shortforms, and a suffix.

  OUTPUT 707;":chan1:rang 100 mV"

Note

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 HP 54501A, then you should use a numeric variable. In this case the headers should be turned off.

Command Set Organization

The command set for the HP 54501A is divided into 13 separate groups: Common commands, root level commands and 11 sets of subsystem commands. Each of the 13 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 those commands, and finally, the commands for that subsystem in alphabetic order. The commands are shown in the longform and shortform using upper and lowercase letters. As an example, AUToscale indicates that the longform of the command is AUTOSCALE and the shortform of the command is AUT. Each command listing contains a description of the command and its arguments, the command syntax, and a programming example.

Table 4-2 lists the commands for the HP 54501A in alphabetical order with their corresponding subsystem or command type.

Table 4-2. Alphabetic Command Cross-Reference

Command	Where Used	Command	Where Used
ADD	FUNCtion Subsystem	ERRor	SYSTem Subsystem
ALL	MEASure Subsystem	*ESE	Common Command
AUToscale	Root Level Command	*ESR	Common Command
BEEPer	Root Level Command	ESTArt	MEASure Subsystem
BLANk	Root Level Command	ESTOp	MEASure Subsystem
*CLS	Common Command	FALLtime	MEASure Subsystem
COLumn	DISPlay Subsystem	FIELd	TRIGger Subsystem
COMPare	MEASure Subsystem	FORMat	DISPlay Subsystem
COMPlete	ACQuire Subsystem	FORMat	WAVeform Subsystem
CONDition	TRIGger Subsystem	FREQuency	MEASure Subsystem
CONNect	DISPlay Subsystem	GRATicule	DISPlay Subsystem
COUNt	ACQuire Subsystem	HEADer	SYSTem Subsystem
COUNt	WAVeform Subsystem	HFReject	CHANnel Subsystem
COUPling	CHANnel Subsystem	HOLDoff	TRIGger Subsystem
CURSor	MEASure Subsystem	*IDN	Common Command
DATA	DISPlay Subsystem	INVerse	DISPlay Subsystem
DATA	WAVeform Subsystem	INVert	FUNCtion Subsystem
DEFine	MEASure Subsystem	*IST	Common Command
DELay	MEASure Subsystem	KEY	SYSTem Subsystem
DELay	TIMebase Subsystem	LENGth	HARDcopy Subsystem
DELay	TRIGger Subsystem	LER	Root Level Command
DELay:SLOPe	TRIGger Subsystem	LEVel	TRIGger Subsystem
DELay:SOURce	TRIGger Subsystem	LIMittest	MEASure Subsystem
DESTination	MEASure Subsystem	LINE	DISPlay Subsystem
DIGitize	Root Level Command	LINE	TRIGger Subsystem
DSP	SYSTem Subsystem	LOGic	TRIGger Subsystem
DUTycycle	MEASure Subsystem	LONGform	SYSTem Subsystem
ECL	CHANnel Subsystem	LOWer	MEASure Subsystem
EOI	Root Level Command	*LRN	Common Command
ERASe	Root Level Command	LTER	Root Level Command

Table 4-2. Alphabetic Command Cross-Reference (Continued)

Command	Where Used	Command	Where Used
MASK	DISPlay Subsystem	PROBe	CHANnel Subsystem
MENU	Root Level Command	PWIDth	MEASure Subsystem
MERGe	Root Level Command	QUALify	TRIGger Subsystem
MODE	MEASure Subsystem	RANGe	CHANnel Subsystem
MODE	TIMebase Subsystem	RANGe	FUNCtion Subsystem
MODE	TRIGger Subsystem	RANGe	TIMebase Subsystem
MULTiply	FUNCtion Subsystem	*RCL	Common Command
NWIDth	MEASure Subsystem	REFerence	TIMebase Subsystem
OCCurrence	TRIGger Subsystem	RESults	MEASure Subsystem
OCCurrence:SLOPe	TRIGger Subsystem	RISetime	MEASure Subsystem
OCCurrence:SOURce	TRIGger Subsystem	ROW	DISPlay Subsystem
OFFSet	CHANnel Subsystem	*RST	Common Command
OFFSet	FUNCtion Subsystem	RUN	Root Level Command
ONLY	FUNCtion Subsystem	*SAV	Common Command
*OPC	Common Command	SCRatch	MEASure Subsystem
*OPT	Common Command	SCReen	DISPlay Subsystem
OVERshoot	MEASure Subsystem	SERial	Root Level Command
PAGE	HARDcopy Subsystem	SETup	SYSTem Subsystem
PATH	TRIGger Subsystem	SLOPe	TRIGger Subsystem
PERiod	MEASure Subsystem	SOURce	DISPlay Subsystem
PERSistence	DISPlay Subsystem	SOURce	MEASure Subsystem
POINts	ACQuire Subsystem	SOURce	TRIGger Subsystem
POINts	WAVeform Subsystem	SOURce	WAVeform Subsystem
POLarity	TRIGger Subsystem	*SRE	Common Command
POSTfailure	MEASure Subsystem	STANdard	TRIGger Subsystem
*PRE	Common Command	STATistics	MEASure Subsystem
PREamble	WAVeform Subsystem	*STB	Common Command
PRECision	MEASure Subsystem	STOP	Root Level Command
PREShoot	MEASure Subsystem	STORe	Root Level Command
PRINt	Root Level Command	STRing	Display Subsystem

Table 4-2. Alphabetic Command Cross-Reference (Continued)

Command	Where Used	Command	Where Used
SUBTract	FUNCtion Subsystem	VMARker	DISPlay Subsystem
TDELta	MEASure Subsystem	VMAX	MEASure Subsystem
TER	Root Level Command	VMIN	MEASure Subsystem
TEXT	DISPlay Subsystem	VPP	MEASure Subsystem
TMARker	DISPlay Subsystem	VRELative	MEASure Subsystem
TMAX	MEASure Subsystem	VRMS	MEASure Subsystem
TMIN	MEASure Subsystem	VSTArt	MEASure Subsystem
TNULl	CALibrate Subsystem	VSTOp	MEASure Subsystem
*TRG	Common Command	VTIMe	MEASure Subsystem
*TST	Common Command	VTOP	MEASure Subsystem
TSTArt	MEASure Subsystem	*WAI	Common Command
TSTOp	MEASure Subsystem	WINDow	TIMebase Subsystem
TTL	CHANnel Subsystem	WINDow:DELay	TIMebase Subsystem
TVOLt	MEASure Subsystem	WINDow:RANGe	TIMebase Subsystem
TYPE	ACQuire Subsystem	XINCrement	WAVeform Subsystem
TYPE	WAVeform Subsystem	XORigin	WAVeform Subsystem
UNITs	MEASure Subsystem	XREFerence	WAVeform Subsystem
UPPer	MEASure Subsystem	YINCrement	WAVeform Subsystem
VAMPlitude	MEASure Subsystem	YORigin	WAVeform Subsystem
VAVerage	MEASure Subsystem	YREFerence	WAVeform Subsystem
VBASe	MEASure Subsystem		
VDELta	MEASure Subsystem		
VERSus	FUNCtion Subsystem		
VFIFty	MEASure Subsystem		
VIEW	Root Level Command		

Common Commands 5

The common commands are defined by the IEEE 488.2 standard. These commands will be common to all instruments that comply with this 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.

Common commands can be received and processed by the HP 54501A whether they are sent over the HP-IB 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 will remain in the selected subsystem. For example, if the program message "**ACQUIRE:COUNT 1024; *CLS; TYPE AVERAGE**" is received by the instrument, the instrument will set the acquire count and type, and clear the status information. This would not be the case if some other type of command were received within the program message. For example the program message "**:ACQUIRE:COUNT 1024; :AUTOSCALE; :ACQUIRE:TYPE AVERAGE**" would set the acquire count, complete the autoscale, then set the acquire type. In this example :ACQUIRE must be sent again in order to reenter the acquire subsystem and set the type.

Refer to figure 5-1 for common commands syntax diagram.

Note

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 chapter 3.

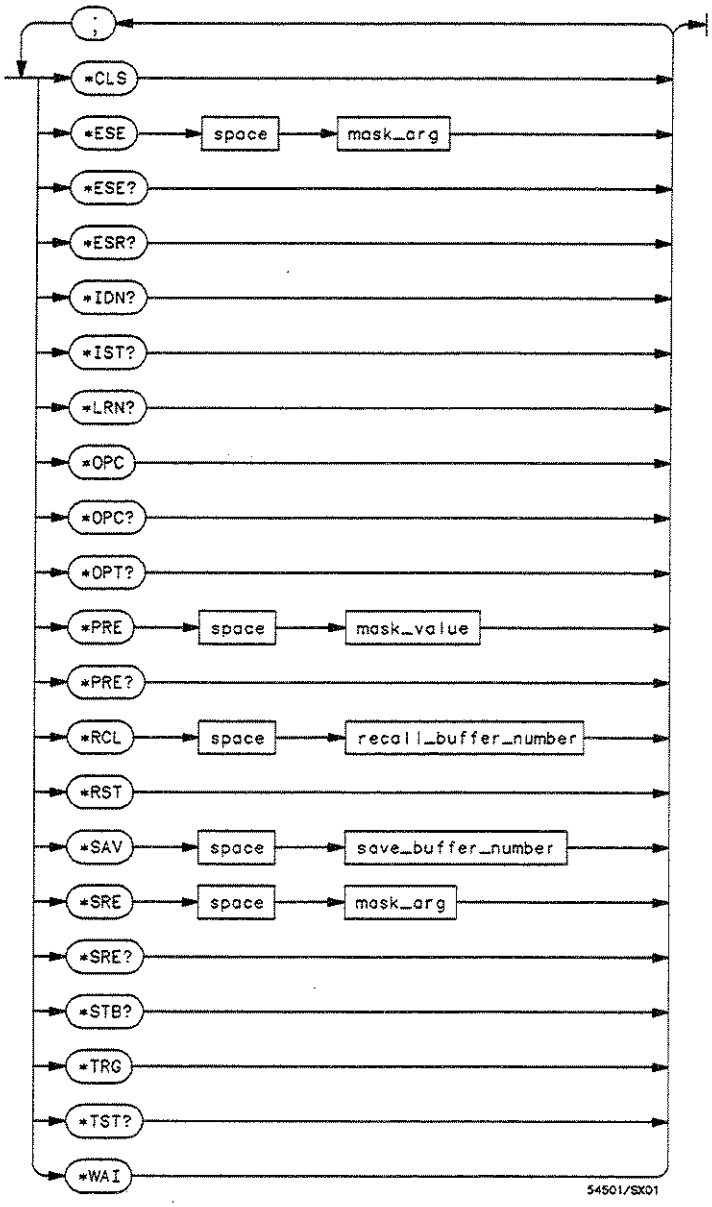


Figure 5-1. Common Commands Syntax Diagram

mask_arg = An integer, 0 through 255. This number is the sum of all the bits in the mask corresponding to conditions that are enabled. Refer to the *ESE and *SRE commands for bit definitions in the enable registers.

mask_value = An integer, 0 through 255. This number is the sum of all the bits in the mask corresponding to conditions that are enabled. Refer to the *IST? query.

recall_buffer_number = An integer, 0 through 4.

save_buffer_number = An integer, 1 through 4.

Figure 5-1. Common Commands Syntax Diagram (continued)

*CLS (Clear Status) command

The *CLS (clear status) common command clears the status data structures, including the device defined error queue. This command also clears the Request-for-OPC flag.

If the *CLS command immediately follows a PROGRAM MESSAGE TERMINATOR, the output queue and the MAV (message available) bit will be cleared.

Command Syntax: *CLS

Example: OUTPUT 707;"*CLS"

Note

Refer to chapter 3 for a complete discussion of status.

*ESE (Event Status Enable) command/query

The *ESE command sets the Standard Event Status Enable Register bits. 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 will enable the corresponding bit in the Standard Event Status Register, a zero will disable the bit. Refer to table 5-1 for the information about the Standard Event Status Enable Register bits, bit weights, and what each bit masks.

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

Command Syntax: *ESE <mask>

Where:

<mask> ::= 0 to 255

Example: OUTPUT 707;"*ESE 64"

In this example, the *ESE 64 command will enable 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 will also be set.

*ESE

Query Syntax: *ESE?

Returned Format: <mask><NL>

Where:

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

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

Table 5-1. Standard Event Status Enable Register

Event Status Enable Register (High - Enables the ESR 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

Note

Refer to chapter 3 for a complete discussion of status.

*ESR (Event Status Register) query

The *ESR query returns the contents of the Standard Event Status Register.

Note

Reading the register clears the Standard Event Status Register.

Query Syntax: *ESR?

Returned Format: <status><NL>

Where:

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

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

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

Table 5-2. Standard Event Status Register

Bit	Bit Weight	Bit Name	Condition
7	128	PON	1 = an OFF to ON transition has occurred
6	64	URQ	0 = no front-panel key has been pressed 1 = 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 = request control - NOT used - always 0
0	1	OPC	0 = operation is not complete 1 = operation is complete

0 = False = Low
1 = True = High

*IDN (Identification Number) query

The *IDN query allows the instrument to identify itself. It returns the string:

"HEWLETT-PACKARD,54501A,<XXXXAYYYYY>,<MMDD>"

Where:

<XXXXAYYYYY> ::= the serial number of this instrument.

<MMDD> ::= the software revision of this instrument. The first two parameters represent the month and the second two parameters represent the day of the month.

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

Query Syntax: *IDN?

Returned Format: HEWLETT-PACKARD,54501A,XXXXAYYYYY,MMDD<NL>

Example:

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

*IST (Individual Status) query

The *IST query allows the instrument to identify itself during a parallel poll by allowing the controller to read the current state of the IEEE 488.1 defined "ist" local message in the instrument. The response to this query is dependent upon the current status of the instrument

Query Syntax: *IST?

Returned Format: <id> <NL>

Where:

<id> ::= 0 or 1

Where:

0 indicates the "ist" local message is false.
1 indicates the "ist" local message is true.

Example:
```
OUTPUT 707;"*IST?"
ENTER 707;Id$
PRINT Id$
```

*LRN (Learn) query

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

This command allows you to store an instrument setup in the controller. The stored setup can then be returned to the instrument when you want that setup at a later time.

This command performs the same function as the :SYSTEM:SETUP? query. The data can be sent to the instrument using the :SYSTEM:SETUP command.

Note

*The returned header for the *LRN query is :SYSTEM:SETUP. The :SYSTEM:HEADER command does NOT effect this returned header.*

Query Syntax: *LRN?

Returned Format: :SYSTem:SETup <setup><NL>

Where:

<setup> ::= #800001024<learn string><NL>

The learn string is 1024 data bytes in length.

Example:
```
DIM Lrn$[2000]
OUTPUT 707;"*LRN?"
ENTER 707 USING "-K";Lrn$
```

*OPC (Operation Complete) command/query

The *OPC (operation complete) command will cause the instrument to set the operation complete bit in the Standard Event Status Register when all pending device operations have finished.

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

Command Syntax: *OPC

Example:
```
OUTPUT 707;"*OPC"
```

Query Syntax: *OPC?

Returned Format: 1<NL>

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

*OPT (Option) query

The *OPT query is used to report the options installed in the instrument. This query will always return a zero because the HP 54501A does not have any possible options to report.

Query Syntax: *OPT?

Returned Format: 0<NL>

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

*PRE (Parallel Poll Register Enable) command/query

The *PRE command sets the parallel poll register enable bits.

The Paralled Poll Enable Register contains a mask value for the bits to be enabled that can produce an "ist" during a parallel poll.

The *PRE query returns the current value.

Command Syntax: *PRE <mask>

Where:

<mask> ::= 0 to 255

Example:
```
OUTPUT 707;"*PRE 16"
```

Note

This example will allow the HP 54501A to generate an "ist" when a message is available in the output queue. When a message is available, the MAV bit in the Status Byte Register will be high.

Query Syntax: *PRE?

Returned Format: <mask_value> <NL>

Where:

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

Example:
```
OUTPUT 707;"*PRE?"
ENTER 707;V$
Print V$
```

*RCL (Recall) command

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 4 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.

Note

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

Command Syntax: *RCL <rcl_register>

Where:

<rcl_register> ::= 0 through 4

Example: OUTPUT 707;"*RCL 3"<NL>

*RST (Reset) command

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

Command Syntax: *RST

Example: OUTPUT 707;"*RST"

Table 5-3. Reset Conditions for the HP 54501A

Timebase Menu	
reference	cntr
Time/Div	100 µs
delay	0.00 s
timebase window	off
Channel Menu	
Channel 1	on
Channel 2, 3, and 4	off
Volts/Div	500 mV
offset	0.00
coupling	dc
probe attenuation	1.000:1
Trigger Menu	
Mode	edge
triggering	auto
source	Channel 1
level	0.0 V
slope	positive
holdoff	40 ns

Table 5-3. Reset Conditions for the HP 54501A (continued)

Display Menu	
Mode	norm
persistence	minimum
off/frame/axes/grid	axes
connect dots	off
# of screens	1
Δt/ΔV	
Δt markers	off
ΔV markers	off
Waveform Math Menu	
f1	off
f2	off
display	off
chan/mem	chan 1
operator	+
chan/mem	chan 1
function sensitivity	1.00 V/div
function offset	0.0 V
Waveform Save Menu	
waveform/pixel	waveform
nonvolatile	m1
display	off
source	chan 1
Define Meas Menu	
meas/def.limit	meas
continuous	on
statistics	off
Utility Menu	
clicker	on

*SAV (SAVE) command

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 4 are valid for this command.

Command Syntax: *SAV <save_register>

Where:

<save_register> ::= 1 through 4

Example: OUTPUT 707;"*SAV 3"

*SRE (Service Request Enable) command/query

The *SRE command sets the Service Request Enable Register bits. 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 will enable the corresponding bit in the Status Byte Register, a zero will disable the bit. Refer to table 5-5 for the bits in the Service Request Enable Register and what they mask.

The *SRE query returns the current value.

Command Syntax: *SRE <mask>

Where:

<mask> ::= 0 to 255

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

Note

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 will be high.

Query Syntax: *SRE?

Returned Format: <mask><NL>

Where:

<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

Service Request Enable Register (High - Enables the SRE bit)		
Bit	Weight	Enables
7	128	not used
6	64	RQS - Request Service
5	32	ESB - Event Status Bit
4	16	MAV - Message Available
3	8	LTF - Limit Test Fail
2	4	MSG - Message
1	2	LCL - Local
0	1	TRG - Trigger

*STB (Status Byte) query

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 5-6 for the meaning of the bits in the status byte.

Note

To read the instrument's status byte with RQS reported on bit 6, use the HP-IB Serial Poll.

Query Syntax: *STB?

Returned Format: <value><NL>

Where:

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

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

Table 5-5. The 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

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

Command Syntax: *TRG

Example: OUTPUT 707;"*TRG"

*TST (Test) query

The *TST query causes the instrument to perform a self-test. The result of the test will be placed in the output queue.

Note

Prior to sending this command all front panel inputs must be disconnected.

A 0 indicates the test passed and a non-zero value indicates the test failed.

If a test fails refer to the troubleshooting section of the Service Manual.

Query Syntax: *TST?

Returned Format: <result><NL>

Where:

<result> ::= 0 or non-zero value

Where:

0 indicates the test passed.
non-zero indicates the test failed.

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

*WAI (Wait) command

The *WAI command has **no function** in the HP 54501A, but is parsed for compatibility with other instruments.

Command Syntax: *WAI

Example: OUTPUT 707;"*WAI"

Root Level Commands 6

Introduction

Root Level commands control many of the basic operations of the oscilloscope. These commands will always be 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. Figure 6-1 lists the root level commands syntax diagram.

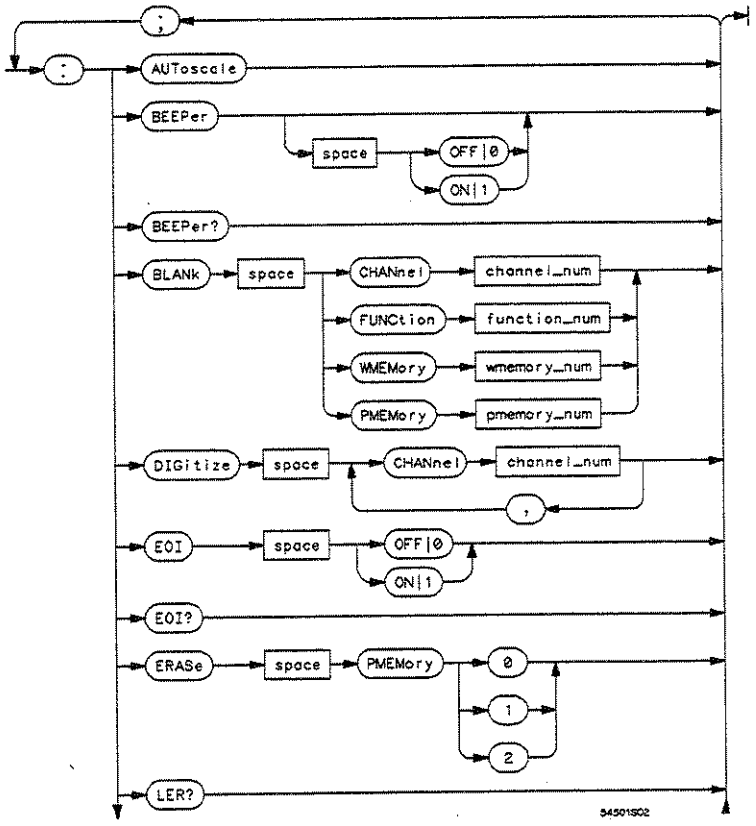


Figure 6-1. Root Level Commands Syntax Diagram

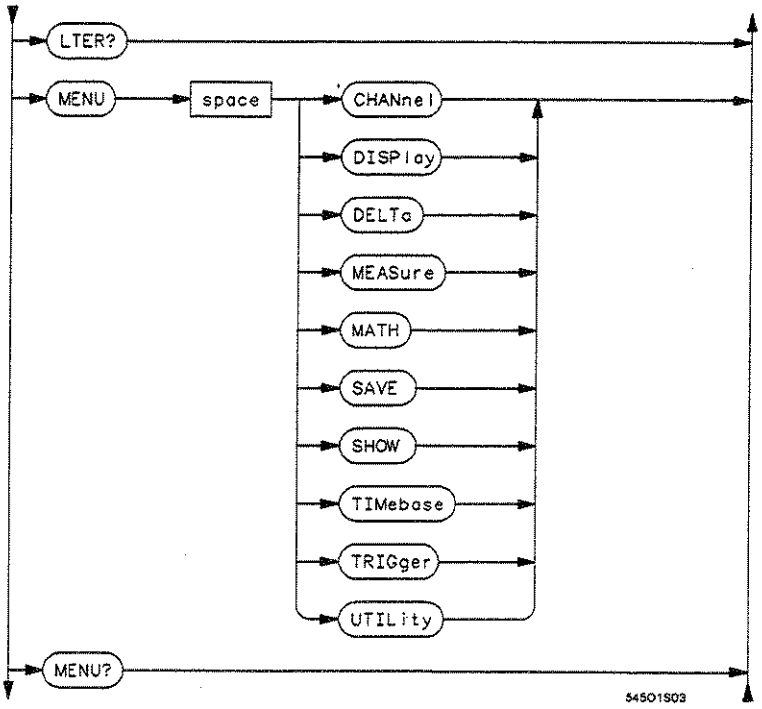
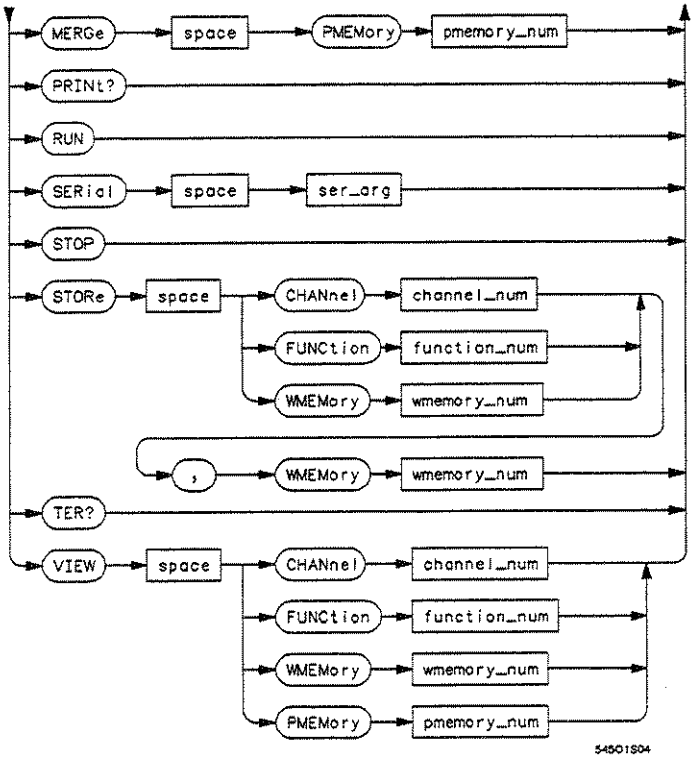


Figure 6-1. Root Level Commands Syntax Diagram (continued)



channel_num = an integer 1, 2, 3, or 4.

function_num = an integer 1 or 2.

wmemory_num = an integer 1 through 4.

pmemory_num = an integer 1 or 2.

ser_arg = a 10 character quoted string.

Figure 6-1. Root Level Commands Syntax Diagram (continued)

AUToscale command

The AUTOSCALE command causes the oscilloscope to evaluate all input signals and set 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 with minimum persistence;
- the trigger level, holdoff, and slope; and
- the sweep speed of the displayed channel.

In addition, the AUTOSCALE command turns off:

- markers;
- all measurements;
- functions;
- windows;
- memories; and
- connect the dots.

If input signals are present on more than one vertical input, the sweep will be triggered on channel 1 if a signal is present on that channel. If a signal is not present on channel 1 then the oscilloscope will be triggered on channel 4 if a signal is present on that channel. If a signal is not present on channel 4 then the oscilloscope will be triggered on channel 2 if a signal is present on that channel, or on channel 3 if no other signals are found. If no signals are found on any vertical input, the oscilloscope is returned to its former state.

Command Syntax: :AUToscale

Example: OUTPUT 707;":AUTOSCALE"

BEEPer — command/query

The BEEPER command sets the beeper mode, which controls the sound 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.

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

Command Syntax: :BEEPer [{{ON | 1} | {OFF | 0}}]

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

Query Syntax: :BEEPer?

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

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

BLANk command

The BLANK command causes the instrument to turn off (stop displaying) the specified channel, function, pixel memory, or waveform memory. To blank a channel display use the command :BLANK CHANNEL{1|2|3|4}. To blank a waveform memory display use :BLANK WMEMORY{1|2|3|4}, to blank a pixel memory display use the command :BLANK PMEMORY{1|2}, and to blank a function use the command :BLANK FUNCTION{1|2}.

Command Syntax: :BLANk <display>

Where:

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

Example: OUTPUT 707;":BLANK CHANNEL1"

DIGitize command

The DIGITIZE command is used to acquire waveform data for transfer over the HP-IB. It causes an acquisition to take place on the specified channel(s) with the resulting data being placed in the channel buffer.

The ACQUIRE subsystem commands are used to set up conditions such as TYPE, number of POINTS, and the 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 will be transferred, refer to the WAVEFORM Subsystem commands. For more information on the DIGITIZE command refer to the section on the DIGITIZE command in the chapter "Introduction to Programming an Oscilloscope."

Note

Sending the DIGITIZE command will turn off any unused channels.

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 will be overwritten. Therefore, ensure all operations that require the digitized data are completed before restarting the HP 54501A.

The speed of the total digitize operations may be improved if two or more DIGITIZE commands are sent without changing other parameters.

The sources for the :DIGITIZE command are channels 1 through 4.

Command Syntax: :DIGitize CHANnel<N>[,CHANnel<N>]...

Where:

<N> ::= 1, 2, 3, or 4.

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

EOI (End or Identify) command/query

The EOI command specifies whether or not the last byte of a reply from the HP 54501A is to be sent with the EOI bus control line set true or not true. The last byte of a response is usually a "new line" character, ASCII decimal 10 (LF).

Note

EOI must be turned on for response messages to be in compliance with IEEE 488.2.

The EOI query returns the current state of EOI.

Command Syntax: :EOI {{ON | 1} | {OFF | 0}}

Example: OUTPUT 707;":EOI OFF"

Query Syntax: :EOI?

Returned Format: [:EOI] {1 | 0}<NL>

Example:
```
OUTPUT 707;":EOI?"
ENTER 707;End$
PRINT End$
```

ERASe command

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 front-panel key. If the scope is running and being triggered and ERASE PMEMORY0 is executed, the instrument will momentarily stop acquiring data, clear the CRT and continue with data acquisition.

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

Note

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

Command Syntax: :ERASe {PMEMory0 | PMEMory1 | PMEMory2}

Example: OUTPUT 707;":ERASE PMEMORY1"

LER (Local Event Register) query

The LER query allows the LCL (Local) Event Register to be read. 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 would like a new Service Request to be generated.

Query Syntax: :LER?

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

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

LTER (Limit Test Event Register) query

The LTER query allows the Limit Test Event Register to be read. The Limit Test Event Register contains the Limit Test Fail bit. This bit is set when the limit test is active and a limit 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.

Query Syntax: :LTER?

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

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

MENU command/query

The MENU command selects one of the 10 menus on the front panel.

The MENU query returns the current menu.

Command Syntax: :MENU <name> <NL>

Where:

<name> ::= {TIMebase | CHANnel | TRIGger | DISPlay | DELTa | MATH | SAVE | MEASure | UTILity | SHOW}

Example: OUTPUT 707;":MENU DISPLAY"

Query Syntax: :MENU?

Returned Format: [:MENU] <name> <NL>

Where:

<name> ::= {TIMebase | CHANnel | TRIGger | DISPlay | DELTa | MATH | SAVE | MEASure | UTILity | SHOW}

Example:
```
DIM Name$[30]
OUTPUT 707;":MENU?"
ENTER 707;Name$
PRINT Name$
```

MERGe command

The MERGE command stores the contents of the active display into the specified pixel memory. The pixel memories are PMEMORY 1 or 2.

This command has a similar function as the "add to memory" key in the pixel menu of the Waveform Save menu.

Command Syntax: :MERGe {PMEMory1 | PMEMory2}

Example: OUTPUT 707;":MERGE PMEMory2"

PRINt query

The PRINT query outputs a copy of the display as soon as the oscilloscope is addressed to talk.

The output includes the displayed waveforms, the graticule, time and voltage markers, trigger setup, and measurement results.

Query Syntax: :PRINt?

Example:
```
OUTPUT 707;":HARDCOPY:PAGE AUTOMATIC"
OUTPUT 707;":PRINT?"
SEND 7;UNT UNL
SEND 7;LISTEN 1     !Assumes printer is at address 1
SEND 7;TALK 7
SEND 7;DATA
```

RUN command

The RUN command acquires data for the active waveform display. The data is acquired in the manner defined by the timebase mode.

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

If the timebase mode is AUTO or TRIGGERED, the RUN command will enable the trigger repeatedly and display the data it acquires continuously on the CRT. This is the same thing that happens when the front panel RUN key is pressed. See the :TIMEBASE:MODE command for a description of the various modes.

Command Syntax: :RUN

Example: OUTPUT 707;":RUN"

SERial (Serial Number) command

The SERIAL command allows you to enter a serial number in the instrument. The instrument serial number is entered at the factory, therefore this will normally not be required.

The serial number is placed in protected non-volatile ram, so the protection switch must be in the unprotected position to write a new serial number to the instrument.

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

Note

A serial number corresponding to the serial number of the board in the HP 54501A is loaded at the factory. Do not use this command unless you need to serialize the instrument for a different application.

Command Syntax: :SERial <string>

Where:

<string> ::= 10 character serial number within quotes

Example: OUTPUT 707;":SER ""1234A56789"""

STOP command

The STOP command causes the instrument to stop acquiring data for the active display.

The RUN command must be executed to restart the acquisition.

Command Syntax: :STOP

Example: OUTPUT 707;":STOP"

STORe command

The STORE command moves a stored waveform, channel, or function to a waveform memory. This command has two parameters. The first is the source of the waveform. The source 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.

Command Syntax: :STORe <source>,<destination>

where:

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

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

TER (Trigger Event Register) query

The TER query allows the Trigger Event Register to be read. When 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 this bit will not be set.

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.

Query Syntax: :TER?

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

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

VIEW command

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

If you want to display a channel use the command :VIEW CHANnel{1 | 2 | 3 | 4}. If you want to display a pixel memory, use the parameter :VIEW PMEMory{1 | 2}. To display a function send the command :VIEW FUNCtion{1 | 2}.

The BLANK command causes the instrument to turn off (stop displaying) a specified channel, function, pixel memory, or waveform memory.

Command Syntax: :VIEW {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | PMEMory{1 | 2} | WMEMory{1 | 2 | 3 | 4}}

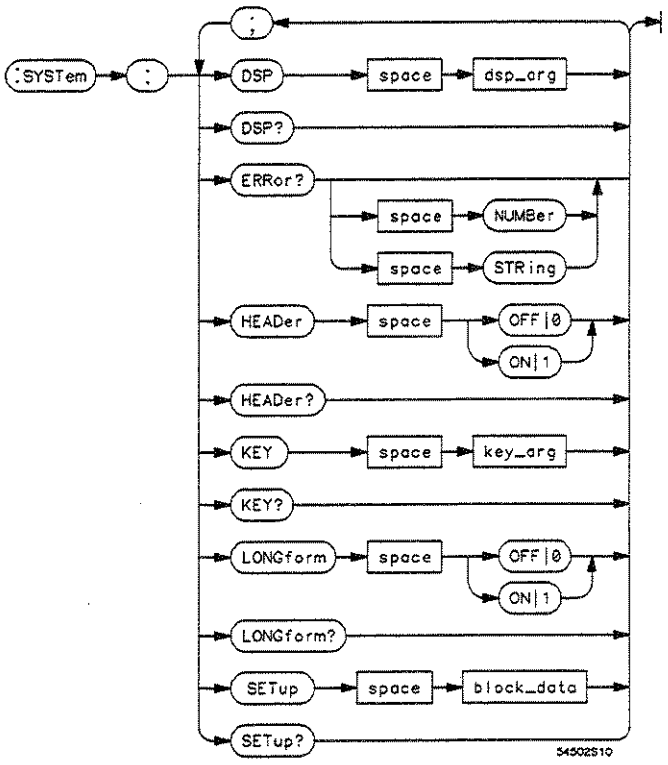
Example: OUTPUT 707;":VIEW CHANNEL1"

System Subsystem 7

Introduction

SYSTEM subsystem commands control the way in which query responses are formatted, simulate front panel key presses, and enable reading and writing to the advisory line of the instrument.

Refer to figure 7-1 for System subsystem commands syntax diagram.



dsp_arg = any quoted string

key_arg = keycode 1 through 44

block_data = block data in IEEE 488.2 # format

Figure 7-1. System Subsystem Commands Syntax Diagram

DSP

command/query

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

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.

Command Syntax: :SYSTem:DSP <quoted ASCII string>

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

Query Syntax: :SYSTem:DSP?

Returned Format: [SYSTem:DSP] <string><NL>

Where:

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

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

ERRor

query

The :SYSTEM:ERROR query outputs the next error number in the error queue over the HP-IB. This instrument has an error queue that is 30 errors deep and operates on a first-in, first-out basis. Successively 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 the numeric error code is output. No parameter specified is the same as specifying NUMBER.

See table 7-1 for the ERROR numbers.

Query Syntax: :SYSTem:ERRor? {NUMBer | STRing | (no_param)}

Returned Format: [:SYSTem:ERRor] <error>[,<quoted string>]<NL>

Where:

<error> ::= an integer error code
<quoted string> ::= an alpha string specifying the error condition

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

Table 7-1. Error Messages

Error Number	Description
11	Questionable horizontal scaling
12	Edges required not found
70	Ram write protected
-100	Command error (unknown command)
-101	Invalid character received
-110	Command header error
-111	Header delimiter error
-120	Numeric argument error
-121	Wrong data type (numeric expected)
-123	Numeric overflow
-129	Missing numeric argument
-130	Non-numeric argument error
-131	Wrong data type (char expected)
-132	Wrong data type (string expected)
-133	Wrong data type (block expected)
-134	Data Overflow: string or block to long
-139	Missing non-numeric argument
-142	Too many arguments
-143	Argument delimiter error
-144	Invalid message unit delimiter
-200	No Can Do (generic execute error)
-201	Not executable in local mode
-202	Settings lost due to rtl or power on*
-203	Trigger ignored
-211	Legal command, but settings conflict
-212	Argument out of range
-221	Busy doing something else
-222	Insufficient capability or configuration
-232	Output buffer full or overflow
-300	Device failure
-301	Interrupt fault
-302	System error
-303	Time out
-310	RAM error
-311	RAM failure (hard error)
-312	RAM data loss (soft error)
-313	Calibration data loss
-320	ROM error
-321	ROM checksum
-322	Hardware and firmware incompatible
-330	Power on test failed
-340	Self test failed
-350	Too Many Errors (error queue overflow)
-400	Query Error (generic)
-410	Query INTERRUPTED
-420	Query UNTERMINATED
-421	Query received, Indefinite block response in progress
-422	Addressed to Talk, Nothing to Say
-430	Query DEADLOCKED

rtl = remote to local

HEADer

command/query

The :SYSTEM:HEADER command tells the instrument whether or not to output a header for query responses. When HEADER is set to ON, query responses will include the command header.

The HEADER query returns the state of the HEADER command.

Command Syntax: :SYSTem:HEADer {{ ON | 1 } | { OFF | 0 }}

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

Query Syntax: :SYSTem:HEADer?

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

Where:

1 ::= ON
0 ::= OFF

Example:
```
DIM Hdr$[20]
OUTPUT 707;":SYSTEM:HEADER?"
ENTER 707;Hdr$
PRINT Hdr$
```

The following example shows the response to the query :CHANNEL1:RANGE? with the headers on and off.

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

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

With headers set to OFF: 6.40000E-01

Note

Headers should be turned off when returning values to numeric variables.

KEY

command/query

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

The KEY query returns the key code for the last key pressed from the front panel or the last simulated key press over the HP-IB. Key codes range from 1 to 44, zero represents no key and will be returned after power up.

Refer to table 7-2 for key codes.

Command Syntax: :SYSTem:KEY <keycode>

Where:

<keycode> ::= 1 to 44

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

Query Syntax: :SYSTem:KEY?

Returned Format: [:SYSTem:KEY] <keycode><NL>

Where:

<keycode> ::= 0 through 44 (integer - NR1 format)

Example:
```
DIM Input$[10]
OUTPUT 707;":SYSTEM:KEY?"
ENTER 707;Input$
PRINT Input$
```

Table 7-2. HP 54501A Front-Panel Key Codes

KEY	KEY CODE	KEY	KEY CODE
Menus - TIMEBASE	1	"-"(minus)	24
Menus - CHAN	2	"." (decimal pt.)	25
Menus - TRIG	3	0	26
Menus - DISPLAY	4	1	27
Menus - Δt ΔV	5	2	28
Menus - WFORM MATH	6	3	29
Menus - WFORM SAVE	7	4	30
Menus - DEFINE MEAS	8	5	31
Menus - UTIL	9	6	32
Function Select 1	10	7	33
Function Select 2	11	8	34
Function Select 3	12	9	35
Function Select 4	13	RUN/STOP	36
Function Select 5	14	SINGLE	37
Function Select 6	15	CLEAR DISPLAY	38
Function Select 7	16	LOCAL	39
FINE	17	HARDCOPY	40
s V	18	AUTO-SCALE	41
m mV	19	RECALL	42
μs	20	SAVE	43
ns	21	SHOW	44
CLEAR	22	no key	0
Shift (blue key)	23		

The function select keys are at the right of the CRT and are numbered from the top (10) to the bottom (16).

LONGform

command/query

The :SYSTEM:LONGFORM command sets the longform variable which tells the HP 54501A how to format query responses. If the LONGFORM command is set to OFF, command headers and alpha arguments are sent from the HP 54501A in the short form. If the LONGFORM command is set to ON, the whole word will be output. This command does not affect the input data messages to the HP 54501A. Headers and arguments may be sent to the HP 54501A in either the longform or shortform regardless of how the LONGFORM command is set. For more information refer to the HEADER command in this chapter.

The LONGFORM query returns the state of the LONGFORM command.

Note

Even though the Longform command can be sent using an alpha or numeric argument, the response is always a 1 or 0 (1 for ON, 0 for OFF).

Command Syntax: :SYSTem:LONGform {{ ON | 1 } | { OFF | 0 }}

Example:
```
OUTPUT 707;":SYST:LONG ON"
```

Query Syntax: :SYSTem:LONGform?

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

Where:

1 ::= ON
0 ::= OFF

Example:
```
DIM Long$[30]
OUTPUT 707;":SYSTEM:LONGFORM?"
ENTER 707;Long$
PRINT Long$
```

SETup

command/query

The :SYSTEM:SETUP command sets the HP 54501A as defined by the data in the learn string sent from the controller. The setup string contains 1024 bytes of setup data. The 1024 bytes does not include the header or "#800001024".

The SETUP query outputs the current HP 54501A setup in the form of a learn string to the controller. The SETUP query operates the same as the *LRN? query.

The 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.

Command Syntax: :SYSTem:SETup <setup>

Example: OUTPUT 707;":SYSTEM:SETUP <setup>"

Where:

<setup> ::= #800001024<setup data string>

Query Syntax: :SYSTem:SETup?

Returned Format: [:SYSTem:SETup] <setup> <NL>

Where:

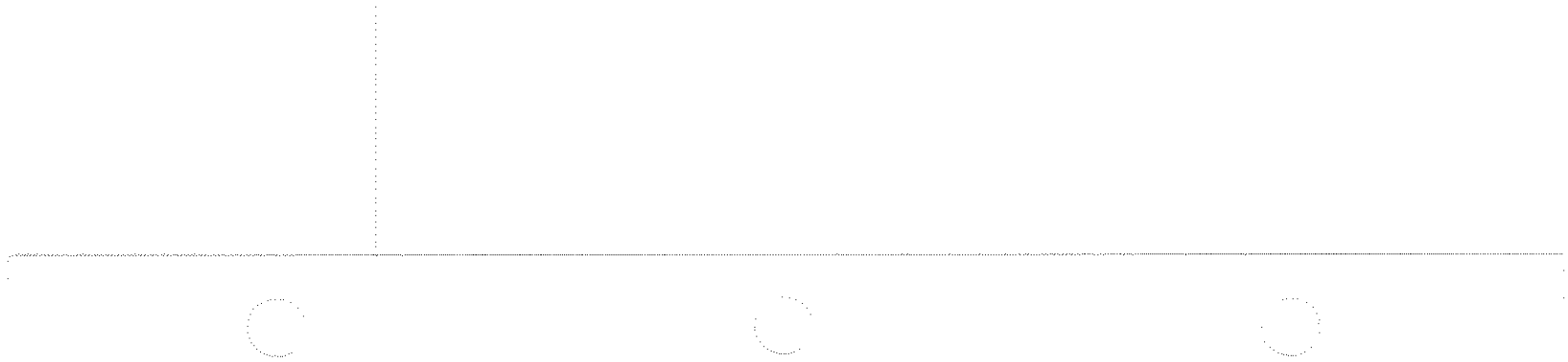
<setup> ::= #800001024 <setup data string>

Example:

```
10 DIM Set$[2000]
20 !Setup the instrument as desired
30 OUTPUT 707;":SYST:HEAD OFF"
40 OUTPUT 707;":SYSTEM:SETUP?"
50 !Transfer the instrument setup to controller
60 ENTER 707 USING "-K";Set$ !Store the setup
70 PAUSE !Change the setup as desired
80 OUTPUT 707 USING "#,K";":SYST:SETUP ";Set$
90 !Returns the instrument to the first setup
100 END
```

Note

The logical order for this instruction is to send the query first followed by the command at a time of your choosing. The query causes the learn string to be sent to the controller and the command causes the learn string to be returned to the HP 54501A.



Acquire Subsystem 8

Introduction

The ACQUIRE subsystem commands set up conditions for executing a DIGITIZE root level command to acquire waveform data. This subsystem selects the type of data, the number of averages, and the number of data points. See figure 8-1 for the Acquire subsystem commands syntax diagram.

Note

The on-screen time is divided into a specific number of horizontal time points as defined by the :ACQUIRE:POINTS command. Each of these increments in time is referred to as a time bucket with each time bucket having a fixed time associated with it.

The ACQUIRE subsystem is also the only HP-IB control for two display parameters: Display Mode and Number of Averages. There is a coupling between the front panel and the ACQUIRE subsystem parameters. This means that when the HP-IB parameters for ACQUIRE TYPE or COUNT are changed, the front panel will change. Also, when the front panel parameters are changed, the HP-IB parameters will change.

(Normal) Persistence Mode

The :ACQUIRE:TYPE NORMAL command will set the HP 54501A to the variable persistence mode. The front panel user activates the same mode by selecting the Display menu, then setting the display mode to Normal. The persistence time is set via the HP-IB in the DISPLAY subsystem using the :DISPLAY:PERSISTENCE command.

The :ACQUIRE:COUNT can be set in this mode, but has no impact on the current display mode or HP-IB acquisition. The :ACQUIRE:COUNT query will always return a 1 when the acquisition type is set to Normal.

Averaging Mode

The :ACQUIRE:TYPE AVERAGE command sets the HP 54501A to the Averaging mode.

COUNT can be set in 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 2. It determines the number of averages that must be acquired.

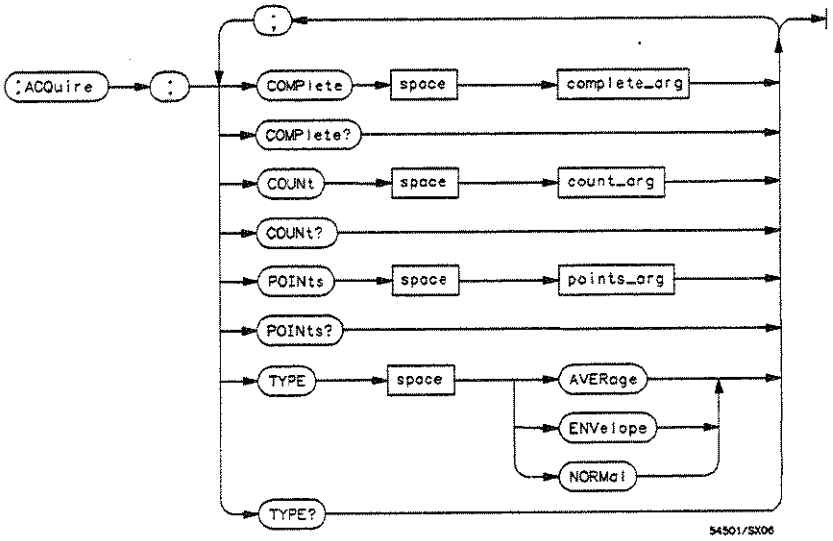
To activate the averaging mode from the front panel, select the Display menu, then select Average. Changing the number of averages changes the COUNT value.

Envelope Mode

The :ACQUIRE:TYPE ENVELOPE command sets the HP 54501A to the Envelope mode.

A count value can be set in the envelope mode. This value determines the number of values to be used, at each time point, when constructing the envelope. The count value cannot be set from the front panel.

To activate the Envelope mode from the front panel, select the Display menu, then select the Envelope mode.



complete_arg ::= An integer, 0 through 100.

count_arg ::= An integer, 1 to 2048. It specifies the number of values to average for each time point when in averaged mode, or the number of hits per each time point to form the envelope in Envelope Acquisition mode.

points_arg ::= 32, 64, 128, 256, 500, 512, or 1024.

Figure 8-1. Acquire Subsystem Commands Syntax Diagram

COMPlete

command/query

The :ACQUIRE:COMPLETE command specifies the completion criteria for an acquisition. 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 bit 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 modes a specified number of data points (COUNT) must be acquired.

The range for the COMPLETE command is 0 to 100 and indicates the 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 0 then one acquisition cycle will take place.

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

Command Syntax: :ACQuire:COMPlete <comp>

Where:

<comp> ::= 0 to 100 percent

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

Query Syntax: :ACQuire:COMPlete?

Returned Format: [:ACQuire:COMPlete] <comp><NL>

Where:

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

Example:
```
DIM Cmp$[50]
OUTPUT 707;":ACQUIRE:COMPLETE?"
ENTER 707;Cmp$
PRINT Cmp$
```

COUNt command/query

In 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.

When acquisition type is set to NORMAL the count is 1.

When the acquisition type is set to AVERAGE the count can range from 1 to 2048. Any value can be sent, however the value will be rounded to the nearest power of 2.

When the acquisition type is set to ENVELOPE the count can be any value between 1 and 2048.

The COUNT query returns the currently selected count value.

Command Syntax: :ACQuire:COUNt <count>

Where:

<count> ::= 1 to 2048

Example:
```
OUTPUT 707;":ACQUIRE:TYPE AVERAGE;COUNT 1024"
```

Query Syntax: :ACQuire:COUNt?

Returned Format: [:ACQuire:COUNt] <count><NL>

Where:

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

Example:
```
DIM Cnt$[50]
OUTPUT 707;":ACQ:COUNT?"
ENTER 707;Cnt$
PRINT Cnt$
```

POINts

command/query

The :ACQUIRE:POINTS command specifies the number of time buckets for each acquisition record. The legal settings are 32, 64, 128, 256, 500, 512, or 1024.

Any value between 32 and 1024 can be sent to the instrument. If a value is sent that is not one of the legal values it is rounded to the nearest power of 2. If a number smaller than 31 or greater than 1024 is sent an error is produced.

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

Note

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

Command Syntax: :ACQuire:POINts <points_arg>

Where:

<points_arg> ::= 32 to 1024 (see above for legal values)

Example:
```
OUTPUT 707;":ACQ:POINTS 512"
```

Query Syntax: :ACQuire:POINts?

Returned Format: [:ACQuire:POINts] <points_arg> <NL>

Where:

<points_arg> ::= 32 - 1024 (see above for legal values)

Example:
```
DIM Pnts$[50]
OUTPUT 707;":ACQUIRE:POINTS?"
ENTER 707;Pnts$
PRINT Pnts$
```

TYPE

command/query

The :ACQUIRE:TYPE command selects the type of acquisition that is to take place when a :DIGITIZE root level command is executed. There are three acquisition types: NORMAL, AVERAGE, and ENVELOPE.

The :ACQUIRE:TYPE query returns the current acquisition type.

Command Syntax: :ACQuire:TYPE {NORMal | AVERage | ENVelope}

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

Query Syntax: :ACQuire:TYPE?

Returned Format: [:ACQuire:TYPE] <type> <NL>

Where:

<type> ::= {NORMal | AVERage | ENVelope}

Example:
```
DIM Tpe$[50]
OUTPUT 707;":ACQUIRE:TYPE?"
ENTER 707;Tpe$
PRINT Tpe$
```

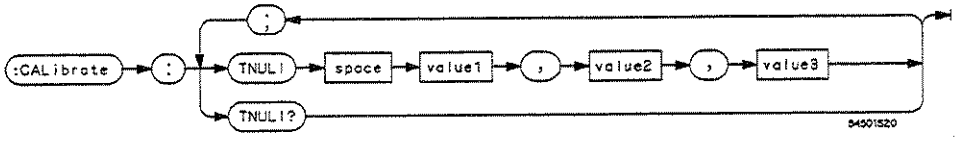
Calibrate Subsystem 9

Introduction

The CALIBRATE subsystem contains only one command. This subsystem calibrates the instrument for different probes, cables, or setups.

Note

The time null is set in the Probe Cal menu of the Utility menus. For more information refer to the HP 54501A Front-Panel Reference Manual.



value1 = channel 1 to channel 2 skew

value2 = channel 1 to channel 3 skew

value3 = channel 1 to channel 4 skew

Figure 9-1. Calibrate Subsystem Commands Syntax Diagram

TNULI command/query

The :CALIBRATE:TNULL command sends the time null (channel-to-channel skew) values into the HP 54501A. The time null values should have been obtained from the instrument during a previous setup.

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

Command Syntax: :CALibrate:TNULI <null_value1>,<null_value2>,<null_value3>

Where:

<null_value1> ::= channel 1 to channel 2 skew
<null_value2> ::= channel 1 to channel 3 skew
<null_value3> ::= channel 1 to channel 4 skew

Example:
```
OUTPUT 707;":CAL:TNUL <null_value1>,<null_value2>,<null_value3>
```

Query Syntax: :CALibrate:TNULI?

Returned Format: [:CALibrate:TNULI] <null_value1>,<null_value2>,<null_value3><NL>

Where:

<null_value1> ::= channel 1 to channel 2 skew (exponential - NR3 format)
<null_value2> ::= channel 1 to channel 3 skew (exponential - NR3 format)
<null_value3> ::= channel 1 to channel 4 skew (exponential - NR3 format)

Example:
```
DIM Nil$[50]
OUTPUT 707;":CALIBRATE:TNULL?"
ENTER 707;Nil$
PRINT Nil$
```

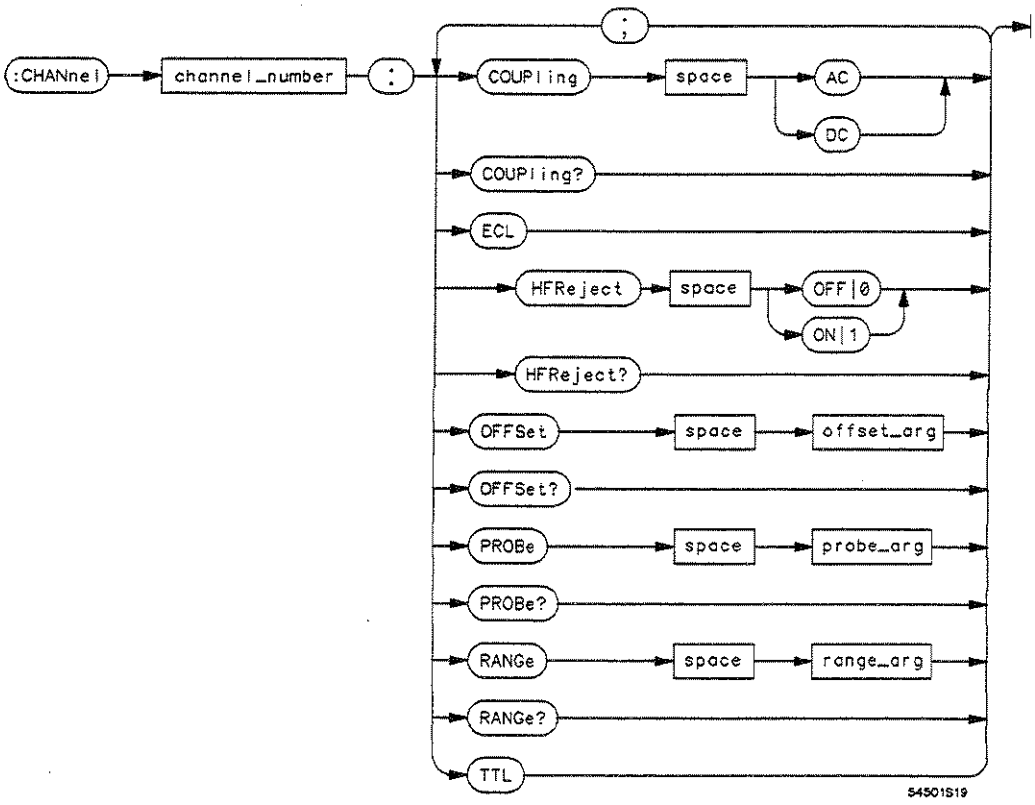
Channel Subsystem 10

Introduction

The CHANNEL subsystem commands control the channel display and vertical or Y axis of the HP 54501A. Channels 1, 2, 3, and 4 are independently programmable for all offset, probe, coupling, and range functions.

The channel commands can be sent with a channel number specified or not specifed. If a channel number is specified in the command then the specified channel is affected, however if the channel number is **not** specified then channel 1 is affected.

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



channel_number = 1, 2, 3, or 4

offset_arg = A real number defining the voltage at the center of the display range.

probe_arg = A real number from 0.9 to 1000.0 specifying the probe attenuation with respect to 1.

range_arg = A real number specifying the size of the acquisition window in volts.

Figure 10-1. Channel Subsystem Commands Syntax Diagram

COUPling

command/query

The :CHANNEL<N>:COUPLING command selects the input coupling for the specified channel. The coupling for each channel can be set to AC or DC.

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

Command Syntax: :CHANnel<N>:COUPling {AC | DC}

Where:

<N> ::= 1, 2, 3, or 4

Example: OUTPUT 707;":CHAN2:COUP DC"

Query Syntax :CHANnel<N>:COUPling?

Where:

<N> ::= 1, 2, 3, or 4

Returned Format: [:CHANnel<N>:COUPling] {AC | DC}<NL>

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

ECL command

The :CHANNEL<N>:ECL command sets the vertical range, offset, channel coupling, and trigger level of the selected channel for optimum viewing of ECL signals. The offset and trigger level are set to - 1.3 volts and the range is set to 12.8 volts full scale. Channel coupling is set to DC.

There is no query form of this command.

Command Syntax: :CHANnel<N>:ECL

Example: OUTPUT 707;":CHAN1:ECL"

Where:

<N> ::= 1, 2, 3, or 4

HFReject command/query

The :CHANNEL<N>:HFREJECT command controls an internal lowpass filter. When ON the bandwidth of the specified channel is limited to approximately 20 MHz. The bandwidth limit filter may be used when either AC or DC coupling is used.

The HFREJECT query returns the current setting.

Command Syntax: :CHANnel<N>:HFReject {{ON | 1} | {OFF | 0}}

Where:

<N> ::= 1, 2, 3, or 4

Example:
```
OUTPUT 707;":CHANNEL2:HFR ON"
```

Query Syntax: :CHANnel<N>:HFReject?

Where:

<N> ::= 1, 2, 3, or 4

Returned Format: [:CHANnel<N>:HFReject] {1 | 0}<NL>

Example:
```
DIM Hf$[50]
OUTPUT 707;":CHAN:HFR?"
ENTER 707;Hf$
PRINT Hf$
```

OFFSet command/query

The :CHANNEL<N>:OFFSET command sets the voltage that is represented at center screen for the selected channel. The range of legal values varies with the value set with the RANGE command. If you set the offset to a value outside the legal range, it will automatically be set to the nearest legal value.

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

Command Syntax: :CHANnel<N>:OFFSet <value>

Where:

<N> ::= 1, 2, 3, or 4
<value> ::= offset value

Example: OUTPUT 707;":CHAN1:OFFS 200M;:CHAN2:OFFSET 20E-3"

Query Syntax: :CHANnel<N>:OFFSet?

Returned Format: [:CHANnel<N>:OFFSet] <value><NL>

Where:

<N> ::= 1, 2, 3, or 4
<value> ::= offset value in volts (exponential - NR3 format)

Example:
```
OUTPUT 707;":CHANNEL2:OFFSET?"
ENTER 707;Offset
PRINT Offset
```

PROBe

command/query

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 HP 54501A. It changes the reference constants for scaling the display factors and for automatic measurements, trigger levels, etc.

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

Command Syntax: :CHANnel<N>:PROBe <atten>

Where:

<N> ::= 1, 2, 3, or 4
<atten> ::= 0.9 to 1000

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

Query Syntax: :CHANnel<N>:PROBe?

Where:

<N> ::= 1, 2, 3, or 4

Returned Format: [:CHANnel<N>:PROBe] <atten><NL>

Where:

<N> ::= 1, 2, 3, or 4
<atten> ::= 0.9 to 1000 (exponential - NR3 format)

Example:
```
DIM Prb$[50]
OUTPUT 707;":CHANNEL1:PROBE?"
ENTER 707;Prb$
PRINT Prb$
```

RANGe

command/query

The :CHANNEL<N>:RANGE command defines the full scale vertical axis of the selected channel. The RANGE for channels 1 and 4 can be set to any value from 40 mV to 40 V, when using 1:1 probe attenuation. The RANGE for channels 2 and 3 can be set to any value from 800 mV to 4.0 V, when using 1:1 probe attenuation. If the probe attenuation is changed, the range value is multiplied by the probe attenuation factor.

The RANGE query returns the current range setting for the specified channel.

Command Syntax: :CHANnel<N>:RANGe <range>

Where:

<N> ::= 1, 2, 3, or 4
<range> ::= range value

Examples: :OUTPUT 707;"CHANNEL1:RANGE .60"

:OUTPUT 707;"CHANNEL2:RANGE 1.2 V"

Query Syntax: :CHANnel<N>:RANGe?

Returned Format: [:CHANnel<N>:RANGe] <range><NL>

Where:

<N> ::= 1, 2, 3, or 4
<range> ::= range value (exponential - NR3 format)

Example:
```
DIM Rng$[50]
OUTPUT 707;":CHAN2:RANGE?"
ENTER 707;Rng$
PRINT Rng$
```

TTL command

The :CHANNEL<N>:TTL command sets the vertical range, offset, channel coupling, and trigger level of the selected channel for optimum viewing of TTL signals. The offset 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.

There is no query form of this command.

Command Syntax: :CHANnel<N>:TTL

Example: OUTPUT 707;":CHAN1:TTL"

Where:

<N> ::= 1, 2, 3, or 4

Display Subsystem 11

Introduction

The DISPLAY subsystem is used to control the display of data, voltage and time markers, text, and graticules.

Note

The command that changes the Display mode is :ACQUIRE:TYPE. The command that controls the number of averages is ACQUIRE:COUNT.

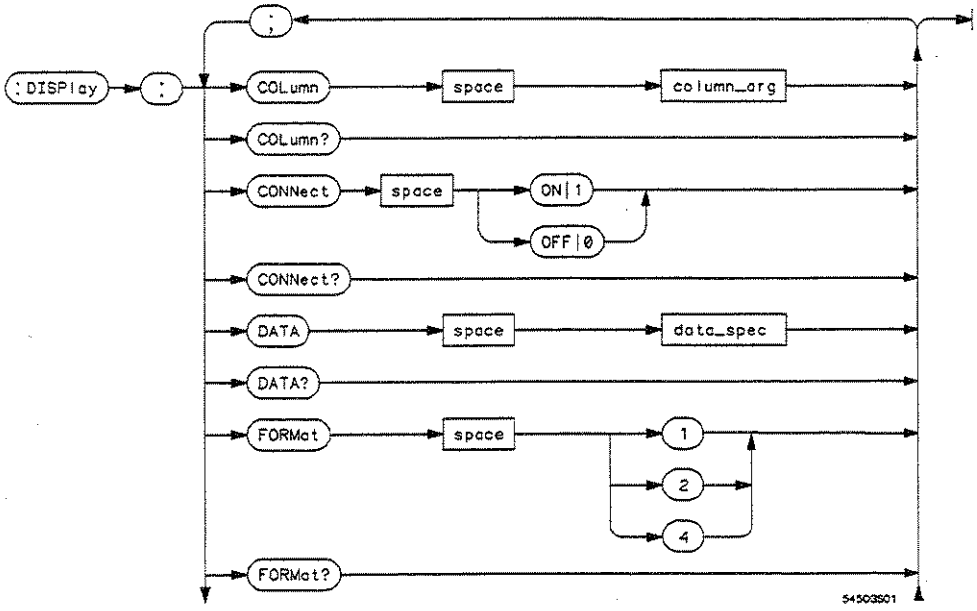


Figure 11-1. Display Subsystem Commands Syntax Diagram

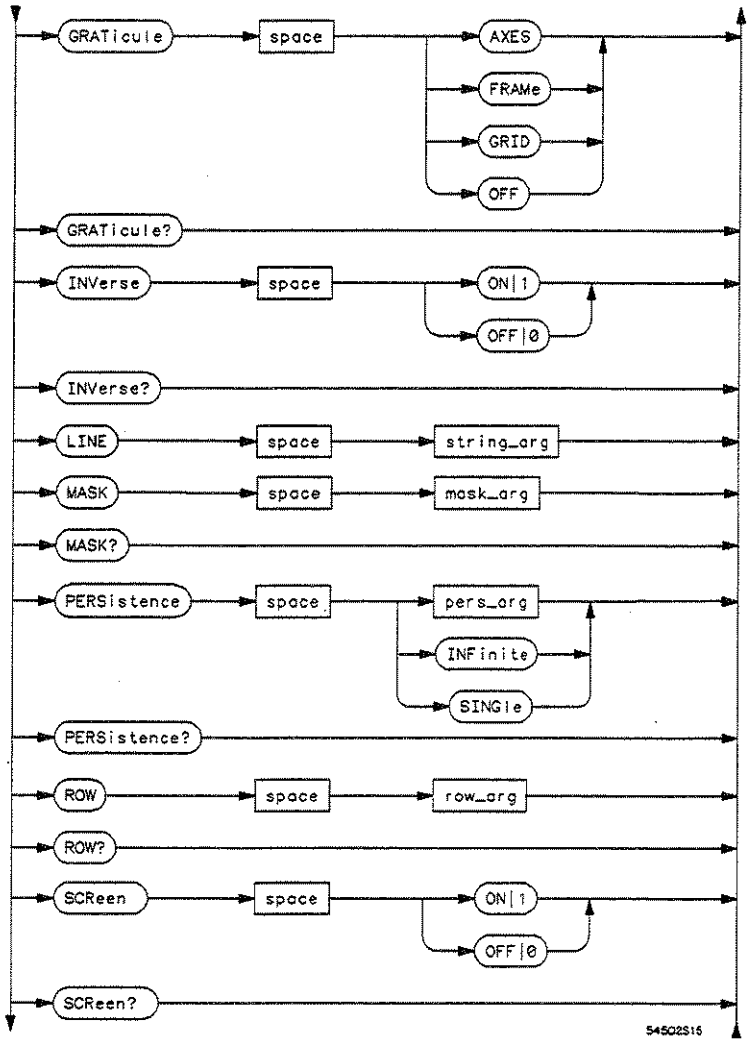
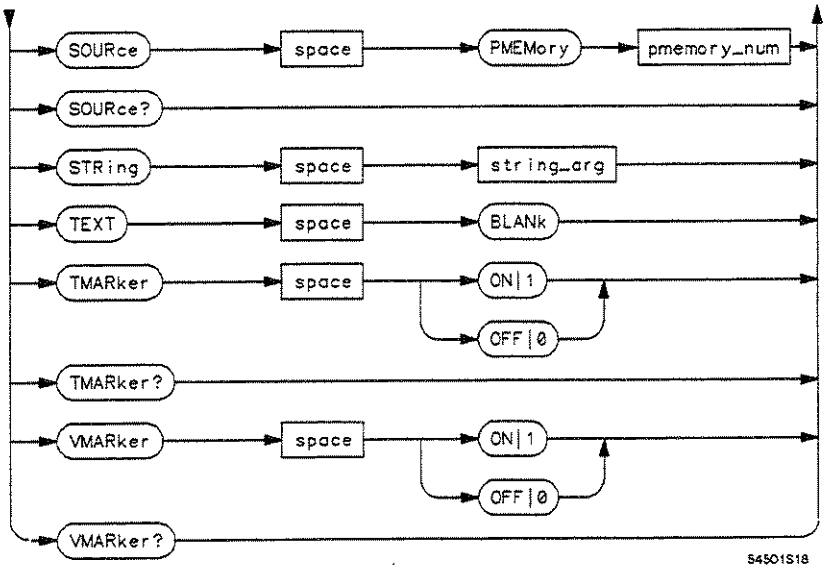


Figure 11-1. Display Subsystem Commands Syntax Diagram (continued)



column_arg = An integer from 0 through 72.

data_spec = Block data in IEEE 488.2 # format.

mask_arg = An integer, 0 through 255.

pers_arg = An real number 0.1 through 11.

pmemory_num = An integer, 0 through 3.

row_arg = An integer, 0 through 24.

string_arg = Any quoted string.

Figure 11-1. Display Subsystem Commands Syntax Diagram (continued)

COLumn

command/query

The :DISPLAY:COLUMN command specifies the starting column for subsequent STRING and LINE commands.

The COLUMN query returns the column where the next LINE or STRING will start.

Command Syntax: :DISPlay:COLumn <number>

Where:

<number> ::= 0 through 72

Example:
```
OUTPUT 707;":DISPLAY:COLUMN 50"
```

Query Syntax: :DISPlay:COLumn?

Returned Format: [:DISPlay:COLumn] <number> <NL>

Where:

<number> ::= 0 through 72 (integer - NR1 format)

Example:
```
DIM Clmn$[30]
OUTPUT 707;":DISPLAY:COLUMN?"
ENTER 707;Clmn$
PRINT Clmn$
```

CONNect — command/query

The :DISPLAY:CONNECT command turns the connect-the-dots function on and off.

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.

Command Syntax: :DISPlay:CONNect {{ON | 1} | {OFF | 0}}

Example:
```
OUTPUT 707;":DISPLAY:CONNECT ON"
```

Query Syntax: :DISPlay:CONNect?

Returned Format: [:DISPlay:CONNect] {1 | 0}<NL>

Example:
```
DIM Cnn$[50]
OUTPUT 707;":DISP:CONNECT?"
ENTER 707;Cnn$
PRINT Cnn$
```

DATA

command/query

The :DISPLAY:DATA command is used to write waveform data to one of the pixel planes in the HP 54501A. The DATA command is followed by a block of binary data that is transferred from the controller to a specific plane in the HP 54501A. Only pixel memories 1 and 2 may be written to. The data is in the IEEE 488.2 definite block form with 16576 bytes of data preceded by seven block header bytes. The block header contains the ASCII characters "#800016576" and is sent prior to the data being sent.

The :DISPLAY:DATA query is used to write waveform data from one of the pixel planes in the HP 54501A. The pixel planes available are planes 0 through 3. The DATA query causes the HP 54501A to output pixel data from the specified plane. If plane 0 is specified, the HP 54501A will transfer the active display. If PMEMORY1 or PMEMORY2 is specified that memory will be transferred. When PMEMORY3 is specifed the half-bright portion of the display (graticule, markers, and displayed memories) will be transferred.

Note

The pixel planes are specified by the :DISPLAY:SOURCE command with PMEMORY0 through PMEMORY3.

Command Syntax: :DISPlay:DATA <binary block>

Where:

<binary block> ::= block data in IEEE 488.2 # format

Query Syntax: :DISPlay:DATA?

Returned Format: [:DISPlay:DATA] #800016576<16576 bytes of binary data><NL>

Example:

```
10  CLEAR 707
20  DIM Plane$ [17000]
30  OUTPUT 707;":SYST:HEAD ON;:EOI ON"
40  OUTPUT 707;":DISPLAY:SOURCE PMEMORY0;DATA?"
50  ENTER 707 USING "-K";Plane$
60  OUTPUT 707;":DISPLAY:SOURCE PMEM1"
70  OUTPUT 707 USING"#,-K";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 HP 54501A.

FORMat command/query

The :DISPLAY:FORMAT command sets the number of display areas on the CRT. 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 four divisions for the full scale range. FORMAT 4 provides four display areas on the CRT and uses two divisions for the full scale range.

The FORMAT query returns the current display format.

Command Syntax: :DISPlay:FORMat {1 | 2 | 4}

Example: OUTPUT 707;":DISP:FORMAT 1"

Query Syntax: :DISPlay:FORMat?

Returned Format: [:DISPlay:FORMat] {1 | 2 | 4}<NL>

Example:
```
DIM Frmt$[30]
OUTPUT 707;":DISPLAY:FORMAT?"
ENTER 707;Frmt$
PRINT Frmt$
```

GRATicule command/query

The :DISPLAY:GRATICULE command selects the type of graticule that is displayed.

The GRATICULE query returns the type of graticule displayed.

Command Syntax: :DISPlay:GRATicule {OFF | GRID | AXES | FRAMe}

Example:
```
OUTPUT 707;":DISPLAY:GRATICULE AXES"
```

Query Syntax: :DISPlay:GRATicule?

Returned Format: [:DISPlay:GRATicule] <type> <NL>

Where:

<type> ::= {OFF | GRID | AXES | FRAMe}

Example:
```
DIM Grt$[30]
OUTPUT 707;":DISPLAY:GRATICULE?"
ENTER 707;Grt$
PRINT Grt$
```

INVerse command/query

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.

The INVERSE query returns the current state of this command.

Command Syntax: :DISPlay:INVerse {{ON | 1} | {OFF | 0}}

Example:
```
OUTPUT 707;":DISPLAY:INVERSE OFF"
```

Query Syntax: :DISPlay:INVerse?

Returned format: [:DISPlay:INVerse] {1 | 0}<NL>

Example:
```
DIM iv$[30]
OUTPUT 707;":DISP:INVERSE?"
ENTER 707;iv$
PRINT iv$
```

LINE command

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 will wrap 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.

Command Syntax: :DISPlay:LINE <quoted string>

Where:

<quoted string> ::= any series of ASCII characters enclosed in quotes.

Example: OUTPUT 707:":DISPLAY:LINE ""ENTER PROBE ATTENUATION"""

MASK

command/query

The :DISPLAY:MASK command inhibits the instrument from writing to selected areas of the screen. Text sent over the HP-IB 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 HP-IB text to be written anywhere on 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.

The MASK query returns the current value of the MASK.

Command Syntax: :DISPlay:MASK <value>

Where:

<value> ::= 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 Syntax: :DISPlay:MASK?

Return Format: [:DISPlay:MASK] <value><NL>

Where:

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

Example:
```
DIM Msk$[30]
OUTPUT 707;":DISP:MASK?"
ENTER 707;Msk$
PRINT Msk$
```

Table 11-1. Display Mask Byte.

Bit	Weight	Screen Area Effected
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

PERSistence command/query

The :DISPLAY:PERSISTENCE command sets the display persistence. The PERSISTENCE command is only effective in the Normal display mode.

The parameters for this command are the keywords INFINITE, or SINGLE, or a real numbers from 0.15 through 11.0 representing the persistence in seconds. Any value less than 0.15 will set the PERSISTENCE to MINIMUM. Any value greater than 10 seconds will set the PERSISTENCE to INFINITE.

When the key word SINGLE is sent as the argument for this command, the persistence value is set to minimum.

The PERSISTENCE query returns the current persistence value. When Minimum is displayed, the value returned is 0. When Infinite is displayed, the value returned is 11.

Command Syntax: :DISPlay:PERSistence {INFinite | SINGle | 0.1 through 11}

Example:
```
OUTPUT 707;":DISPLAY:PERSISTENCE 3.0"
```

Query Syntax: :DISPlay:PERSistence?

Returned Format: [:DISPlay:PERSistence] <value> <NL>

Where:

<value> ::= {0 | .2 - 10 | 11} (exponential - NR3 format)
where 1.100E+1 = infinite

Example:
```
DIM Prs$[50]
OUTPUT 707;":DISPLAY:PERSISTENCE?"
ENTER 707;Prs$
PRINT Prs$
```

ROW

command/query

The :DISPLAY:ROW command specifies the starting row on the CRT for subsequent STRING and LINE commands. The ROW number remains constant until another ROW command is received, or it is incremented by the LINE command. The ROW value is 0 through 24.

The ROW query returns the current value of ROW.

Command Syntax: :DISPlay:ROW <row number>

Where:

<row number> ::= 0 through 24

Example:
```
OUTPUT 707;":DISPLAY:ROW 10"
```

Query Syntax: :DISPLAY:ROW?

Returned Format: [:DISPlay:ROW] <row number> <NL>

Where:

<row number> ::= 0 through 24 (integer - NR1 format)

Example:
```
DIM Rw$[30]
OUTPUT 707;":DISPLAY:ROW?"
ENTER 707;Rw$
PRINT Rw$
```

SCReen command/query

The :DISPLAY:SCREEN command turns the displayed screen on and off. The only part of the screen that remains on after the :DISPLAY:SCREEN OFF command is executed is the status line. The screen can be turned on again with the ON parameter.

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.

The command :DISPLAY:TEXT BLANK removes only the text from the display.

Command Syntax: :DISPlay:SCReen {{ON | 1} | {OFF | 0}}

Example:
```
OUTPUT 707;":DISPLAY:SCREEN ON"
```

Query Syntax: :DISPlay:SCReen?

Returned Format: [:DISPlay:SCReen] {1 | 0}<NL>

Example:
```
DIM Srn$[50]
OUTPUT 707;":DISP:SCREEN?"
ENTER 707;Srn$
PRINT Srn$
```

SOURce command/query

The :DISPLAY:SOURCE command specifies the source or destination for the :DISPLAY:DATA query and command. The SOURCE command has one parameter, PMEMORY0 through PMEMORY3.

The SOURCE query returns the currently specified SOURCE.

Command Syntax: :DISPlay:SOURce PMEMory{0 | 1 | 2 | 3}

Where:

PMEMory0 ::= active display
PMEMory1 ::= pixel memory 1
PMEMory2 ::= pixel memory 2
PMEMory3 ::= half-bright portion of the display (graticule, markers, and displayed memories)

Example:

```
10  CLEAR 707
20  DIM Plane$ [17000]
30  OUTPUT 707;":SYST:HEAD ON;:EOI ON"
40  OUTPUT 707;":DISPLAY:SOURCE PMEMORY0;DATA?"
50  ENTER 707 USING "-K";Plane$
60  OUTPUT 707;"DISPLAY:SOURCE PMEM1"
70  OUTPUT 707 USING "#,-K";Plane$
80  END
```

This example transfers data from the active display to the controller and then back to pixel memory 1 in the HP 54501A.

Query Syntax: :DISPlay:SOURce?

Returned Format: [:DISPlay:SOURce] PMEMory{0 | 1 | 2 | 3}<NL>

Example:

```
DIM Src$[30]
OUTPUT 707;":DISP:SOUR?"
ENTER 707;Src$
PRINT Src$
```

STRing command

The :DISPLAY:STRING command writes a text string to the CRT of the HP 54501A. The text will be written starting at the current ROW and COLUMN values. If the column limit is reached (column 72) the excess text is written over the text on the left side of that line. If 90 or more characters are sent an error is produced. The STRING command does not increment the ROW value, however the LINE command does.

Command Syntax: :DISPlay:STRing <quoted string>

Example: OUTPUT 707;":DISP:STRING 'INPUT SIGNAL TO CHANNEL 2'"

TEXT command

The :DISPLAY:TEXT command allows you to blank the user text area on the CRT. All text on the entire screen will be blanked. This command has only one parameter.

There is no query form of this command.

Command Syntax: :DISPlay:TEXT BLANk

Example: OUTPUT 707;":DISPLAY:TEXT BLAN"

TMARker command/query

The :DISPLAY:TMARKER command turns the time markers on and off.

The TMARKER query returns the state of the time markers.

Command Syntax: :DISPlay:TMARker {{ON | 1} | {OFF | 0}}

Example:
```
OUTPUT 707;":DISP:TMAR OFF"
```

Query Syntax: :DISPlay:TMARker?

Returned Format: [:DISPlay:TMARker] {1 | 0}<NL>

Example:
```
DIM Tmr$[30]
OUTPUT 707;":DISP:TMARKER?"
ENTER 707;Tmr$
PRINT Tmr$
```

Note

It is a recommended practice to turn the Tmarkers on before attempting to set them using a :MEASURE:TSTART or MEASURE:TSTOP command.

VMARker command/query

The :DISPLAY:VMARKER command turns the voltage markers on and off.

The VMARKER query returns the state of the Vmarkers.

Command Syntax: :DISPlay:VMARker {{ON | 1} | {OFF | 0}}

Example:
```
OUTPUT 707;":DISP:VMARKER ON"
```

Query Syntax: :DISPlay:VMARker?

Returned Format: [:DISPlay:VMARker] {1 | 0}<NL>

Example:
```
DIM Vmrk$[30]
OUTPUT 707;":DISP:VMARKER?"
ENTER 707;Vmrk$
PRINT Vmrk$
```

Note

It is a recommended practice to turn the Vmarkers on before attempting to set them using a :MEASURE:VSTART or MEASURE:VSTOP command.

Function Subsystem 12

Introduction

The FUNCTION subsystem defines six functions using the displayed channels and/or the waveform memories as operands. The operators are ADD, SUBTRACT, MULTIPLY, VERSUS, ONLY, and INVERT.

If a channel or memory that is not on is specified as an operand, then that channel is enabled.

See figure 12-1 for a syntax diagram of the Function subsystem commands.

Channel 1 through 4 and waveform Memories 1 through 4 are available for functions.

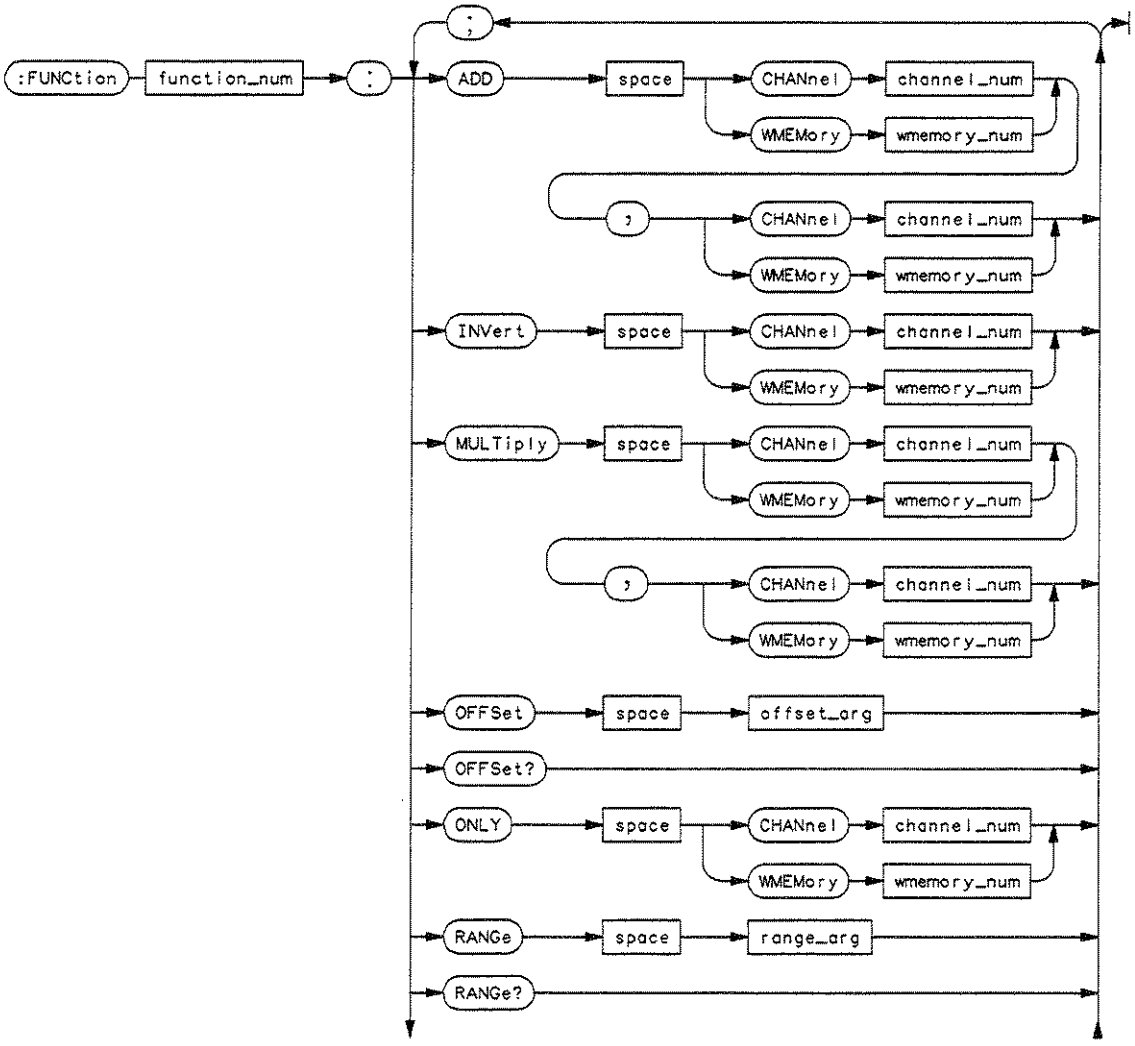
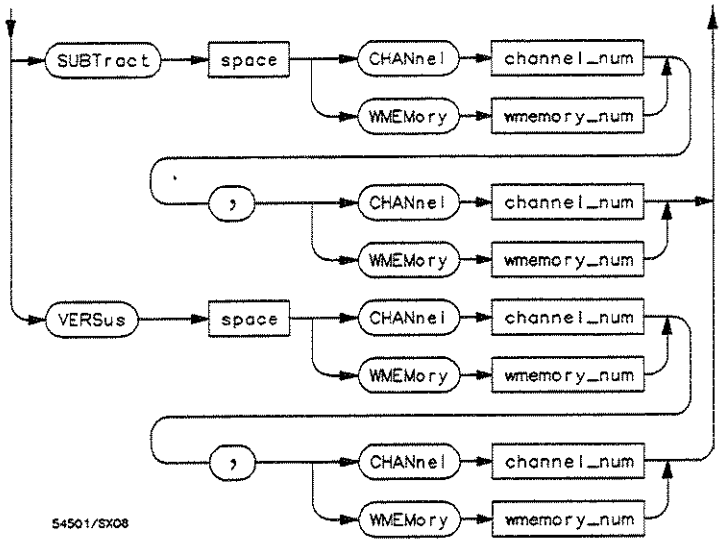


Figure 12-1. Function Subsystem Commands Syntax Diagram



channel_num = 1, 2, 3, or 4

function_num = 1 or 2

offset_arg = 0 to ± voltage full scale

range_arg = full screen voltage

wmemory_num = 1, 2, 3, or 4

Figure 12-1. Function Subsystem Commands Syntax Diagram (continued)

ADD command

The :FUNCTION<N>:ADD command algebraically sums the two defined operands.

Command Syntax: :FUNCtion<N>:ADD <operand>,<operand>

Where:

<N> ::= 1 or 2

<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example: OUTPUT 707;":FUNCTION2:ADD WMEMORY3,WMEMORY4"

INVert command

The :FUNCTION<N>:INVERT command inverts the operand.

Command Syntax: :FUNCtion<N>:INVert <operand>

Where:

<N> ::= 1 or 2

<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example: OUTPUT 707;":FUNCTION2:INVERT WMEMORY3"

MULTiply command

The :FUNCTION<N>:MULTIPLY command causes the instrument to algebraically multiply the two operands.

Command Syntax: :FUNCtion<N>:MULTiply <operand>,<operand>

Where:

<N> ::= 1 or 2
<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example: OUTPUT 707;":FUNCTION2:MULTIPLY CHANNEL1,CHANNEL2"

OFFSet

command/query

The :FUNCTION<N>:OFFSET command sets the voltage represented at center screen for the selected function. The maximum value of offset is +/- volts full screen.

The OFFSET query returns the current offset value for the selected function.

Command Syntax: :FUNCtion<N>:OFFSet <offset>

Where:

<N> ::= 1 or 2
<offset> ::= offset value (see above)

Example:
```
OUTPUT 707;":FUNCTION1:OFFSET 650E-4"
```

Query Syntax: :FUNCtion<N>:OFFSet?

Where:

<N> ::= 1 or 2

Returned Format: [:FUNCtion<N>:OFFSet] <offset><NL>

Where:

<N> ::= 1 or 2
<offset> ::= offset value (see above) (exponential - NR3 format)

Example:
```
DIM Off$[50]
OUTPUT 707;":FUNCTION2:OFFSET?"
ENTER 707;Off$
PRINT Off$
```

ONLY command

The :FUNCTION<N>:ONLY command is just another copy of the operand. The ONLY command is useful for scaling channels and memories with the :FUNCTION<N>:RANGE and :FUNCTION<N>:OFFSET commands.

Command Syntax: :FUNCtion<N>:ONLY<operand>

Where:

<N> ::= 1 or 2
<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example: OUTPUT 707;":FUNCTION2:ONLY WMEMORY4"

RANGe — command/query

The :FUNCTION<N>:RANGE command defines the full scale vertical axis of the selected function.

The RANGE query returns the current range setting for the specified function.

Command Syntax: :FUNCtion<N>:RANGe <range>

Where:

<N> ::= 1 or 2
<range> ::= voltage value

Example:
```
OUTPUT 707;":FUNCTION2:RANGE 400 MV"
```

Query Syntax: :FUNCtion<N>:RANGe?

Where:

<N> ::= 1 or 2

Returned Format: [:FUNCtion<N>:RANGe] <range><NL>

Where:

<N> ::= 1 or 2
<range> ::= current range setting (exponential - NR3 format)

Example:
```
DIM Rng$[50]
OUTPUT 707;":FUNCTION2:RANGE?"
ENTER 707;Rng$
PRINT Rng$
```

SUBTract command

The :FUNCTION<N>:SUBTRACT command algebraically subtracts operand 2 from operand 1.

Command Syntax: :FUNCtion<N>:SUBTract <operand>,<operand>

Where:

<N> ::= 1 or 2
<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example: OUTPUT 707;":FUNCTION2:SUBTRACT WMEMORY3,WMEMORY2"

In this example Waveform Memory 2 would be algebraically subtracted from Waveform Memory 3.

VERSus command

The :FUNCTION<N>: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<N>:RANGE and FUNCTION<N>:OFFSET commands.

The X axis range and offset is always equal to that of the second operand. It can only be changed by changing the vertical settings of the second operand. This will also change the Y axis vertical sensitivity and offset.

Command Syntax: :FUNCtion<N>:VERSus <Y_operand>,<X_operand>

Where:

<N> ::= 1 or 2

<Y_operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

<X_operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

Example: OUTPUT 707;":FUNCTION2:VERSUS CHAN1,CHAN2"

Hardcopy Subsystem 13

Introduction

The HARDCOPY subsystem commands set various parameters for printing waveforms from the HP 54501A. Everything on the display is printed when the root level command PRINT is sent.

The portion of the waveform to be copied must be placed on the display.

To actually make the hardcopy print, refer to the root level command :PRINT for the sequence of bus commands that actually get the data to the printer.

Refer to figure 13-1 for the syntax diagram of the Hardcopy subsystem commands.

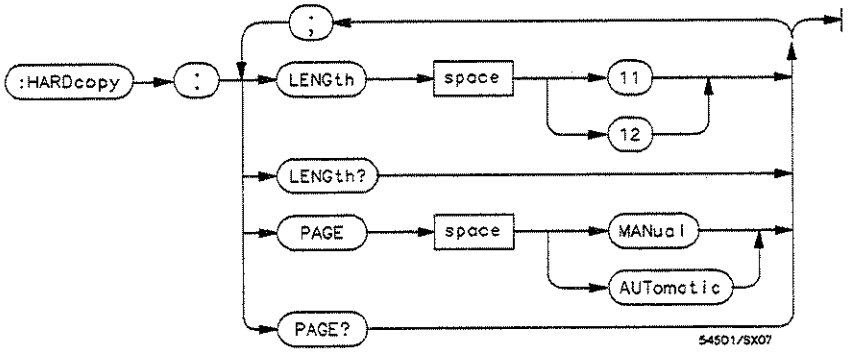


Figure 13-1. Hardcopy Subsystem Commands Syntax Diagram

LENGth

command/query

The :HARDCOPY:LENGTH command sets the length of the page to either 11 inches or 12 inches.

The LENGTH query returns the current length setting.

Command Syntax: :HARDcopy:LENGth {11 | 12}

Example:
```
OUTPUT 707;":HARDCOPY:LENGTH 12"
```

Query Syntax: :HARDcopy:LENGth?

Returned Format: [:HARDcopy:LENGth] {11 | 12}<NL>

Example:
```
OUTPUT 707;":HARDCOPY:LENGTH?"
ENTER 707;Lgth$
PRINT Lgth$
```

PAGE

command/query

The :HARDCOPY:PAGE command sets the HP 54501A to send a formfeed to the printer after a hardcopy output.

If the PAGE command is set to AUTomatic, a formfeed will occur at the end of the hardcopy, otherwise the page will scroll up by 4 lines.

The PAGE query returns the current state of the page command.

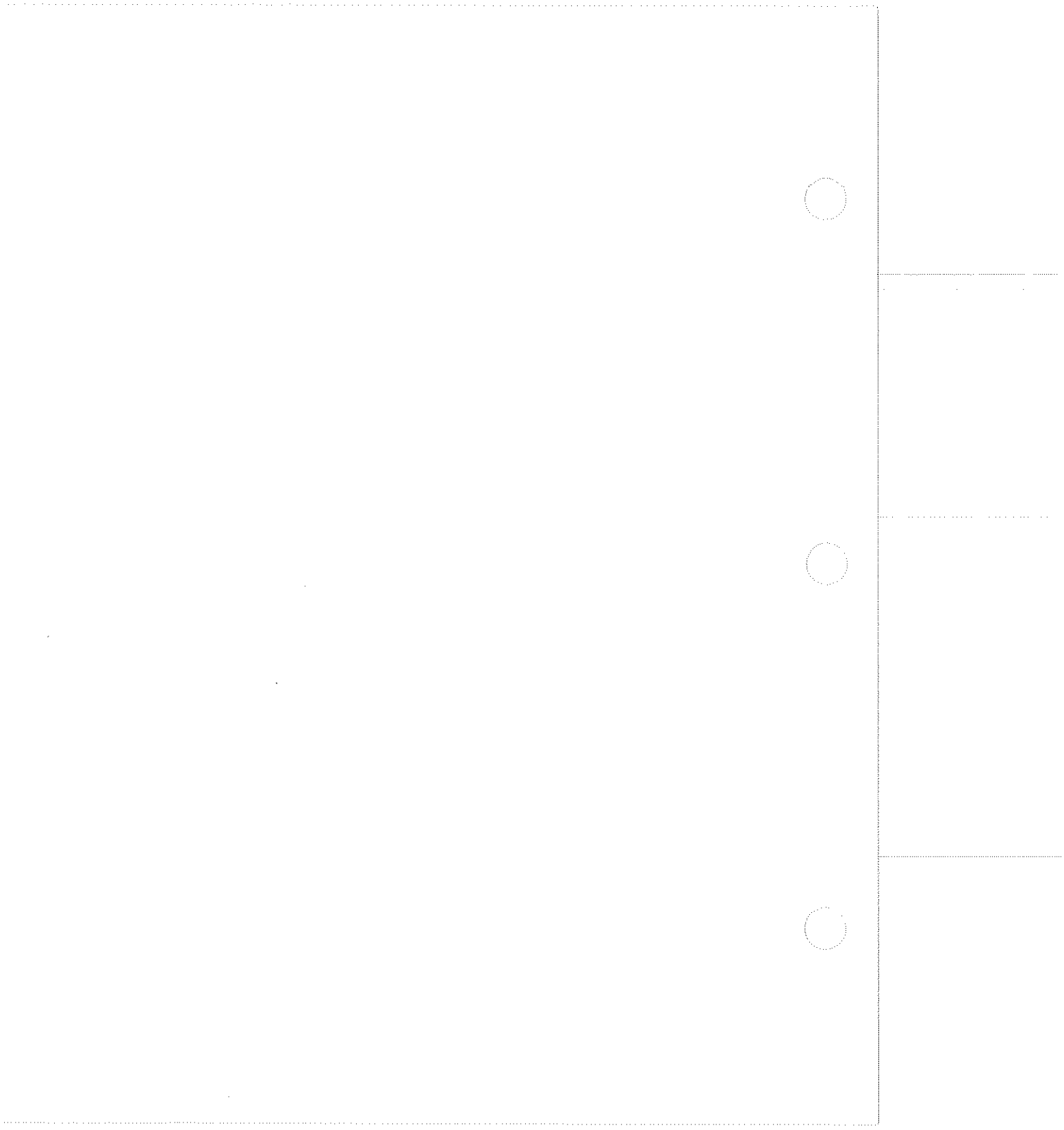
Command Syntax: :HARDcopy:PAGE {MANual | AUTomatic}

Example:
```
OUTPUT 707;":HARD:PAGE AUT"
```

Query Syntax: :HARDcopy:PAGE?

Returned Format: [:HARDcopy:PAGE] {MANual | AUTomatic} <NL>

Example:
```
DIM Pg$[30]
OUTPUT 707;":HARDCOPY:PAGE?"
ENTER 707;Pg$
PRINT Pg$
```



Measure Subsystem 14

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.

Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must be displayed on the oscilloscope. That is:

- 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 risetime measurement, the leading (positive-going) edge of the waveform must be displayed;
- For a falltime measurement, the trailing (negative-going) edge of the waveform must be displayed.

Note

When WINDOW is ON, measurements are ONLY applied to the windowed portion of the waveform.

User-Defined Measurements

When User-Defined Measurements are made, the defined parameters must be set before actually sending the measurement command or query.

In User-Defined the mid threshold is the mid point between the upper and lower threshold when the lower threshold value is less than the upper threshold value.

Measurement Error

If a measurement cannot be made, typically because the proper portion of the waveform is not displayed, the value returned for that parameter is +9.99999E+37. This is an error value that is output when a measurement cannot be made.

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 that can be used. 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% voltage, 90% voltage, and 50% voltage) for making the measurements. The 10% and 90% voltage values are used in the risetime and falltime 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 front-panel measurements, preshoot, or overshoot is used the instrument is placed into the continuous measurement mode. When the query form of these measurements is used continuous measurement mode is turned off, the measurement is made one time and the measurement result is returned.

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 display.

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 VMarker 1 and VMarker 2 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 stop and start markers.

Measurements are made on the displayed waveform(s) specified by the SOURCE command. The SOURCE command allows two sources to be specified. When two sources are specified, VMarker 1 is assigned to the first specified source and VMarker 2 is assigned to the second specified source. VDELTA is the only measurement that uses two sources.

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.

More information about measurement algorithms can be found in Appendix A.

If the horizontal scaling is questionable, an "error 11" is placed in the error queue. In this case the value returned is the most accurate 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.

Refer to figure 14-1 for the syntax diagram of the Measure subsystem commands.

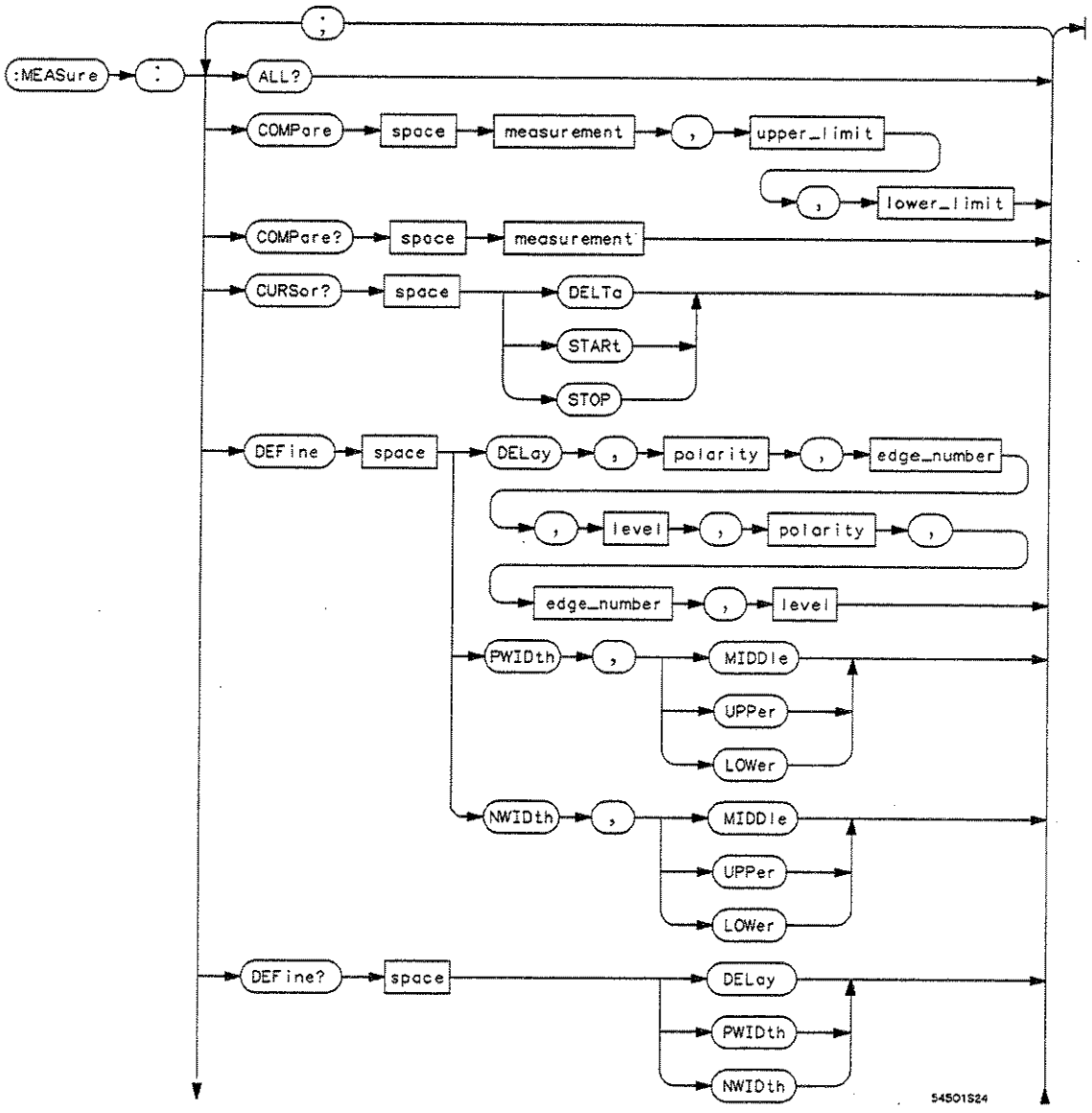


Figure 14-1 Measure Subsystem CommandsSyntax Diagram

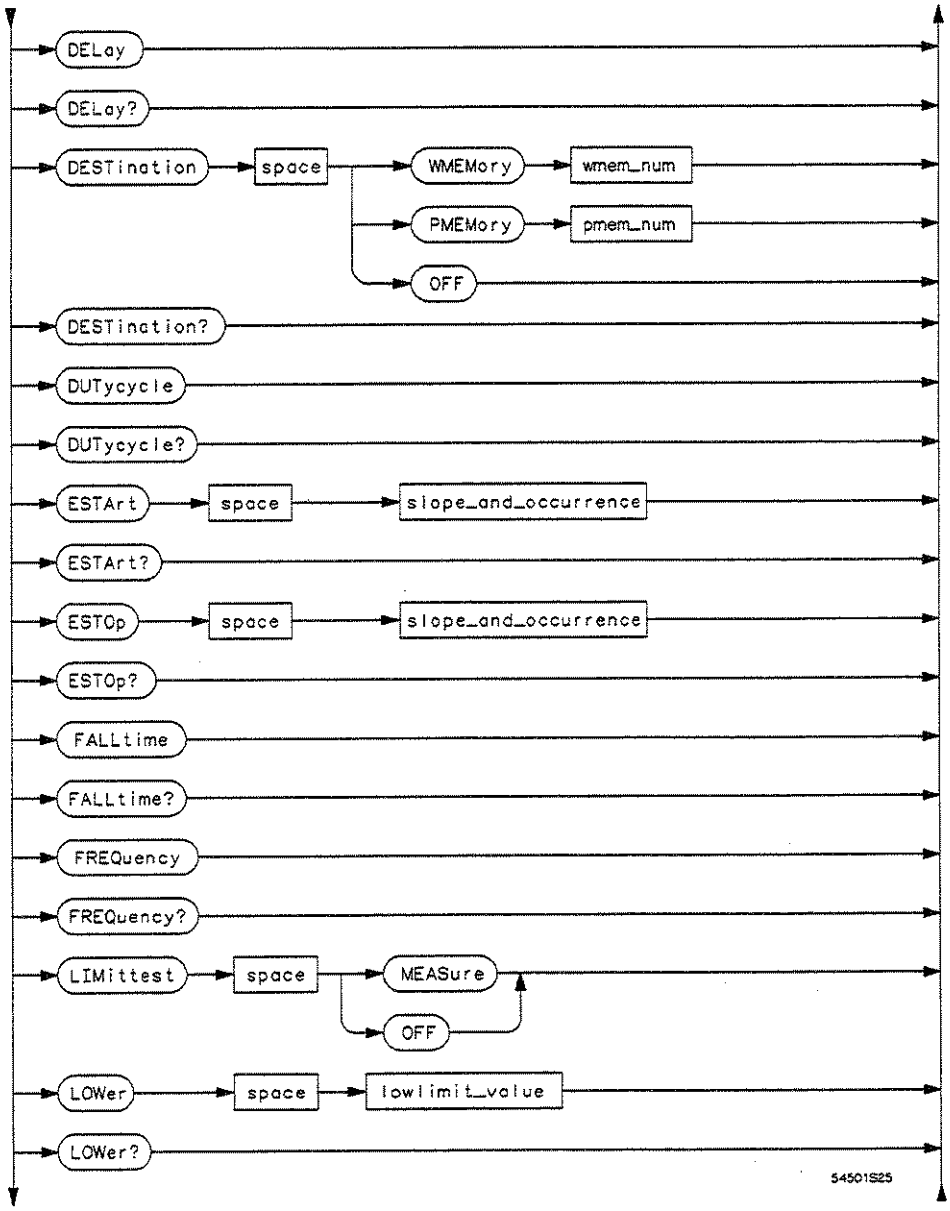


Figure 14-1. Measure Subsystem Commands Syntax Diagram (continued)

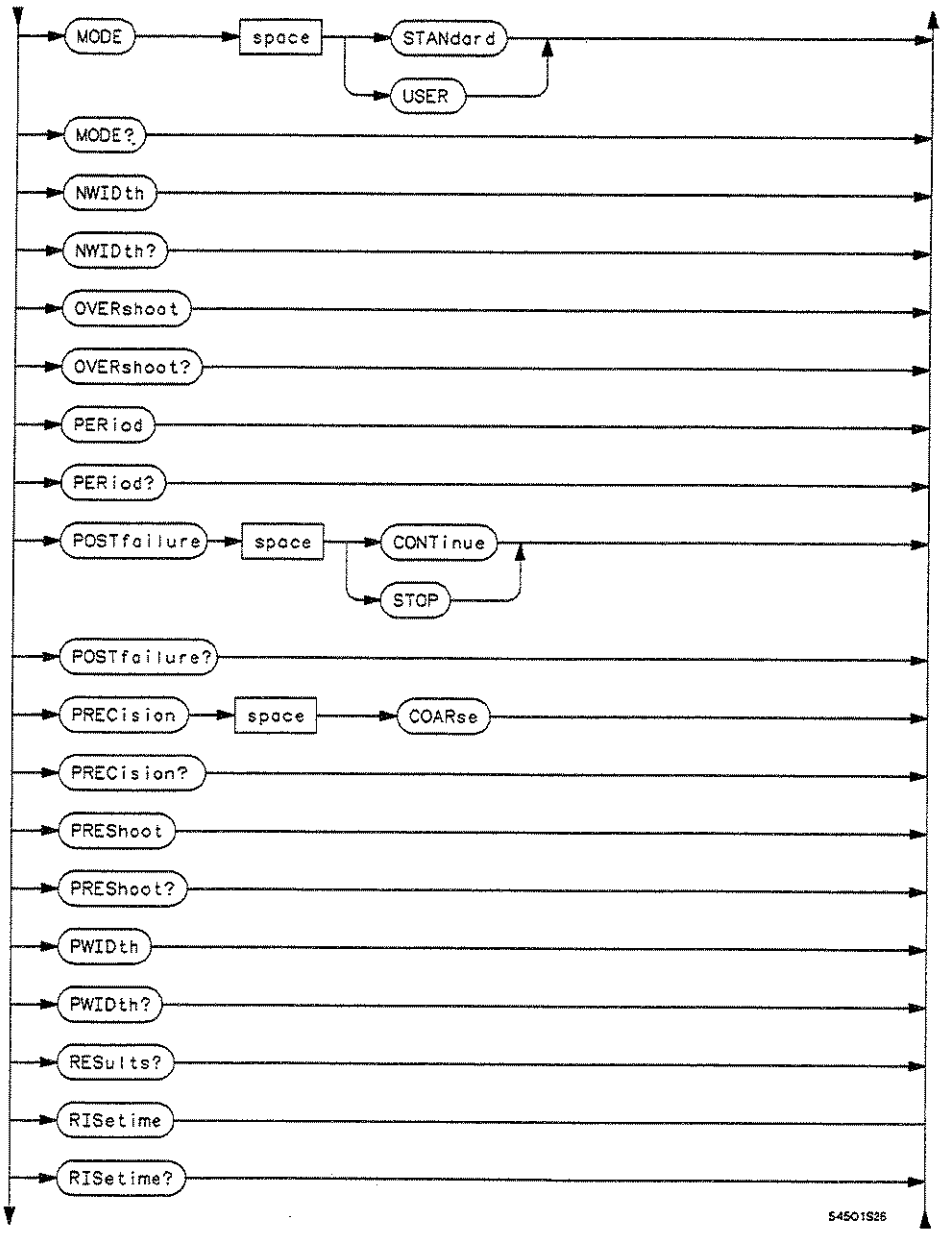


Figure 14-1. Measure Subsystem Commands Syntax Diagram (continued)

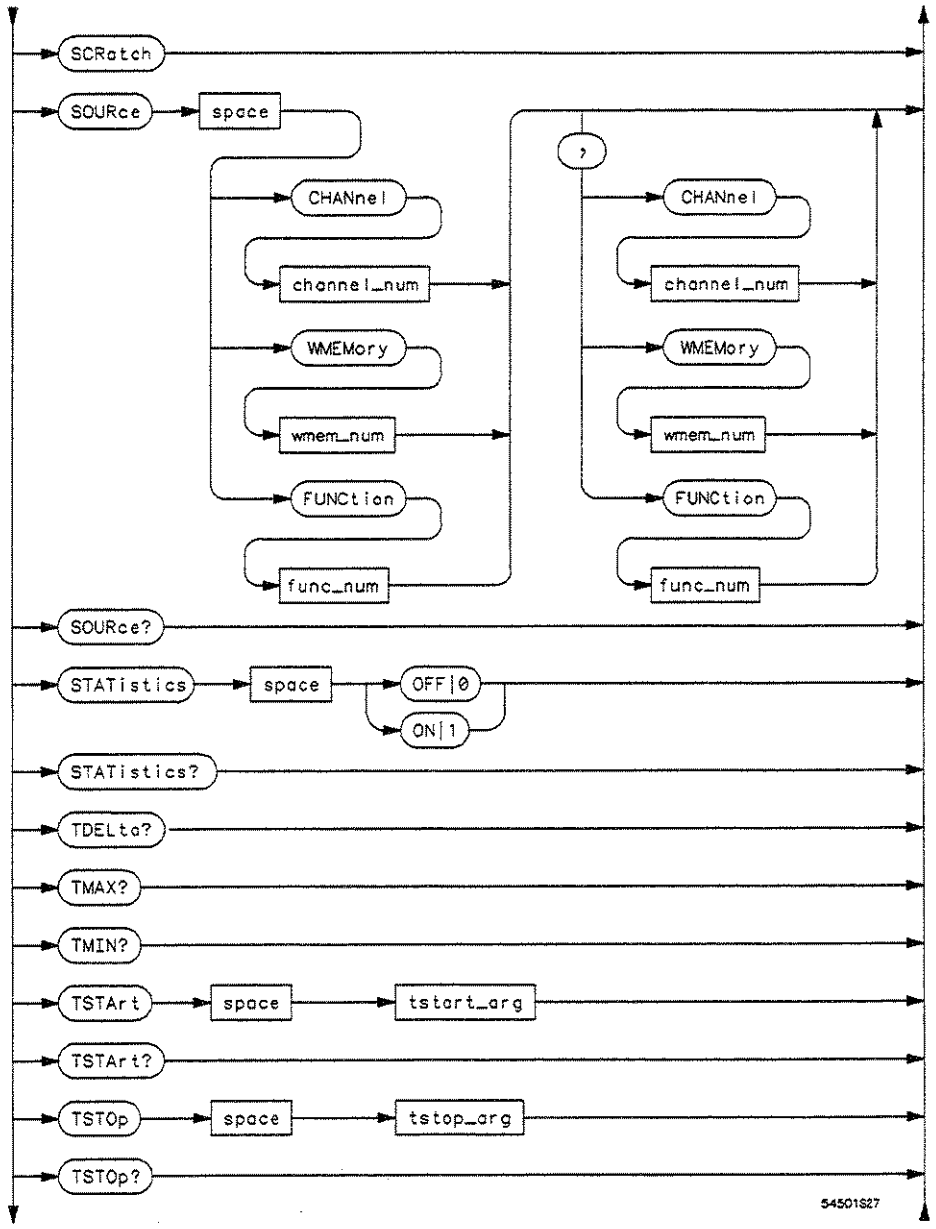


Figure 14-1. Measure Subsystem Commands Syntax Diagram (continued)

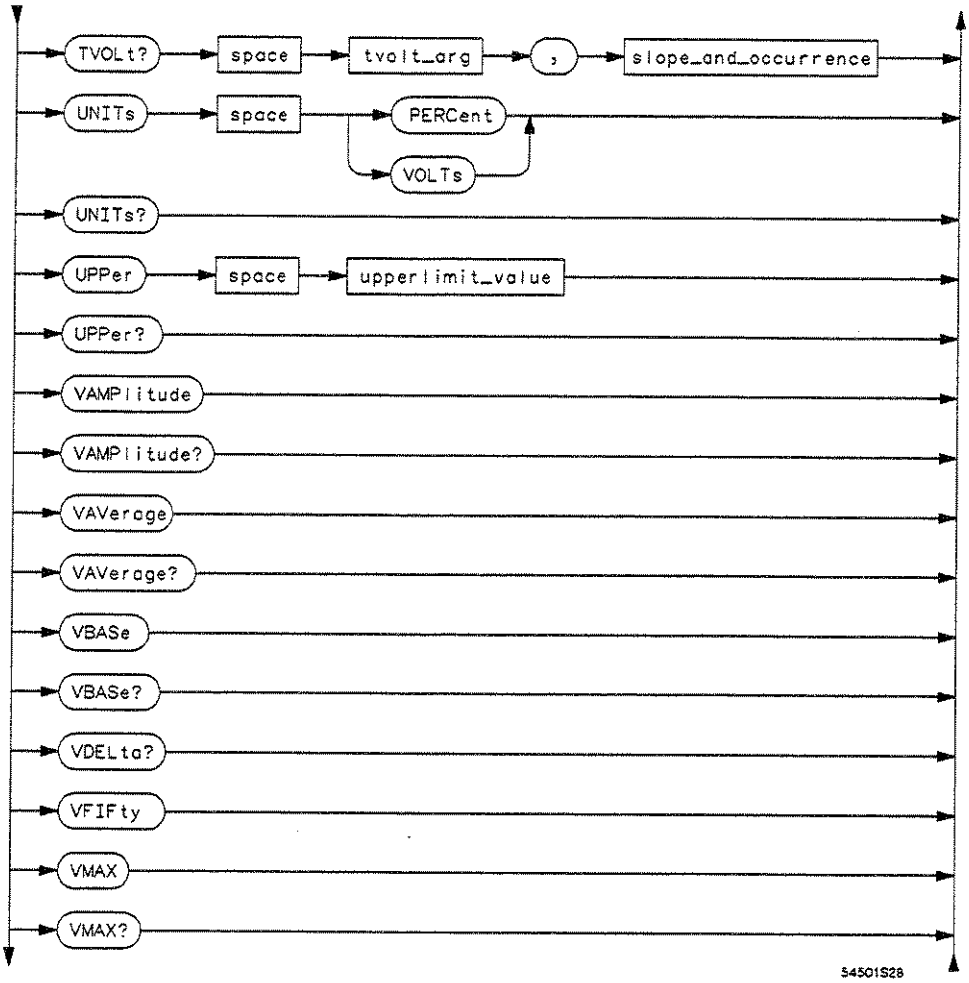


Figure 14-1. Measure Subsystem Commands Syntax Diagram (continued)

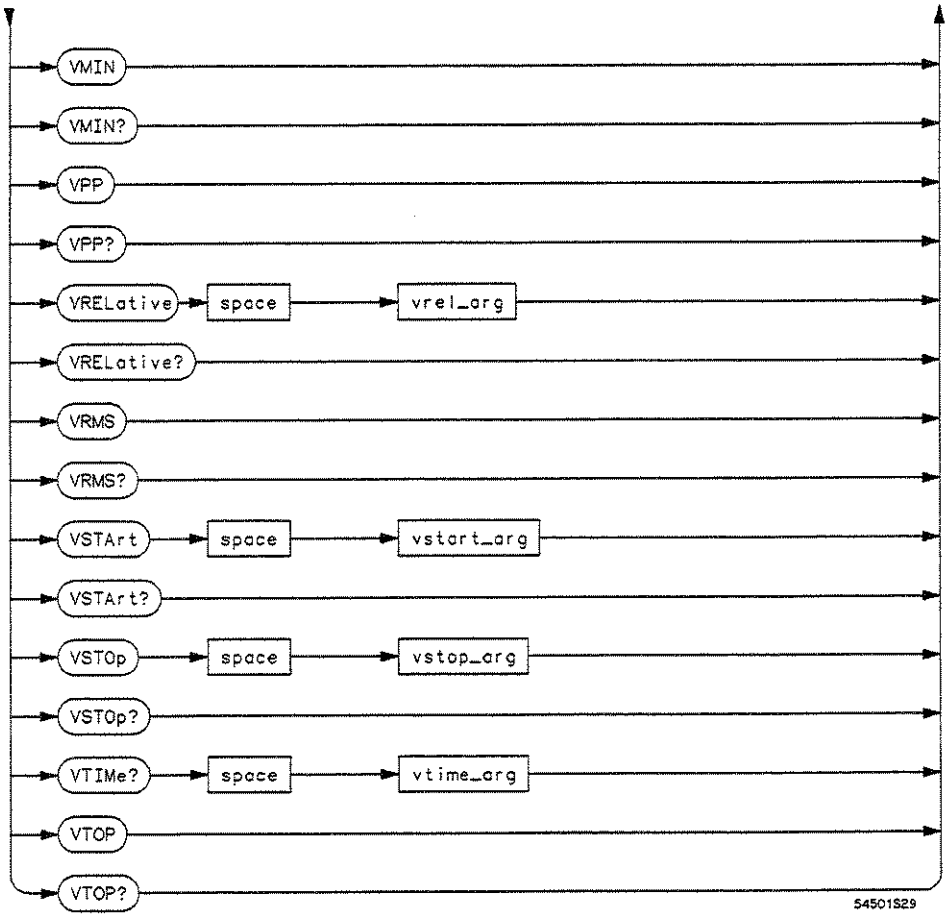


Figure 14-1. Measure Subsystem Commands Syntax Diagram (continued)

channel_number = An integer, 1, 2, 3, or 4.

edge_number = An integer, 1 through 127.

func_num = An integer, 1 or 2.

level = MIDDle, UPPer, or LOWer.

lower_limit = Lower limit for compare.

lowlimit_value = Lower threshold value in percent or volts.

measurement = Name of measurement to be compared.

pmem_num = An integer, 1 or 2.

polarity = POSitive or NEGative

slope_and_occurrence = An integer, -127 to 127 (excluding 0) specifying a displayed edge.

tstart_arg = Time in seconds from trigger.

tstop_arg = Time in seconds from trigger.

tvolt_arg = A real number specifying voltage.

upper_limit = Upper limit for compare.

upperlimit_value = Upper threshold value in percent or volts.

vrel_arg = An integer, 0 to 100.

vstart_arg = A real number within voltage range.

vstop_arg = A real number within voltage range.

vtime_arg = A real number in the horizontal display window.

wmem_num = An integer, 1 through 4.

Figure 14-1. Measure Subsystem Commands Syntax Diagram (continued)

ALL

query

The :MEASURE:ALL query makes a set of measurements on the displayed signal and buffers the measurement results for output over the HP-IB.

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 the returned format of the measurement results.

Query Syntax: :MEASure:ALL?

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>;[VRMS] <result>;[VMAX] <result>;[VMIN] <result>;[VTOP] <result>;[VBASe] <result> [VAVerage] <result> <NL>

Where:

<result> ::= individual measurement results (exponential - NR3 format)

Example:

```
DIM A$[500]
OUTPUT 707;":MEASURE:ALL?"
ENTER 707;"A$"
PRINT A$
```

Note

These values can be returned to numeric variables instead of the string variables as shown. If numeric variables are used, the headers must be turned off.

COMPare

command/query

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.

This command does not start the test, but only sets the test parameters.

The COMPARE query returns the current specification

Command Syntax: :MEASure:COMPare <measurement>,<upper_limit>,<lower_limit>

Where:

<measurement> ::= {RISetime | FALLtime | FREQuency | PERiod | PWIDth | NWIDth | VAMPlitude | VBASe | VTOP | VPP | VAVerage | VMAX | VMIN | VRMS | DUTycycle | DELAY}

<upper_limit> ::= High limit value.
<lower_limit> ::= Low limit value.

Note

When setting the limits for Frequency the suffix "HZ" can be used.

Example:

```
OUTPUT 707;":MEASURE:COMPARE RISETIME,90,10"
```

Query Syntax: :MEASure:COMPare? <measurement>

Where:

<measurement> ::= {RISetime | FALLtime | FREQuency | PERiod | PWIDth | NWIDth | VAMPlitude | VBASe | VTOP | VPP | VAVerage | VMAX | VMIN | VRMS | DUTycycle | DELAY}

Returned Format: [:MEASure:COMPare] <measurement>, <upper_limit>, <lower_limit> <NL>

Example:
```
DIM Cmp$[50]
OUTPUT 707;"MEASure:COMPare? VPP"
ENTER 707;Cmp$
PRINT Cmp$
```

For example, the sequence required to do a limit test on frequency is:

```
OUTPUT 707;":MEASURE:FREQ"                          !Select measurement
OUTPUT 707;":MEASURE:COMPARE FREQ,1000HZ,10HZ"      !Set measurement limits
OUTPUT 707;":MEASURE:LIMITTEST MEASURE"             !Start test
```

Note

The only way to see if a limit test failure has occurred over the bus is by checking if bit 3 of the status byte is set to a 1.

CURSor

query

The :MEASURE:CURSOR query returns the time and voltage values of the specified marker as an ordered pair of time/voltage values.

When the CURSOR query is sent, no measurement is made and the cursors are not moved.

If DELTA is specified:

the instrument returns the value of delta V and delta T.

If START is specified:

the positions of the start marker and VMarker 1 are returned.

If STOP is specified:

the positions of the stop marker and VMarker 2 are returned.

Query Syntax: :MEASure:CURsor? {DELTa | STARt | STOP}

Returned Format: [:MEASure:CURSor] <time>,<voltage><NL>

Where:

<time> ::= delta time, start time or stop time
<voltage> ::= delta voltage, VMarker 1 voltage or VMarker 2 voltage

Example:
```
OUTPUT 707;":MEAS:SOURCE CHAN1"
OUTPUT 707;":MEAS:CURSOR? START"
ENTER 707;Tme,Vlt
PRINT Tme,Vlt
```

DELay

command/query

The :MEASURE:DELAY command causes the instrument to determine 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.

One or two sources can be specified with the :MEASURE:SOURCE command.

If user defined measurement specifications are selected, ensure the defined measurement is displayed.

The DELAY query returns the specified delay value.

Command Syntax: :MEASure:DELay

Example:
```
OUTPUT 707;":MEAS:DEL"
```

Query Syntax: :MEASure:DELay?

Returned Format: [:MEASure:DELay] <delay_value><NL>

Where:

<delay_value> ::= time value in seconds (exponential - NR3 format)

Example:
```
DIM Dly$[50]
OUTPUT 707;":MEAS:DELAY?"
ENTER 707;Dly$
PRINT Dly$
```

DESTination command/query

The :MEASURE:DESTINATION command specifies the destination to be used when a comparison violation is found.

If a waveform memory is specified, the memory is overwritten each time a violation is found.

The DESTINATION query returns the destination currently specified.

Note

If WMEMORIES are used as the destination, the source must be set up separately using the WAVEFORM:SOURCE command.

Command Syntax: :MEASure:DESTination {WMEMory{1 | 2 | 3 | 4} | PMEMory{1 | 2} | OFF}

Example: OUTPUT 707;":MEAS:DEST PMEMORY2"

Query Syntax: :MEASure:DESTination?

Returned Format: [:MEASure:DESTination] {WMEMory{1 | 2 | 3 | 4} | PMEMory{1 | 2} | OFF}<NL>

Example:
```
DIM Dst$[50]
OUTPUT 707;":MEAS:DEST?"
ENTER 707;Dst$
PRINT Dst$
```

DEFine — command/query

The :MEASURE:DEFINE command sets up the definition for a measurement.

The DEFINE query returns the current setup.

Command Syntax: :MEASure:DEFine <measurement_spec>

Where:

<measurement_spec> ::= {DELay <polarity>, <edge_num>, <level>, <polarity>, <edge_num>, <level> | PWIDth {MIDDle | UPPer | LOWer} | NWIDth {MIDDle | UPPer | LOWer}}

Where:

<polarity> ::= {POSitive | NEGative}
<edge_num> ::= an integer, -127 to 127 (excluding 0) specifying a displayed edge
<level> ::= {MIDDle | UPPer | LOWer}

Example: OUTPUT 707;":MEAS:DEFINE DELAY,POSITIVE,1,UPPER,NEGATIVE,2,MIDDLE"

This example will set 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 specifed, 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.

DEFine

Query Syntax: :MEASure:DEFine? {DELay | PWIDth | NWIDth}

Returned Format: [:MEASure:DEFine] <measurement_spec><NL>

Where:

<measurement_spec> ::= {DELay <polarity>, <edge_num>, <level>, <polarity>, <edge_num>, <level> | PWIDth {MIDDle | UPPer | LOWer} | NWIDth {MIDDle | UPPer | LOWer}}

Where:

<polarity> ::= {POSitive | NEGative}
<edge_num> ::= -127 to 127 (excluding 0) (integer - NR1 format)
<level> ::= {MIDDle | UPPer | LOWer}

Example:

```
DIM Dfn$[100]
OUTPUT 707;":MEASure:DEFine? DELay"
ENTER 707;Dfn$
PRINT Dfn$
```

DUTycycle command/query

The :MEASURE:DUTYCYCLE command places the instrument in the continuous measurement mode and starts the Dutycycle measurement.

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 period.

The positive pulse width and the period of the specified signal are measured, then the duty cycle is calculated.

The duty cycle is calculated with the following formula:

duty cycle = +pulse width/period

Command Syntax: :MEASure:DUTycycle

Example: OUTPUT 707;":MEASURE:DUTYCYCLE"

Query Syntax: :MEASure:DUTycycle?

Returned Format: [MEASure:DUTycycle] <value><NL>

Where:

<value> ::= ratio of +pulse width to period (exponential - NR3 format)

Example:
```
DIM Dc$[50]
OUTPUT 707;":MEASURE:DUTYCYCLE?"
ENTER 707;Dc$
PRINT Dc$
```

ESTArt

command/query

The :MEASURE:ESTART command causes the instrument to position the start marker 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 start marker is positioned where VMarker 1 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 will place the start marker on a positive-going waveform edge. If a negative integer is sent, the start marker will be placed on a negative-going waveform edge. If VMarker 1 does not intersect the waveform as specified, the error message "Edges required not found" is displayed.

The ESTART query returns the currently specified edge.

Note

The short form of this command does not follow the defined convention. The short form "EST" is the same for ESTART and ESTOP, so be careful not to send this form for the ESTART command. Sending "EST" will produce an error.

Command Syntax: :MEASure:ESTArt <edge>

Where:

<edge> ::= -127 to 127 excluding 0 (if a positive value is sent the + sign may be omitted or a space may be used)

Example: OUTPUT 707;":MEASURE:ESTART 2"

This example places the start marker at the second displayed positive-going intersection of the waveform and VMarker 1.

Query Syntax: :MEASure:ESTArt?

Returned Format: [:MEASure:ESTArt] <edge> <NL>

Where:

<edge> ::= edge number (integer - NR1 format)

Example:
```
OUTPUT 707;":MEAS:ESTART?"
ENTER 707;Estart
PRINT Estart
```

ESTOp

command/query

The :MEASURE:ESTOP command causes the instrument to position the stop marker 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 stop marker is positioned where VMarker 2 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 stop marker on a positive-going waveform edge. If a negative integer is sent, the stop marker is placed on a negative-going waveform edge.

If VMarker 2 does not intersect the waveform as specified, the error message "Edges required not found" is displayed.

The ESTOP query returns the currently specified edge.

Note

The short form of this command does not follow the defined convention. The short form "EST" is the same for ESTART and ESTOP, so be careful not to send this form for the ESTOP command. Sending "EST" will produce an error.

Command Syntax: :MEASure:ESTOp <edge>

Where:

<edge> ::= -127 to 127 excluding 0 (if a positive value is sent the + sign may be omitted or a space may be used)

Example: OUTPUT 707;":MEAS:ESTOP -2"

This example places the stop marker at the second displayed negative-going intersection of the waveform at VMarker 2.

Query Syntax: :MEASure:ESTOp?

Returned Format: [:MEASure:ESTOp] <edge> <NL>

Where:

<edge> ::= edge number (integer - NR1 format)

Example:
```
OUTPUT 707;":MEASURE:ESTOP?"
ENTER 707;Estop
PRINT Estop
```

FALLtime command/query

The :MEASURE:FALLTIME command places the instrument in the continuous measurement mode and starts a Falltime measurement.

The FALLTIME query measures and outputs the fall time of the first displayed falling (negative-going) edge. For best measurement accuracy set the sweep speed as fast as possible while leaving the falling edge of the waveform on the display. The falltime 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 falltime with the formula:

fall time = time at lower threshold point - time at upper threshold point.

If the horizontal scaling is questionable when this measurement is made and "error 11" is placed in the error queue.

Command Syntax: :MEASure:FALLtime

Example:
```
OUTPUT 707;":MEAS:FALL"
```

Query Syntax: :MEASure:FALLtime?

Returned Format: [:MEASure:FALLtime] <value> <NL>

Where:

<value> ::= time in seconds between lower threshold and upper threshold voltage points (exponential - NR3 format)

Example:
```
DIM FII$[50]
OUTPUT 707;":MEASURE:FALLTIME?"
ENTER 707;FII$
PRINT FII$
```

FREQuency command/query

The :MEASURE:FREQUENCY command places the instrument in the continuous measurement mode and starts a Frequency measurement.

The FREQUENCY query measures and outputs the frequency of the first complete cycle on screen using 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 second rising edge
- time at first rising edge)
else
frequency = 1/(time at second falling edge
- time at first falling edge)

Command Syntax: :MEASure:FREQuency

Example:
```
OUTPUT 707;":MEASURE:FREQ"
```

Query Syntax: :MEASURE:FREQuency?

Returned Format: [:MEASure:FREQuency] <value> <NL>

Where:

<value> ::= frequency in Hertz (exponential - NR3 format)

Example:
```
DIM Frq$[50]
OUTPUT 707;":MEASURE:FREQUENCY?"
ENTER 707;Frq$
PRINT Frq$
```

LIMittest command

The :MEASURE:LIMITTEST command allows a limit test to be performed. 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 will be set when a failure is found.

Command Syntax: :MEASure:LIMittest {MEASure | OFF}

Example: OUTPUT 707;":MEAS:LIM MEAS"

LOWer command/query

The :MEASURE:LOWER command sets the lower measurement threshold. This command sends the value to the instrument. The value that is sent will be in the units selected with the UNITS command.

Note

The measure UNITS should be set prior to sending this value.

The LOWER query returns the current setting of the lower measurement threshold.

Command Syntax: :MEASure:LOWer <lowlimit_value>

Where:

<lowlimit_value> ::= user defined lower threshold in percent or volts

Example:
```
OUTPUT 707;":MEASURE:LOWER 47"
```

Query Syntax: :MEASure:LOWer?

Returned Format: [:MEASure:LOWer] <lowlimit_value><NL>

Where:

<lowlimit_value> ::= user defined lower threshold in percent or volts

Example:
```
DIM Lwr$[50]
OUTPUT 707;":MEAS:LOW?"
ENTER 707;Lwr$
PRINT Lwr$
```

MODE

command/query

The :MEASURE:MODE command sets the measurement mode (definitions and thresholds).

The MODE query returns the current mode setting.

Command Syntax: :MEASure:MODE {STANdard | USER}

Example:
```
OUTPUT 707;":MEAS:MODE STAN"
```

Query Syntax: :MEASure:MODE?

Returned Format: [:MEASure:MODE] {STANdard | USER} <NL>

Example:
```
DIM Md$[50]
OUTPUT 707;":MEASURE:MODE?"
ENTER 707;Md$
PRINT Md$
```

NWIDth

command/query

The :MEASURE:NWIDTH command places the instrument in the continuous measurement mode and starts a NWIDTH measurement.

The NWIDTH query measures and outputs the width of the first negative pulse on screen using the 50% levels with Standard measurements selected.

If User Defined measurements are selected, then the measurement is made at the mid threshold value.

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)

Command Syntax: :MEASure:NWIDth

Example:
```
OUTPUT 707;":MEAS:NWIDTH"
```

Query Syntax: :MEASure:NWIDth?

Returned Format: [:MEASure:NWIDth] <value> <NL>

Where:

<value> ::= negative pulse width in seconds (exponential - NR3 format)

Example:
```
DIM Nwd$[50]
OUTPUT 707;":MEASURE:NWIDTH?"
ENTER 707;Nwd$
PRINT Nwd$
```

OVERshoot

command/query

The :MEASURE:OVERSHOOT command places the instrument in the continuous measurement mode and selects the Overshoot measurement.

The OVERSHOOT query measures and outputs the overshoot of a selected signal. Overshoot measures the first edge on screen with the following algorithm:

if the first edge on screen is rising
then
overshoot = (Vmax - Vtop)/Vamplitude
else
overshoot = (Vbase - Vmin)/Vamplitude

Command Syntax: :MEASure:OVERshoot

Example:
```
OUTPUT 707;":MEAS:OVER"
```

Query Syntax: :MEASure:OVERshoot?

Returned Format: [:MEASure:OVERshoot] <value> <NL>

Where:

<value> :: = ratio of overshoot to Vamplitude (exponential - NR3 format)

Example:
```
DIM Ovr$[50]
OUTPUT 707;":MEASURE:OVERSHOOT?"
ENTER 707;Ovr$
PRINT Ovr$
```

PERiod — command/query

The :MEASURE:PERIOD command places the instrument in the continuous measurement mode and selects the Period measurement.

The PERIOD query measures and outputs the period of the first complete cycle on 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)

Command Syntax: :MEASure:PERiod

Example:
```
OUTPUT 707;":MEAS:PERIOD"
```

Query Syntax: :MEASure:PERiod?

Returned Format: [:MEASure:PERiod] <value> <NL>

Where:

<value> ::= waveform period in seconds (exponential - NR3 format)

Example:
```
DIM Prd$[50]
OUTPUT 707;":MEASURE:PERIOD?"
ENTER 707;Prd$
PRINT Prd$
```

POSTfailure command/query

The :MEASURE:POSTFAILURE command specifies what will occur after a violation has been found by the limit test. If CONTINUE is selected, the instrument will continue to look for another violation. If STOP is selected the instrument will stop the limit test.

If CONTINUE is selected and a violation is found the violation will be written to the desired location. If a waveform memory is selected as the destination then all subsequent violations will overwrite the previous violation.

The POSTFAILURE query returns the current selection.

Command Syntax: :MEASure:POSTfailure {CONTinue | STOP}

Example:
```
OUTPUT 707;":MEAS:POST"
```

Query Syntax: :MEASure:POSTfailure?

Returned Format: [:MEASure:POSTfailure] {CONTinue | STOP}<NL>

Example:
```
DIM Pf$[50]
OUTPUT 707;":MEASURE:POSTFAILURE?"
ENTER 707;Pf$
PRINT Pf$
```

PRECision command/query

The :MEASURE:PRECISION command is included in the HP 54501A for compatibility with other HP instruments. It has no effect on the HP 54501A.

The PRECISION query always returns COARSE in the HP 54501A.

Command Syntax: :MEASure:PRECision COARse

Example:
```
OUTPUT 707;":MEAS:PREC COARSE"
```

Query Syntax: :MEASure:PRECision?

Returned Format: [:MEASure:PRECision] COARse<NL>

Example:
```
DIM Pc$[50]
OUTPUT 707;":MEAS:PRECISION?"
ENTER 707;Pc$
PRINT Pc$
```

PREShoot

command/query

The :MEASURE:PRESHOOT command places the instrument in the continuous measurement mode and starts the Preshoot measurement.

The PRESHOOT query measures and outputs the preshoot of the selected signal. Preshoot measures the first edge on screen with the following algorithm:

if the first edge on screen is rising
then
preshoot = (Vbase - Vmin)/Vamplitude
else
preshoot = (Vmax - Vtop)/Vamplitude

Command Syntax: :MEASure:PREShoot

Example: OUTPUT 707;":MEASURE:PRES"

Query Syntax: :MEASure:PREShoot?

Returned Format: [:MEASure:PREShoot] <value><NL>

Where:

<value> ::= ratio of preshoot to Vamplitude (exponential - NR3 format)

Example:
```
DIM Prs$[50]
OUTPUT 707;":MEASURE:PRESHOOT?"
ENTER 707;Prs$
PRINT Prs$
```

PWIDth command/query

The :MEASURE:PWIDTH command places the instrument in the continuous measurement mode and starts the Pwidth measurement.

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 and at the mid level threshold value with User-Defined measurements selected. 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)

Command Syntax: :MEASure:PWIDth

Example:
```
OUTPUT 707;":MEAS:PWIDTH"
```

Query Syntax: :MEASure:PWIDth?

Returned Format: [:MEASure:PWIDth] <value> <NL>

Where:

<value> ::= width of positive pulse in seconds (exponential - NR3 format)

Example:
```
DIM Pwd$[50]
OUTPUT 707;":MEASURE:PWIDTH?"
ENTER 707;Pwd$
PRINT Pwd$
```

RESults

query

The :MEASURE:RESULTS query tells the instrument to return the currently active measurements. If statistics are on the current, minimum, maximum, and average will be returned for each measurement. If the limit test is on and POSTFAILURE is set to CONTINUED then the pass ratio will be returned instead of the average.

If the number of measurements returned is 0 then no <measurement>s are returned.

Query Syntax: :MEASure:RESults?

Returned Format: [:MEASure:RESults] <No. of Meas>[;<measurement>]...<NL>

Where:

<No. of Meas> ::= number of measurements displayed on the CRT, 0 through 8. (integer - NR1 format)
<measurement> ::= measurement_name measurement_result.

Example:
```
DIM Mr$[100]
OUTPUT 707;":MEASURE:RESULTS?"
ENTER 707;Mr$
PRINT Mr$
```

RISetime

command/query

The :MEASURE:RISETIME command places the instrument in the continuous measurement mode and starts a Risetime measurement.

The RISETIME query measures and outputs the rise time of the first displayed rising (positive-going) edge. For best 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 then the time at the upper threshold of the rising edge and calculating the rise time with the formula:

rise time = (time at upper threshold point - time at lower threshold point)

If the horizontal scaling is questionable when this measurement is made an "error 11" is placed in the error queue.

Command Syntax: :MEASure:RISetime

Example:
```
OUTPUT 707;":MEAS:RIS"
```

Query Syntax: :MEASure:RISetime?

Returned Format: [:MEASure:RISetime] <value><NL>

Where:

<value> ::= rise time in seconds (exponential - NR3 format)

Example:
```
DIM Rs$[50]
OUTPUT 707;":MEAS:RIS?"
ENTER 707;Rs$
PRINT Rs$
```

SCRatch command

The :MEASURE:SCRATCH command clears the measurement results from the oscilloscope display.

Command Syntax: :MEASure:SCRatch

Example: OUTPUT 707;":MEASURE:SCRATCH"

SOURce

command/query

The :MEASURE:SOURCE command selects the source(s) for the measurements. The source specified will become 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 will use two sources if two have been specified. If only one source is specified the DELAY measurement will use that source for both of its parameters.

The SOURCE query returns the current source selection. If the specified sources are different both will be returned, otherwise one source will be returned.

Command Syntax: :MEASure:SOURce <source1>[,<source2>]

Where:

<source1> and <source2> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4}}

Example:
```
OUTPUT 707;":MEASURE:SOURCE CHANNEL1, WMEMORY1"
```

Query Syntax: :MEASure:SOURce?

Returned Format: [:MEASure:SOURce] <source1>[,<source2>]<NL>

Where:

<source1> and <source2> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | WMEMory{1 | 2 | 3 | 4}}

Example:
```
DIM Src$[50]
OUTPUT 707;":MEAS:SOUR?"
ENTER 707;Src$
PRINT Src$
```

STATistics command/query

The :MEASURE:STATISTICS command allows the statistics mode to be controlled. When this mode is on, and the measurements are in the continuous mode, the min, max, avg, and current measurement will be shown as the active measurements. If a RESULTS query is executed, all of the displayed data will be returned to the controller.

The STATISTICS query returns the current mode.

Note

"Average" will be replaced by "pass ratio" when limit test is selected and "after failure" is set to continue. Pass ratio lists the percentage of times a certain test passed.

Command Syntax: :MEASure:STATistics {{ON | 1} | {OFF | 0}}

Example: OUTPUT 707;":MEASURE:STAT ON"

Query Syntax: :MEASure:STATistics?

Returned Format: [:MEASure:STATistics] {1 | 0}<NL>

Example:
```
DIM Stt$[50]
OUTPUT 707;":MEASURE:STAT?"
ENTER 707;Stt$
PRINT Stt$
```

TDELta

query

The :MEASURE:TDELTA query returns the time difference between the start and stop time markers, that is:

Tdelta = Tstop - Tstart

where Tstart is the time at the start marker and Tstop is the time at the stop marker

Query Syntax: :MEASure:TDELta?

Returned Format: [:MEASure:TDELta] <value> <NL>

Where:

<value> ::= difference between start and stop markers (exponential - NR3 format)

Example:

```
DIM Tdl$[50]
OUTPUT 707;":MEASURE:TDELTA?"
ENTER 707;Tdl$
PRINT Tdl$
```

TMAX

query

The :MEASURE:TMAX query returns the time at which the first maximum voltage occurred.

Query Syntax: :MEASure:TMAX?

Returned Format: [:MEASure:TMAX] <time at max voltage> <NL>

Example:
```
DIM Tmx$[50]
OUTPUT 707;":MEASURE:TMAX?"
ENTER 707;Tmx$
PRINT Tmx$
```

TMIN

query

The :MEASURE:TMIN query returns the time at which the first minimum voltage occurred.

Query Syntax: :MEASure:TMIN?

Returned Format: [:MEASure:TMIN] <time at min voltage> <NL>

Example:

```
DIM Tmn$[50]
OUTPUT 707;":MEAS:TMIN?"
ENTER 707;Tmn$
PRINT Tmn$
```

TSTArt

command/query

The :MEASURE:TSTART command moves the start marker to the specified time with respect to the trigger time.

The TSTART query returns the time at the start marker.

Note

The short form of this command does not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP, so be careful not to send this form for the TSTART command. Sending "TST" will produce an error.

Command Syntax: :MEASure:TSTArt <start marker time>

Where:

<start marker time> ::= time at start marker in seconds (exponential - NR3 format)

Example:
```
OUTPUT 707;":MEASURE:TSTART 30 NS"
```

Query Syntax: :MEASure:TSTArt?

Returned Format: [:MEASure:TSTArt] <value><NL>

Where:

<value> ::= time at start marker in second (exponential - NR3 format)

Example:
```
DIM Tst$
OUTPUT 707;":MEASURE:TSTART?"
ENTER 707;Tst$
PRINT Tst$
```

TSTOp

command/query

The :MEASURE:TSTOP command moves the stop marker to the specified time with respect to the trigger time.

The TSTOP query returns the time at the stop marker.

Note

The short form of this command does not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP, so be careful not to send this form for the TSTOP command. Sending "TST" will produce an error.

Command Syntax: :MEASure:TSTOp <stop marker time>

Where:

<stop marker time> ::= time at stop marker in seconds

Example:
```
OUTPUT 707;":MEAS:TSTOP 40 NS"
```

Query Syntax: :MEASure:TSTOp?

Returned Format: [:MEASure:TSTOp] <value><NL>

Where:

<value> ::= time at stop marker in seconds (exponential - NR3 format)

Example:
```
DIM Tst$[50]
OUTPUT 707;":MEASURE:TSTOP?"
ENTER 707;Tst$
PRINT Tst$
```

TVOLt

query

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 <slope> selects a rising (+) or falling (-) edge.

The magnitude of <occurrence> defines the occurrence to be reported. For example, +3 will return 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 will output 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 will happen 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.

Query Syntax: :MEASure:TVOLt? <voltage>, <slope><occurrence>

Where:

<voltage> ::= voltage level the waveform must cross. This can be a positive or negative voltage.

<slope> ::= direction of waveform when <voltage> is crossed rising (sp or +) or falling (-)

<occurrence> ::= number of crossing to be reported (if one - first crossing reported, if two - second crossing is reported, etc.)

Returned Format: [:MEASure:TVOLt] <time> <NL>

Where:

<time> ::= time in seconds of specified voltage crossing (exponential - NR3 format)

Example:

```
DIM Tvlt$[50]
OUTPUT 707;"MEASURE:TVOLT? -.250, +3"
ENTER 707;Tvlt$
PRINT Tvlt$
```

UNITs

command/query

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.

The UNITS query returns the currently selected units.

Command Syntax: :MEASure:UNITs {PERCent | VOLTs}

Example:
```
OUTPUT 707;":MEASURE:UNITS PERCENT"
```

Query Syntax: :MEASure:UNITs?

Returned Format: [:MEASure:UNITs] {PERCent | VOLTs} <NL>

Example:
```
DIM Unt$[50]
OUTPUT 707;":MEASURE:UNITS?"
ENTER 707;Unt$
PRINT Unt$
```

UPPer command/query

The :MEASURE:UPPER command sets the upper measurement threshold.

Note

The measure UNITS should be set prior to sending this value.

The UPPER query returns the value of the upper measurement threshold.

Command Syntax: :MEASure:UPPer <value>

Where:

<value> ::= upper threshold value in percent or volts

Example:
```
OUTPUT 707;":MEAS:UPPER 90"
```

Query Syntax: :MEASure:UPPer?

Returned Format: [:MEASure:UPPer] <upper_threshold value><NL>

Where:

<upper_threshold value> ::= upper threshold value in percent or volts (exponential - NR3 format)

Example:
```
DIM Upp$[50]
OUTPUT 707;":MEAS:UPP?"
ENTER 707;Upp$
PRINT Upp$
```

VAMPlitude

command/query

The :MEASURE:VAMPLITUDE command places the instrument in the continuous measurement mode and starts a Vamplitude measurement.

The VAMPLITUDE query returns the difference between the top and base voltage of the displayed signal. The VAMPLITUDE value will not normally be the same as the Vp-p value if the input signal is a pulse.

The Vamplitude value is calculated with the formula:

Vamplitude = Vtop - Vbase

Command Syntax: :MEASure:VAMPlitude

Example: OUTPUT 707;":MEAS:VAMP"

Query Syntax: :MEASure:VAMPlitude?

Returned Format: [:MEASure:VAMPlitude] <value><NL>

Where:

<value> ::= difference between top and base voltages (exponential - NR3 format)

Example:
```
DIM Vmp$[50]
OUTPUT 707;":MEASURE:VAMPLITUDE?"
ENTER 707;Vmp$
PRINT Vmp$
```

VAVerage command/query

The :MEASURE:VAVERAGE command places the instrument in the continuous measurement mode and starts a Vaverage measurement.

The VAVERAGE query calculates the average voltage over the waveform or the displayed data points.

Command Syntax: :MEASure:VAVerage

Example:
```
OUTPUT 707;":MEAS:VAV"
```

Query Syntax: :MEASure:VAVerage?

Returned Format: [:MEASure:VAVerage] <avg_value><NL>

Where:

<avg_value> ::= calculated average voltage (exponential - NR3 format)

Example:
```
DIM Vv$[50]
OUTPUT 707;":MEAS:VAV?"
ENTER 707;Vv$
PRINT Vv$
```

VBASe

command/query

The :MEASURE:VBASE command places the instrument in the continuous measurement mode and starts a Vbase measurement.

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.

Command Syntax: :MEASure:VBASe

Example:
```
OUTPUT 707;":MEAS:VBASE"
```

Query Syntax: :MEASure:VBASe?

Returned Format: [:MEASure:VBASe] <value> <NL>

Where:

<value> :: = voltage at base of selected waveform (exponential - NR3 format)

Example:
```
DIM Vbs$[50]
OUTPUT 707;":MEASURE:VBASE?"
ENTER 707;Vbs$
PRINT Vbs$
```

VDELta

query

The :MEASURE:VDELTA query outputs the voltage difference between VMarker 1 and VMarker 2. 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 V.

VDELTA = Voltage at VMarker 2 - Voltage at VMarker 1

Query Syntax: :MEASure:VDELta?

Returned Format: [:MEASure:VDELta] <value> <NL>

Where:

<value> ::= delta V value in volts (exponential - NR3 format)

Example:

```
DIM Vdl$[50]
OUTPUT 707;":MEAS:VDELTA?"
ENTER 707;Vdl$
PRINT Vdl$
```

VFIFty command

The :MEASURE:VFIFTY command instructs the oscilloscope to find the top and base values of the specified waveforms, then places the voltage markers at the 50% voltage point on the specified source(s).

If only one source has been specified with the source command, the VFIFty command sets both voltage markers (VMarker 1 and VMarker 2) to the 50% voltage level on that source.

If two sources are specified with the source command, VMarker 1 is set to the 50% level of the first specified source and VMarker 2 is set to the 50% level of the second specified source.

There is no query form of this command.

Command Syntax: :MEASure:VFIFty

Example: OUTPUT 707;":MEASURE:VFIFTY"

VMAX

command/query

The :MEASURE:VMAX command places the instrument in the continuous measurement mode and starts a Vmax measurement.

The VMAX query measures and outputs the absolute maximum voltage present on the selected waveform.

Command Syntax: :MEASure:VMAX

Example:
```
OUTPUT 707;":MEAS:VMAX"
```

Query Syntax: :MEASure:VMAX?

Returned Format: [:MEASure:VMAX] <value> <NL>

Where:

<value> ::= maximum voltage of selected waveform (exponential - NR3 format)

Example:
```
DIM Vmx$[50]
OUTPUT 707;":MEASURE:VMAX?"
ENTER 707;Vmx$
PRINT Vmx$
```

VMIN command/query

The :MEASURE:VMIN command places the instrument in the continuous measurement mode and starts a VMIN measurement.

The VMIN query measures and outputs the absolute minimum voltage present on the selected waveform.

Command Syntax: :MEASure:VMIN

Example: OUTPUT 707;":MEAS:VMIN"

Query Syntax: :MEASure:VMIN?

Returned Format: [:MEASure:VMIN] <value><NL>

Where:

<value> ::= minimum voltage value of the selected waveform (exponential - NR3 format)

Example:
```
DIM Vmn$[50]
OUTPUT 707;":MEASURE:VMIN?"
ENTER 707;Vmn$
PRINT Vmn$
```

VPP command/query

The :MEASURE:VPP command places the instrument in the continuous measurement mode and starts a VPP measurement.

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 formula:

Vpp = Vmax - Vmin

where Vmax and Vmin are the maximum and minimum voltages present on the selected source.

Command Syntax: :MEASure:VPP

Example:
```
OUTPUT 707;":MEAS:VPP"
```

Query Syntax: :MEASure:VPP?

Returned Format: [:MEASure:VPP] <value><NL>

Where:

<value> ::= voltage peak to peak (exponential - NR3 format)

Example:
```
DIM Vp$[50]
OUTPUT 707;":MEAS:VPP?"
ENTER 707;Vp$
PRINT Vp$
```

VRELative

command/query

The :MEASURE:VRELATIVE command moves the voltage markers 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 :MEAS:VAMPLITUDE? query has been sent VMarker 1 is located at the base (0%) of the signal and VMarker 2 is at the top (100%) of the signal. If the VRELATIVE 10 command was executed, VMarker 1 is moved to the 10% level and VMarker 2 to the 90% level of the signal.

Any value between 0% and 100% can be used. If VRELATIVE 0 is sent the markers 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 VMarker 1 to 10% and VMarker 2 to 90%
20 moves VMarker 1 to 20% and VMarker 2 to 80%
50 moves both markers to 50%
80 moves VMarker 1 to 20% and VMarker 2 to 80%
90 moves VMarker 1 to 10% and VMarker 2 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 :MEAS:VAMPLITUDE? query.

The VRELATIVE query returns the current relative position of VMarker2 which is always in the range 50% through 90%.

Note

The VRELATIVE command does not affect the upper and lower thresholds selected by the UPPER and LOWER commands.

Command Syntax: :MEASure:VRELative <percent>

Where:

<percent> ::= {0 through 100}

Example:
```
OUTPUT 707;":MEASURE:VRELATIVE 20"
```

Query Syntax: :MEASure:VRELative?

Returned Format: [:MEASure:VRELative] <value> <NL>

Where:

<value> ::= Vmarker 2 relative position in percent {50 through 100} (integer - NR1 format)

Example:
```
DIM Vrl$[50]
OUTPUT 707;":MEAS:VREL?"
ENTER 707;Vrl$
PRINT Vrl$
```

VRMS

command/query

The :MEASURE:VRMS command places the instrument in the continuous measurement mode and starts a Vrms measurement.

The VRMS query measures and outputs the RMS voltage of the selected waveform. The RMS voltage is computed using the data available on the display.

Note

This RMS measurement is an AC RMS measurement. This means that the average value of the waveform is subtracted from each data point before RMS is computed.

Command Syntax: :MEASure:VRMS

Example:
```
OUTPUT 707;":MEAS:VRMS"
```

Query Syntax: :MEASure:VRMS?

Returned Format: [:MEASure:VRMS] <value> <NL>

Where:

<value> ::= rms voltage of displayed points (exponential - NR3 format)

Example:
```
DIM Vrm$[50]
OUTPUT 707;":MEASURE:VRMS?"
ENTER 707;Vrm$
PRINT Vrm$
```

VSTArt command/query

The :MEASURE:VSTART command moves VMarker 1 to the specified voltage. The values are limited to the currently defined channel, function, or memory range.

The VSTART query returns the current voltage level of VMarker 1.

Note

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP, so be careful not to send this form for the VSTART command. Sending "VST" will produce an error.

Command Syntax: :MEASure:VSTArt <voltage>

Example: OUTPUT 707;":MEAS:VSTA -10MV"

Where:

<voltage> ::= voltage value for Vmarker 1

Query Syntax: :MEASure:VSTArt?

Returned Format: [:MEASure:VSTArt] <value><NL>

Where:

<value> ::= voltage at VMarker 1 (exponential - NR3 format)

Example:
```
DIM Vst$[50]
OUTPUT 707;":MEASURE:VSTART?"
ENTER 707;Vst$
PRINT Vst$
```

VSTOp

command/query

The :MEASURE:VSTOP command moves VMarker 2 to the specified voltage.

The VSTOP query returns the current voltage level of VMarker 2.

Note

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP, so be careful not to send this form for the VSTOP command. Sending "VST" will produce an error.

Command Syntax: :MEASure:VSTOp <voltage>

Where:

<voltage> ::= voltage value for Vmarker 2

Example:
```
OUTPUT 707;":MEAS:VSTO -100MV"
```

Query Syntax: :MEASure:VSTOp?

Returned Format: [:MEASure:VSTOp] <value><NL>

Where:

<value> ::= voltage at VMarker 2 (exponential - NR3 format)

Example:
```
DIM Vst$[50]
OUTPUT 707;":MEASURE:VSTOP?"
ENTER 707;Vst$
PRINT Vst$
```

VTIMe query

The :MEASURE:VTIME query returns the voltage at a specified time. The time is referenced to the trigger event and must be on screen.

Query Syntax: :MEASure:VTIMe? <time>

Where:

<time> ::= displayed time from trigger in seconds

Returned Format: [:MEASure:VTIMe] <value><NL>

Where:

<value> ::= voltage at specified time (exponential - NR3 format)

Example:

```
DIM Vtm$[50]
OUTPUT 707;":MEASURE:VTIME? .001"
ENTER 707;Vtm$
PRINT Vtm$
```

VTOP

command/query

The :MEASURE:VTOP command places the instrument in the continuous measurement mode and starts a Vtop measurement.

The VTOP query returns the voltage at the top of a waveform.

Command Syntax: :MEASure:VTOP

Example: OUTPUT 707;":MEASURE:VTOP"

Query Syntax: :MEASure:VTOP?

Returned Format: [:MEASure:VTOP] <value><NL>

Where:

<value> ::= voltage at top of waveform (exponential - NR3 format)

Example:
```
DIM Vtp$
OUTPUT 707;":MEASURE:VTOP?"
ENTER 707;Vtp$
PRINT Vtp$
```

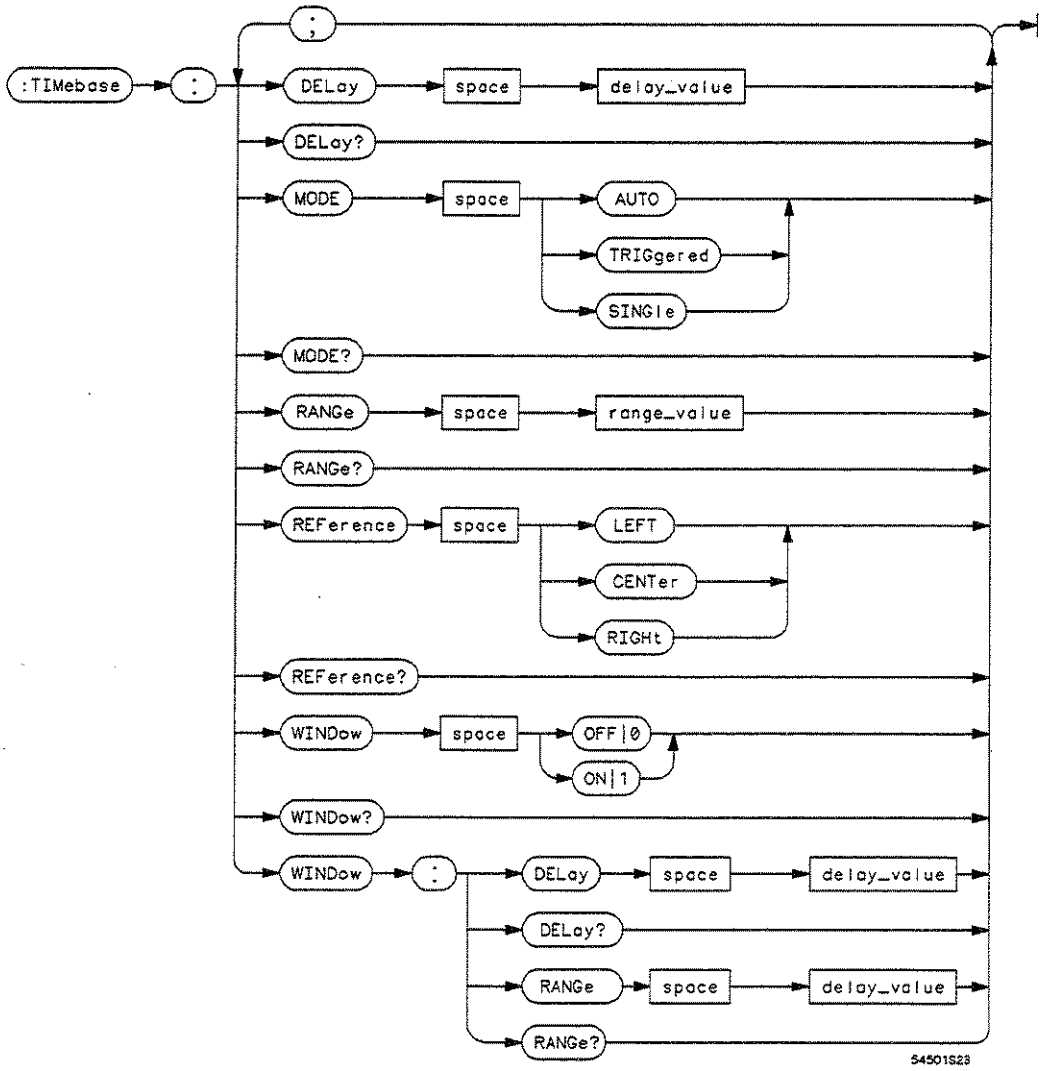
Timebase Subsystem 15

Introduction

The TIMEBASE subsystem commands control the horizontal axis, "X axis," oscilloscope functions.

The TIMEBASE subsystem also contains the commands that control the Timebase Window mode.

The Timebase Window mode allows a second timebase to be used. The Window settings are WINDOW:DELAY (window position) and WINDOW:RANGE (window width).



delay_value = A real number, maximum depends on sweep range.

range_value = A real number, 20 ns through 50 s (in a 1,2,5 sequence).

Figure 15-1. Timebase Subsystem Commands Syntax Diagram

DELay

command/query

The :TIMEBASE:DELAY command sets the timebase delay. This delay is the time interval 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 center of the display, and is set with the :TIMEBASE:REFERENCE command.

The DELAY query returns the current delay value.

Command Syntax: :TIMebase:DELay <delay>

Where:

<delay> ::= time in seconds from trigger to on screen delay reference point. Maximum value depends on time/division setting.

Example:
```
OUTPUT 707;":TIM:DEL 2MS"
```

Query Syntax: :TIMebase:DELay?

Returned Format: [:TIMebase:DELay] <delay> <NL>

Where:

<delay> ::= time from trigger to display reference in seconds. Display reference is left, center, or right (exponential - NR3 format)

Example:
```
DIM DI$[50]
OUTPUT 707;":TIMEBASE:DELAY?"
ENTER 707;DI$
PRINT DI$
```

MODE

command/query

The :TIMEBASE:MODE command selects the timebase mode. This function is the same as the trigger menu Auto/Trig'd key and the SINGLE key on the front panel.

If the AUTO mode is selected, the unit will provide a baseline on the display in the absence of a signal. If a signal is present but the oscilloscope is not triggered, the display will be unsynchronized but will not be a baseline.

If the TRIGGERED mode is selected and no trigger is present, the unit will not sweep, and the data acquired on the previous trigger will remain on screen.

If the SINGLE mode is selected the screen will be cleared and the instrument will be stopped. The RUN command will arm the trigger, then data will be acquired when the trigger is found. To make a single acquisition a RUN command should be sent.

The MODE query returns the current mode.

Command Syntax: :TIMebase:MODE {AUTO | TRIGgered | SINGle}

Example: OUTPUT 707;":TIM:MODE TRIGGERED"

Query Syntax: :TIMebase:MODE?

Returned Format: [:TIMbase:MODE] <mode><NL>

<mode> ::= {AUTO | TRIGgered | SINGle}

Example:
```
DIM Mode$[30]
OUTPUT 707;"TIMEBASE:MODE?"
ENTER 707;Mode$
PRINT Mode$
```

RANGe command/query

The :TIMEBASE:RANGE command sets the full scale horizontal time in seconds. The RANGE value is ten times the time per division.

The RANGE query returns the current range value.

Command Syntax: :TIMebase:RANGe <range>

<range> ::= 20 ns to 50 s in a 1,2,5 sequence"

Example: OUTPUT 707;":TIM:RANG 100 MS"

Query Syntax: :TIMebase:RANGe?

Returned Format: [:TIMebase:RANGe] <range> <NL>

Where:

<range> ::= 20 ns to 50 s (exponential - NR3 format)

Example:
```
DIM Rng$[50]
OUTPUT 707;":TIMEBASE:RANGE?"
ENTER 707;Rng$
PRINT Rng$
```

REFerence command/query

The :TIMEBASE:REFERENCE command sets the display reference to the left side of the screen, the right side of the screen, or to the center of the screen.

The REFERENCE query returns the current display reference.

Command Syntax: :TIMebase:REFerence {LEFT | CENTer | RIGHt}

Example:
```
OUTPUT 707;":TIMEBASE:REFERENCE LEFT"
```

Query Syntax: :TIMebase:REFerence?

Returned Format: [:TIMebase:REFerence] {LEFT | CENTer | RIGHt}<NL>

Example:
```
DIM Rf$[30]
OUTPUT 707;":TIMEBASE:REFERENCE?"
ENTER 707;Rf$
PRINT Rf$
```

WINDow

command/query

The :TIMEBASE:WINDOW command controls whether or not the second timebase is in use. If this command is set to ON the second timebase is displayed on the bottom half of the screen. When the Timebase Window is on, all measurements are taken on the second (expanded) timebase.

When the Timebase Window is on the second timebase data can be acquired over the HP-IB by sending the command WAVEFORM:DATA?, however care must be taken in order to ensure valid data is present when the data is requested.

The WINDOW query returns the current state of this command.

Command Syntax: :TIMebase:WINDow {{ON | 1} | {OFF | 0}}

Example:
```
OUTPUT 707;":TIM:WIND 1"
```

Query Syntax: :TIMbase:WINDow?

Returned Format: [:TIMebase:WINDow] {1 | 0}<NL>

Example:
```
DIM Tw$[50]
OUTPUT 707;":TIMEBASE:WINDOW?"
ENTER 707;Tw$
PRINT Tw$
```

WINDow:DELay (Position) command/query

The :TIMEBASE:WINDOW:DELAY command sets the timebase window delay. The window delay actually sets the position of the timebase window on the main sweep. The range for this command is determined by the main sweep seconds/division and delay values. The value for this command must keep the window on the main sweep display.

The WINDOW:DELAY query returns the current value.

Command Syntax: :TIMebase:WINDow:DELay <wid_delay>

Where:

<wid_delay> ::= time in seconds from trigger to on screen delay reference point. Maximum value depends on time/division setting.

Example:
```
OUTPUT 707;":TIM:WIND:DEL 20 NS"
```

Query Syntax: :TIMebase:WINDow:DELay?

Returned Format: [:TIMebase:WINDow:DELay] <wid_delay><NL>

Where:

<wid_delay> ::= current setting in seconds (exponential - NR3 format)

Example:
```
DIM Dly$[50]
OUTPUT 707;":TIMEBASE:WINDOW:DELAY?"
ENTER 707;Dly$
PRINT Dly$
```

WINDow:RANGe (Timebase) command/query

The :TIMEBASE:WINDOW:RANGE command sets the full scale horizontal time in seconds for the second (expanded) timebase. The RANGE value is ten times the time per division of the second timebase.

The WINDOW:RANGE query returns the current range value.

Command Syntax: :TIMebase:WINDow:RANGe <range>

Where:

<range> ::= 1 ns to 50 s (depends on main sweep setting)

Example:
```
OUTPUT 707;":TIM:WIND:RANG 100 NS"
```

Query Syntax: :TIMebase:WINDow:RANGe?

Returned Format: [:TIMebase:WINDow:RANGe] <range><NL>

Where:

<range> ::= 1 ns to 50 s (depends on main sweep setting) (exponential - NR3 format)

Example:
```
DIM Wrng$[50]
OUTPUT 707;":TIMEBASE:WINDOW:RANGE?"
ENTER 707;Wrng$
PRINT Wrng$
```

Trigger Subsystem 16

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 used in more than one of the trigger modes. If the command is a valid command for a trigger mode that setting will be accepted. If the command is not valid for a trigger mode an error will be generated.

You must be sure that the instrument is in the proper trigger mode for the command being sent. The instrument can be placed in the trigger mode from the front panel or over the HP-IB. 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. As an example, send:

```
":TRIGGER:MODE TV;LEVEL 200 MV"
```

This will place the instrument in the TV Trigger Mode and set the trigger level to 200 mV. This is necessary because the LEVEL command is also valid for the other trigger modes.

The Trigger modes are described on the next few pages prior to the command syntax. Table 16-1 lists the different TRIGGER subsystem commands that are available for each trigger mode.

Note

Auto or triggered mode is selected with the TIMEBASE:MODE command.

Table 16-1. Valid Commands for Specific Trigger Modes

EDGE	PATTERN	STATE	DELAY	TV
HOLDOFF LEVEL SLOPE SOURCE	CONDITION HOLDOFF LEVEL LOGIC PATH	CONDITION HOLDOFF LEVEL LOGIC PATH SLOPE SOURCE	CONDITION DELAY DELAY:SLOPE DELAY:SOURCE LEVEL LOGIC OCCURRENCE OCCURRENCE:SLOPE OCCURRENCE:SOURCE PATH QUALIFY SLOPE SOURCE	CONDITION FIELD HOLDOFF LEVEL LINE OCCURRENCE OCCURRENCE:SLOPE POLARITY QUALIFY SOURCE STANDARD

Note

Auto or triggered mode is selected with the TIMEBASE:MODE command.

The EDGE Trigger Mode

The Edge Trigger Mode is the easiest trigger mode to understand and use from the front panel or over the HP-IB. This is true because the Edge Trigger Mode has the least number of parameters to be set. This explanation of the trigger mode commands follows the front panel keys very closely. Refer to the *Front Panel Operation Reference* for further explanations of the trigger operation.

In this mode you must set the trigger source, using the TRIGGER SOURCE command. This selects the channel that the oscilloscope will trigger on. The argument for this command is channel1 through channel4.

The next thing that must be set in this mode is the trigger level. This value is set using the LEVEL command and can be set for each trigger source. The trigger level values that are set in this mode are used for all modes except TV Trigger Mode, or conversely, the LEVELS set in the PATTERN, STATE, or DELAY modes will set the EDGE LEVELS as well.

The trigger level is used in the PATTERN and STATE mode to define the voltage that determines if the input voltage is a logic high or a logic low for the logic triggers.

The next field to be set in the Edge Trigger Mode is the actual edge that will create the trigger. This command is the SLOPE command and can be set to POSITIVE or NEGATIVE for each of the sources.

The last setting in this mode is the Trigger Holdoff value. This value is only used for the EDGE mode.

The Pattern Trigger Mode

This description of the Pattern Trigger Mode is also related to the front panel keys. 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 mode. The one parameter that carriers over is LEVEL. If the level command is used in this mode it will also change the level value for that source in the Edge Trigger Mode.

The logic pattern for the Pattern Trigger Mode is set using the PATH and LOGIC commands. The PATH command specifies which of the four inputs is selected for the logic pattern. Once the path has been 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 next command sets the "when" field on the front panel. This is set with the CONDITION command. 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, LT, and RANGE.

When the command TRIGGER:CONDITION ENTER or TRIGGER:CONDITION EXIT is sent the Entered or Exited parameter is set on the front panel. When the 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,<range_low>,<range_high>.

Also, in the Pattern Trigger Mode, you can set the holdoff time using 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 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.

Next 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 could be sent using the following syntax ":TRIGGER:PATH CHAN2;:TRIGGER:LOGIC LOW", or ":TRIGGER:PATH CHAN2; LOGIC LOW".

The TRIGGER:CONDITION command in the State Trigger Mode will set the "is/is not present" state using the parameters TRUE for "is present" and FALSE for "is not present."

In this mode a holdoff value can be set as in most 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>, and 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" and to set the delay slope use the command "TRIGGER:DELAY:SLOPE POSITIVE".

Next (on the front panel) is the "trigger on" field. The values within this field are set with the OCCURRENCE command. To set the number of occurrences use the command syntax "TRIGGER:OCCURRENCE 3333". To set the source for the number of occurrences use the command syntax ":TRIGGER:OCCURRENCE:SOURCE CHANNEL2" and to set the slope of the trigger occurrence 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 will allow you to select one of the TV signal frames and one of the lines within that frame.

Once the TV Trigger Mode has been 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 source of the trigger signal and the trigger level.

The source is set by sending the SOURCE command. The SOURCE command allows the selection of channel 1 through 4 using the command ":TRIGGER:SOURCE CHANNEL2".

The trigger level is set by sending the command ":TRIGGER:LEVEL <value>".

With the standard set to 525 or 625 the commands that can be used are POLARITY, FIELD, and LINE. The POLARITY command can accept NEGATIVE and POSITIVE as its parameters and sets the edge for the trigger. The FIELD command uses 1 and 2 for its parameters which select the first or second field of the television signal. The LINE command parameters are different for the two standards, refer to the 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 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 uses the parameters HIGH and LOW.

The edge defined by the QUALIFY command must occur within the range of time values that are displayed in the next 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 "TRIGGER:CONDITION RANGE,<gt_value>, <lt_value>".

The next 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>" and to set the slope for the occurrences send the command ":TRIGGER:OCCURRENCE:SLOPE POSITIVE". The slope command can also use NEGATIVE as a parameter.

The description for each of the commands will tell you in which modes that command is valid.

See figure 16-1 for Trigger subsystem commands syntax diagram.

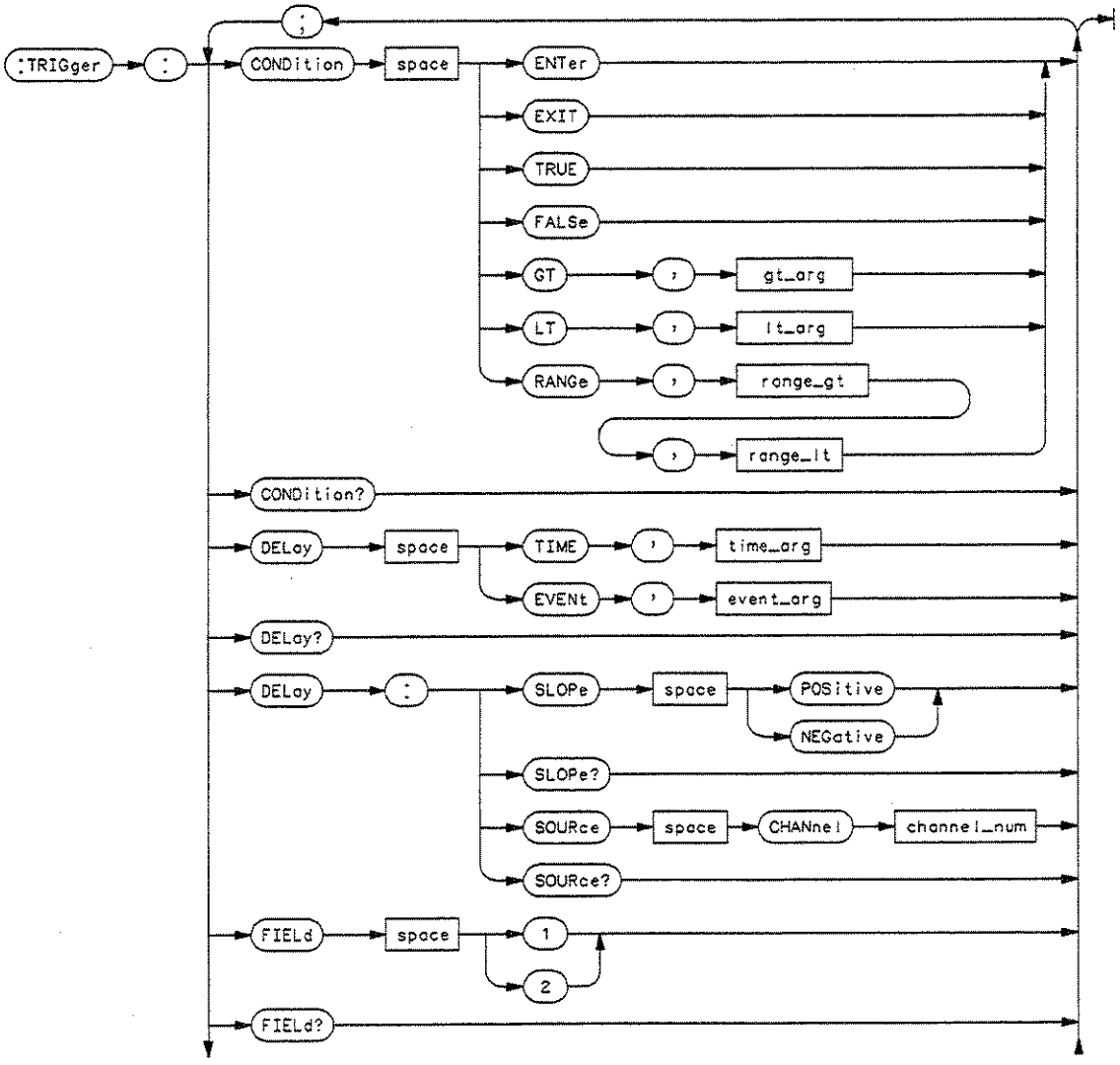


Figure 16-1. Trigger Subsystem Commands Syntax Diagram

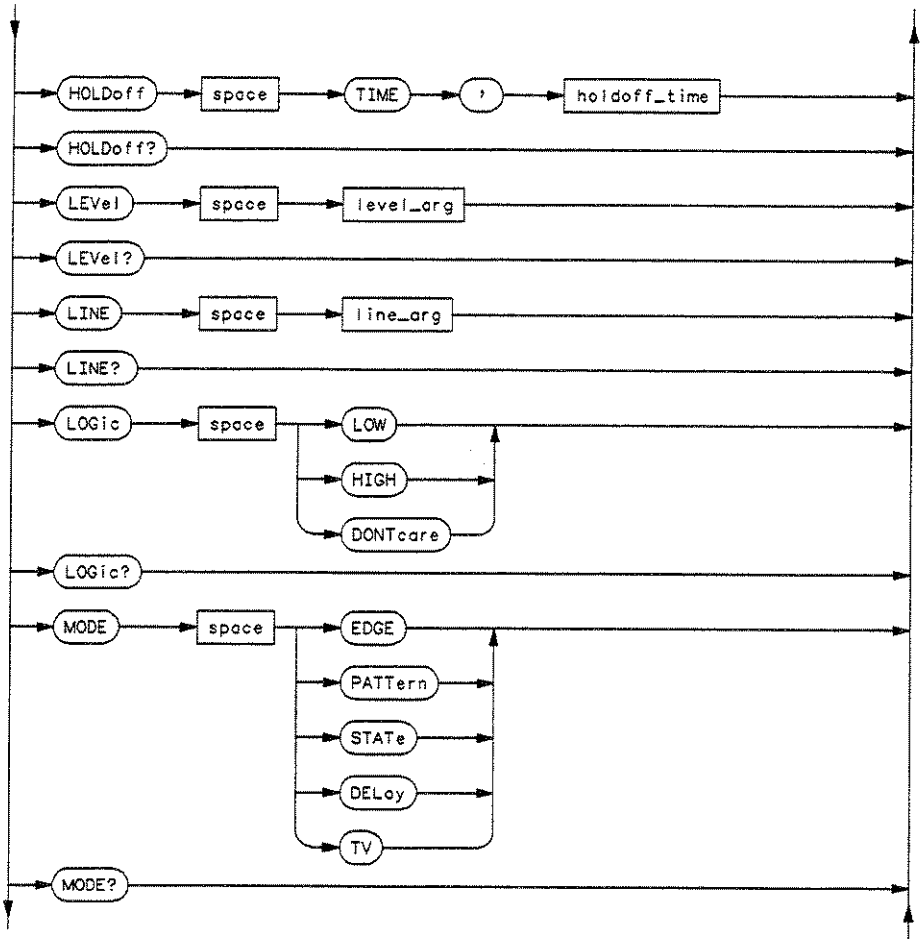


Figure 16-1. Trigger Subsystem Commands Syntax Diagram (continued)

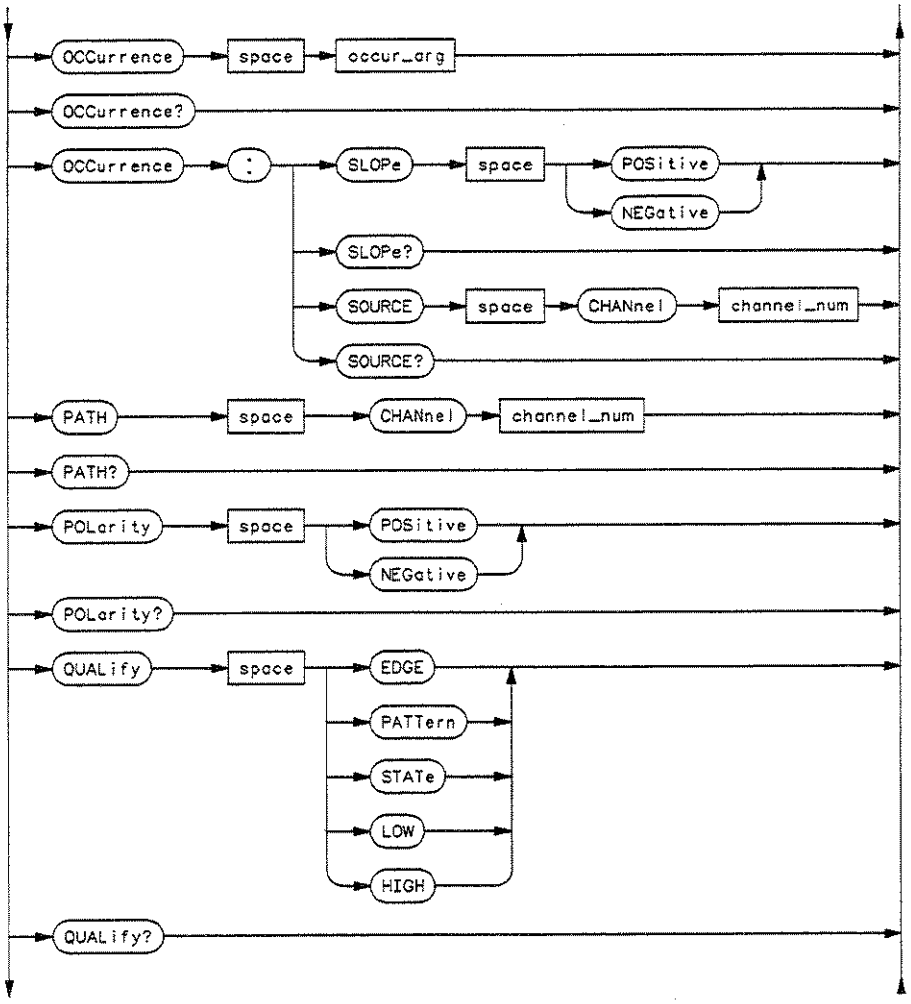
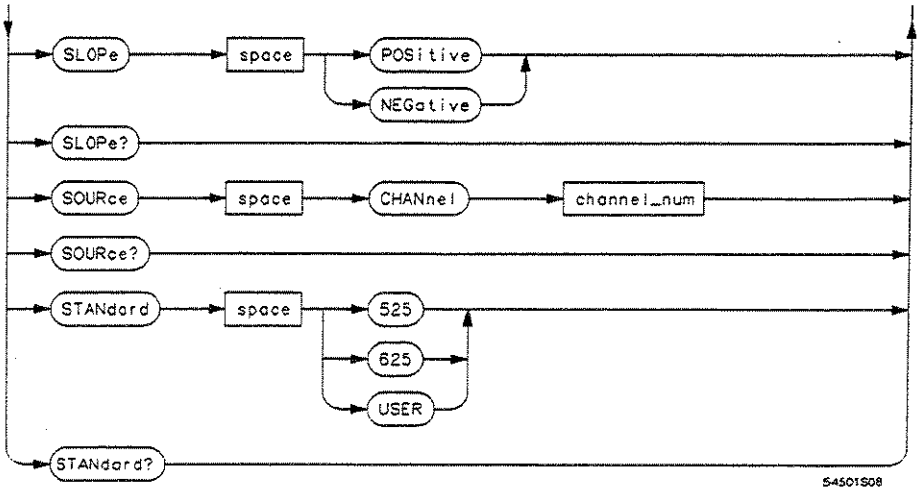


Figure 16-1. Trigger Subsystem Commands Syntax Diagram (continued)



channel_num = An integer, 1, 2, 3, or 4.

event_arg = An integer, 1 to 16000000.

gt_arg = A time value, 20 ns to 160 ms.

holdoff_time = A time value, 40 ns to 320 ms.

level_arg = A real number, specifying the trigger level in volts.

line_arg = An integer, 1 to 625 (depends on video STANDARD selected).

lt_arg = A time value, 20 ns to 160 ms.

range_gt = A time value, 20 ns to 159.999 ms (value must be less than range_lt).

range_lt = A time value, 30 ns to 160 ms (value must be greater than range_gt).

time_arg = A time value, 30 ns to 160 ms.

occur_arg = An integer, 1 to 16000000.

Figure 16-1. Trigger Subsystem Commands Syntax Diagram (continued)

CONDition

command/query

The :TRIGGER:CONDITION command is valid in the PATTERN, STATE, DELAY, and TV trigger modes. The function of the CONDITION command in each of these modes is described below.

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 if the trigger will be generated upon entry to the specified logic pattern, upon exiting the specified logic pattern, or if the pattern must be present for a specified amount of time. The time in the pattern trigger mode can be specified to be greater than a value (GT), less then a value (LT), or between two values (RANGE).

These are the same settings that are specified using the front panel "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 for the trigger to occur. This command is only valid in the "user defined" mode.

The CONDITION query returns the currently selected condition, for the currently selected mode.

CONDition

Command Syntax: :TRIGger:CONDition <argument>

Where in **PATTERN or DELAY:PATTERN** mode:

<argument> ::= {ENTer | EXIT | GT,<value> | LT,<value> | RANGe,<range_gt>,<range_lt>}

Where in **STATE or DELAY:STATE** mode:

<argument> ::= {TRUE | FALSe}

Where in **TV** mode:

<argument> ::= RANGe,<range_gt>,<range_lt>

Where:

<value> ::= 20 ns to 160 ms
<range_gt> ::= 20 ns to 159.999 ms (must be less than range_lt)
<range_lt> ::= 30 ns to 160 ms (must be greater than range_gt)

Example: OUTPUT 707;":TRIG:COND RANGE,22ms,33ms"

Query Syntax: :TRIGger:CONDition?

Returned Format: [:TRIGger:CONDition] <argument> <NL>

Where in **PATTERN** or **DELAY PATTERN** mode:

<argument> ::= {ENTer | EXIT | GT,<value> | LT,<value> | RANGe,<range_gt>,<range_lt>}

Where in **STATE** or **DELAY STATE** mode:

<argument> ::= {TRUE | FALSe}

Where in **TV** mode:

<argument> ::= RANGe,<range_gt>,<range_lt>

Where:

<value> ::= 20 ns to 160 ms
<range_gt> ::= 20 ns to 159.999 ms (must be less than range_lt)
<range_lt> ::= 30 ns to 160 ms (must be greater than range_gt)

Example:

```
DIM Con$[50]
OUTPUT 707;"CONDITION?"
ENTER 707;Con$
PRINT Con$
```

DELay

command/query

The :TRIGGER:DELAY command is valid only in the **Delay Trigger Mode.** This command allows you to set 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.

The Delay query returns the current delay setting.

Command Syntax: :TRIGger:DELay {TIME,<time_value> | EVENt,<event_value>}

Where:

<time_value> ::= time of delay from 30 ns to 160 ms
<event_value> ::= number of events from 1 to 16000000

Example:
```
OUTPUT 707;":TRIGGER:DELAY TIME,1.23E-01"
```

Query Syntax: :TRIGger:DELay?

Returned Format: [:TRIGger:DELay] {TIME,<time_value> | EVENt,<event_value>}<NL>

Where:

<time_value> ::= time of delay from 30 ns to 160 ms
<event_value> ::= number of events from 1 to 16000000

Example:
```
DIM Value$[50]
OUTPUT 707;":TRIG:DELAY?"
ENTER 707;Value$
PRINT Value$
```

DELay:SLOPe — command/query

The :TRIGGER:DELAY:SLOPE command sets the edge that will be counted by the delay command. The parameters for this command are NEGATIVE or POSITIVE. This command is valid in the **Delay Trigger Mode**.

The DELAY:SLOPE query returns the current delay slope.

Command Syntax: :TRIGger:DELay:SLOPe {POSitive | NEGative}

Example: OUTPUT 707;":TRIG:DEL:SLOP POS"

Query Syntax: :TRIGger:DELay:SLOPe?

Returned Format: [:TRIGger:DELay:SLOPe] {POSitive | NEGative} <NL>

Example:
```
DIM Tos$[50]
OUTPUT 707;"TRIGGER:DELAY:SLOP?"
ENTER 707;Tos$
PRINT Tos$
```

DELay:SOURce command/query

The :TRIGGER:DELAY:SOURCE command sets the edge that will be counted by the delay command. The parameters for this command are CHANNEL1 through CHANNEL4. This command is only valid in the **Delay Trigger Mode**.

The DELAY:SOURCE query returns the source of the delay in the Delay Trigger Mode.

Command Syntax: :TRIGger:DELay:SOURce {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example: OUTPUT 707;":TRIG:DEL:SOURCE CHANNEL2"

Query Syntax: :TRIGger:DELay:SOURce?

Returned Format: [:TRIGger:DELay:SOURce] {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4} <NL>

Example:
```
DIM Tos$[50]
OUTPUT 707;"TRIGGER:DELAY:SOUR?"
ENTER 707;Tos$
PRINT Tos$
```

FIELd

command/query

The :TRIGGER:FIELD command is only valid in the **TV Trigger Mode.** This command is used to select the field of the TV signal when the STANDARD is set to 525 or 625. The only valid parameters for this command are 1 or 2.

The FIELD query returns the current setting of the FIELD command.

Command Syntax: :TRIGger:FIELd { 1 | 2 }

Example:
```
OUTPUT 707;"TRIGGER:FIEL 2"
```

Query Syntax: :TRIGger:FIELd?

Returned Format: [:TRIGger:FIELd] { 1 | 2 }<NL>

Example:
```
DIM F$[50]
OUTPUT 707;":TRIG:FIELD?"
ENTER 707;"F$
PRINT F$
```

HOLDoff command/query

The :TRIGGER:HOLDOFF command is valid in the **Edge, Pattern, State, and TV Trigger Modes.** This command will allow a holdoff by time value to be entered.

The HOLDOFF query returns the value of the holdoff for the current mode.

Command Syntax: :TRIGger:HOLDoff TIME,<holdoff_value>

Where:

<holdoff_value> ::= 40 ns to 320 ms rounded to nearest 20 ns increment.

Examples:
```
OUTPUT 707;":TRIGGER:HOLDOFF TIME,216 US"
OUTPUT 707;":TRIGGER:HOLDOFF TIME,2.16E-4"
```

Query Syntax: :TRIGger:HOLDoff?

Returned Format: [:TRIGger:HOLDoff] <holdoff_value><NL>

Where:

<holdoff_value> ::= 40 ns to 320 ms (exponential - NR3 format)

Example:
```
DIM Ho$[50]
OUTPUT 707;":TRIGGER:HOLD?"
ENTER 707;Ho$
PRINT Ho$
```

LEVel command/query

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. If you are in the Pattern Trigger Mode and set a trigger level value, that level will also be used for the Edge, State, and Delay Trigger Modes.

The LEVEL query returns the trigger level of the current trigger mode.

Command Syntax: :TRIGger:LEVel <level>

Where:

<level> ::= -140.0 volts to +140.0 volts (maximum value depends on volts/division and offset settings).

Examples:
```
OUTPUT 707;":TRIGGER:LEVEL .30"
OUTPUT 707;":TRIGGER:LEV 300MV"
OUTPUT 707;":TRIG:LEV 3E-1"
```

Refer to chapter 3 for the syntax of using values with multipliers.

Query Syntax: :TRIGger:LEVel?

Returned Format: [:TRIGger:LEVel] <level><NL>

Where:

<level> ::= trigger level in volts (exponential - NR3 format)

Example:
```
DIM Tlevel$[30]
OUTPUT 707;":TRIGGER:LEVEL?"
ENTER 707;Tlevel$
PRINT Tlevel$
```

LINE

command/query

The :TRIGGER: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 will be set in that standard and selected field.

The LINE query returns the current line of the selected standard.

Command Syntax: :TRIGger:LINE <line_number>

Where:

<line_number> ::= 1 to 625 (depends on STANDARD and FIELD selection).

Example:
```
OUTPUT 707;":TRIG:LINE 22"
```

Query Syntax: :TRIGger:LINE?

Returned Format: [:TRIGger:LINE] <line_number><NL>

Where:

<line_number> ::= 1 to 625 (depends on STANDARD and FIELD selection).
(integer - NR1 format)

Example:
```
DIM Ln$[50]
OUTPUT 707;":TRIGGER:LINE?"
ENTER 707;Ln$
PRINT Ln$
```

LOGic

command/query

The :TRIGGER:LOGIC command is valid in the **Pattern and State Trigger Modes**, as well as the **DELAY Trigger Mode** when qualifing by PATTERN or STATE. The LOGIC command is used to specify the relation 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 it is less than the trigger level, it is considered LOW.

The LOGIC query returns the last specified logic level of the currently enabled path.

Command Syntax: :TRIGger:LOGic {HIGH | LOW | DONTcare}

Example:
```
OUTPUT 707;":TRIG:LOGIC DONT"
```

Query Syntax: :TRIGger:LOGic?

Returned Format: [:TRIGger:LOGic] {HIGH | LOW | DONTcare}<NL>

Example:
```
DIM L$[50]
OUTPUT 707;":TRIGGER:LOGIC?"
ENTER 707;L$
PRINT L$
```

MODE

command/query

The :TRIGGER:MODE command selects the trigger mode. The mode command can be sent from any trigger mode.

The MODE query returns the currently selected trigger mode.

Command Syntax: :TRIGger:MODE {EDGE | PATTern | STATe | DELay | TV}

Example: OUTPUT 707;":TRIGGER:MODE PATT"

Query Syntax: :TRIGger:MODE?

Returned Format: [:TRIGger:MODE] {EDGE | PATTern | STATe | DELay | TV}<NL>

Example:
```
DIM Mode$[50]
OUTPUT 707;":TRIGGER:MODE?"
ENTER 707;Mode$
PRINT Mode$
```

OCCurrence — command/query

The :TRIGGER:OCCURRENCE command sets the number of trigger events that must occur before the oscilloscope sweep is actually triggered. This command is valid in the **Delay Trigger Mode** and in the **TV Trigger Mode**.

The OCCURRENCE query returns the current value of occurrence if the oscilloscope is in the Delay Trigger Mode, or in the TV Trigger Mode with USER DEFINED selected.

Command Syntax: :TRIGger:OCCurrence <occ_number>

Where:

<occ_number> ::= 1 to 16000000

Example:
```
OUTPUT 707;":TRIGGER:OCC 14"
```

Query Syntax: :TRIGger:OCCurrence?

Returned Format: [:TRIGger:OCCurrence] <occ_number><NL>

Where:

<occ_number> ::= 1 to 16000000 (integer - NR1 format)

Example:
```
DIM Oc$[50]
OUTPUT 707;":TRIGGER:OCCURRENCE?"
ENTER 707;Oc$
PRINT Oc$
```

OCCurrence:SLOPe command/query

The :TRIGGER:OCCURRENCE:SLOPE command sets 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.**

The OCCURRENCE:SLOPE query returns the slope of the current mode.

Command Syntax: :TRIGger:OCCurrence:SLOPe {POSitive | NEGative}

Example:
```
OUTPUT 707;":TRIG:OCC:SLOP POS"
```

Query Syntax: :TRIGger:OCCurrence:SLOPe?

Returned Format: [:TRIGger:OCCurrence:SLOPe] {POSitive | NEGative} < NL >

Example:
```
DIM Tos$[50]
OUTPUT 707;"TRIGGER:OCCURRENCE:SLOP?"
ENTER 707;Tos$
PRINT Tos$
```

OCCurrence:SOURce command/query

The :TRIGGER:OCCURRENCE:SOURCE command sets the edge that will be counted by the occurrence command. The parameters for this command are CHANNEL1 through CHANNEL4. This command is valid in the **Delay Trigger Mode.**

The OCCURRENCE:SOURCE query returns the source of the occurrence in the Delay Trigger Mode.

Command Syntax: :TRIGger:OCCurrence:SOURce {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example:
```
OUTPUT 707;":TRIG:OCC:SOURCE CHANNEL2"
```

Query Syntax: :TRIGger:OCCurrence:SOURce?

Returned Format: [:TRIGger:OCCurrence:SOURce] {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4} < NL >

Example:
```
DIM Tos$[50]
OUTPUT 707;"TRIGGER:OCCURRENCE:SOUR?"
ENTER 707;Tos$
PRINT Tos$
```

PATH

command/query

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 logic commands.

The PATH query returns the current trigger source of the present mode.

Command Syntax: :TRIGger:PATH <path_name>

Where:

<path_name> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example:
```
OUTPUT 707;":TRIGGER:PATH CHANNEL2"
```

Query Syntax: :TRIGger:PATH?

Returned Format: [:TRIGger:PATH] CHANnel{1 | 2 | 3 | 4}<NL>

Example:
```
DIM Tp$[50]
OUTPUT 707;":TRIG:PATH?"
ENTER 707;Tp$
PRINT Tp$
```

POLarity — command/query

The :TRIGGER:POLARITY command is valid in the **TV Trigger Mode.** It 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.

The POLARITY query will return the current polarity setting.

Command Syntax: :TRIGger:POLarity {POSitive | NEGative}

Example:
```
OUTPUT 707;":TRIGGER:POL NEGATIVE"
```

Query Syntax: :TRIGger:POLarity?

Returned Format: [:TRIGger:POLarity] {POSitive | NEGative}<NL>

Example:
```
DIM Tp$[50]
OUTPUT 707;":TRIG:POL?"
ENTER 707;Tp$
PRINT Tp$
```

QUALify command/query

The :TRIGGER:QUALIFY command is valid in the **Delay and TV Trigger Mode.** The parameters for this command when in the Delay Trigger Mode are:

- EDGE
- PATTERN
- STATE

The parameters for this command when in the TV Trigger Mode are:

- LOW
- HIGH

The QUALIFY query returns the current setting of the QUALIFY command in the currently selected mode.

Command Syntax: :TRIGger:QUALify <qualify_parameter>

Where in **Delay Trigger Mode**:

<qualify_parameter> ::= {EDGE | PATTern | STATe}

Where in **TV Trigger Mode**:

<qualify_parameter> ::= {LOW | HIGH}

Example: OUTPUT 707;":TRIGGER:QUALIFY PATT"

Query Syntax: :TRIGger:QUALify?

Returned Format: [:TRIGger:QUALify] {EDGE | PATTern | STATe | LOW | HIGH}<NL>

Example:

```
DIM Tq$[50]
OUTPUT 707;":TRIG:QUALIFY?"
ENTER 707;Tq$
PRINT Tq$
```

SLOPe

command/query

The :TRIGGER:SLOPE command specifies the slope of the edge for the trigger. 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.

The SLOPE query returns the current slope for the currently selected trigger mode.

Command Syntax: :TRIGger:SLOPe {NEGative | POSitive}

Example:
```
OUTPUT 707;":TRIGGER:SLOPE POSITIVE"
```

Query Syntax: :TRIGger:SLOPe?

Returned Format: [:TRIGger:SLOPe] {POSitive | NEGative} < NL >

Example:
```
DIM Ts$[50]
OUTPUT 707;":TRIG:SLOP?"
ENTER 707;Ts$
PRINT Ts$
```

SOURce — command/query

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.

The SOURCE query returns current source for the selected trigger mode.

Command Syntax: :TRIGger:SOURce {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}

Example:
```
OUTPUT 707;":TRIGGER:SOURCE CHAN2"
```

Query Syntax: :TRIGger:SOURce?

Returned Format: [:TRIGger:SOURce] {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4}<NL>

Example:
```
DIM Src$[30]
OUTPUT 707;":TRIGGER:SOURCE?"
ENTER 707;Src$
PRINT Src$
```

STANdard

command/query

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 (defined).

The STANDARD query returns the currently selected standard.

Command Syntax: :TRIGger:STANdard {525 | 625 | USER}

Example: OUTPUT 707;":TRIGGER:STAN USER"

Query Syntax: :TRIGger:STANdard?

Returned Format: [:TRIGger:STANdard] {525 | 625 | USER}<NL>

Example:
```
DIM Ts$[50]
OUTPUT 707;":TRIG:STANDARD?"
ENTER 707;Ts$
PRINT Ts$
```

Waveform Subsystem 17

Introduction

The WAVEFORM subsystem is used to transfer waveform data between a controller and the HP 54501A's waveform memories. 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. This 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 the front panel store key is pressed. The Preamble values are based on the settings of variables in the ACQUIRE subsystem or the front panel setup if the store key is pressed. Although the preamble values can be changed with a controller, the way the data was 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, extreme caution must be used when changing any waveform preamble values to ensure the data will still be useful. For example, setting POINTS in the preamble to a value different from the actual number of points in the waveform will result in inaccurate data.

The waveform data and preamble must be read (by the controller) or sent (to the HP 54501A) with two separate commands, DATA and PREAMBLE.

Sending consecutive :DIGITIZE commands may improve the data throughput. Refer to the Root Level Command :DIGITIZE for more information.

Data Acquisition Types

There are three types of waveform acquisition that can be selected with the :ACQUIRE:TYPE command. The three types are NORMAL, AVERAGE, and ENVELOPE. When the data is acquired using the DIGITIZE command the data is placed in the channel buffer of the specified source.

After a DIGITIZE command the instrument is stopped. If the instrument is restarted, over the HP-IB or the Front Panel, the data acquired with the DIGITIZE command will be overwritten.

Note

The on-screen time is divided into a specific number of horizontal time points as defined by the :ACQUIRE:POINTS command. Each of these increments in time is referred to as a time bucket with each time bucket having a fixed time associated with it.

Normal

Normal data consists of the last data point (hit) in each time bucket. This data is transmitted over HP-IB 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. Time buckets that don't have data in them return -1. Only the magnitude values of each data point are transmitted, the time values correspond to the position in the data array. The first voltage value corresponds to the first time bucket on the left of the CRT and the last value corresponds to the next to last time bucket on the right side of the CRT.

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. Again, if a time bucket doesn't have any data in it, it will return -1. This data is transmitted over the HP-IB 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 is one point away from the right side of the screen.

Envelope Envelope data consists of two arrays of data, one containing the minimum of the first n hits in each time bucket and the other containing the maximum of the first n hits in each time bucket, where n is the value returned by the ACQUIRE:COUNT query. 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 HP-IB linearly, starting with time bucket 0 (on the left side of the CRT) 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 CRT. The last value is one data point away from the right side of the CRT.

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 3 , if WMEMORY 1 is specified as the source, or Waveform Memories 2 and 4 if WMEMORY 2 is specified as the source. The data is then 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 3. If waveform Memory 2 is specified as the source, the first array is transferred in Waveform Memory 2 and the second array is transferred into Waveform Memory 4. The data type is then changed to normal for each of the waveform memories.

Data Conversion

Data sent from the HP 54501A is raw data and 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 formula to convert a data value from waveform memories 1-4 to a voltage value is:

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 third data point sent with XORIGIN = 16 ns, XREFERENCE = 0, and XINCREMENT = 2 ns. Using the formula:

time = [(data point number - xreference)*xincrement] + xorigin

would result in the following calculation:

time = [(3 - 0) * 2 ns] + 16 ns = 22 ns.

Data Format for HP-IB Transfer

There are four formats for transferring waveform data over the HP-IB. These formats are 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 this format, the ASCII character string "#8<DD...D>" is sent before the actual data. The 8 indicates how many <D>'s will follow. The <D>'s are ASCII numbers, which indicate how many data bytes will follow.

For example, if 512 points were acquired the Block Header "#800000512" would be sent. The 8 indicates that eight length bytes follow, 512 indicates that 512 data bytes (binary) 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, it will be 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

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, it is represented by a value of -1.

BYTE formatted data will transfer over the HP-IB 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. 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 will give greater vertical precision than BYTE formatted data, with faster transfer times than WORD formatted data, but will probably require more time once transferred to be unpacked.

ASCII Format

ASCII formatted waveform records 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 HP-IB.

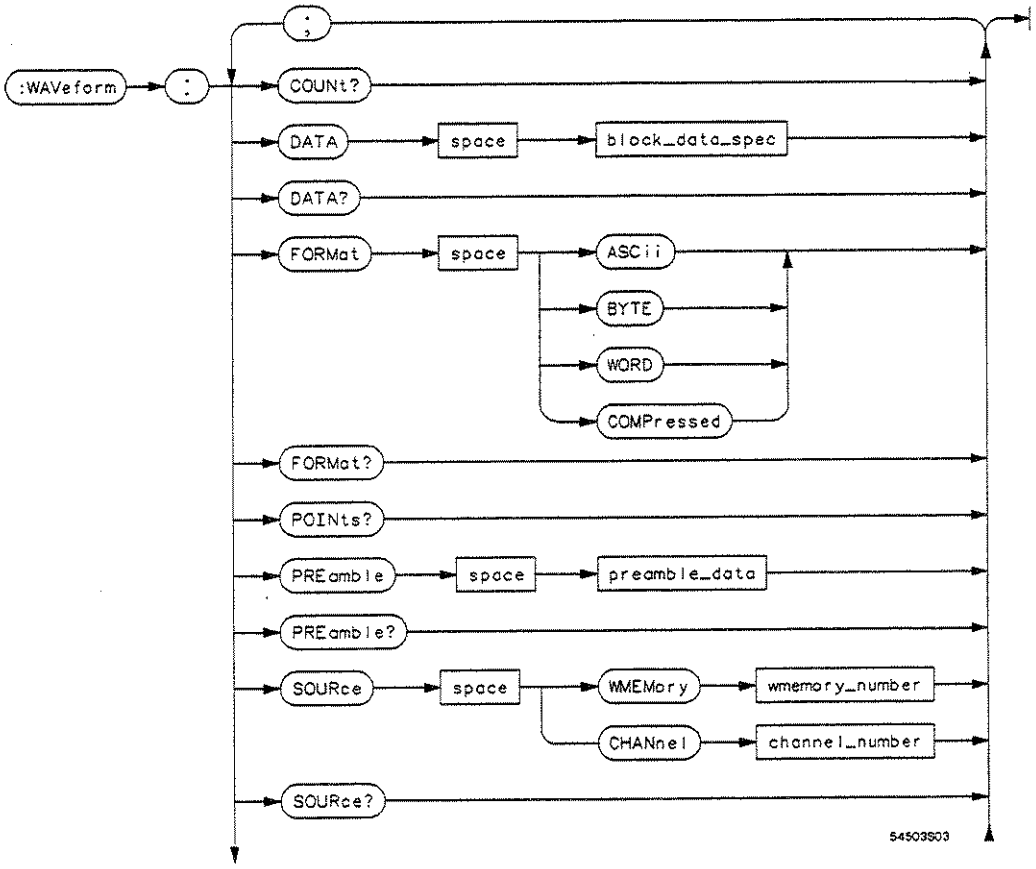
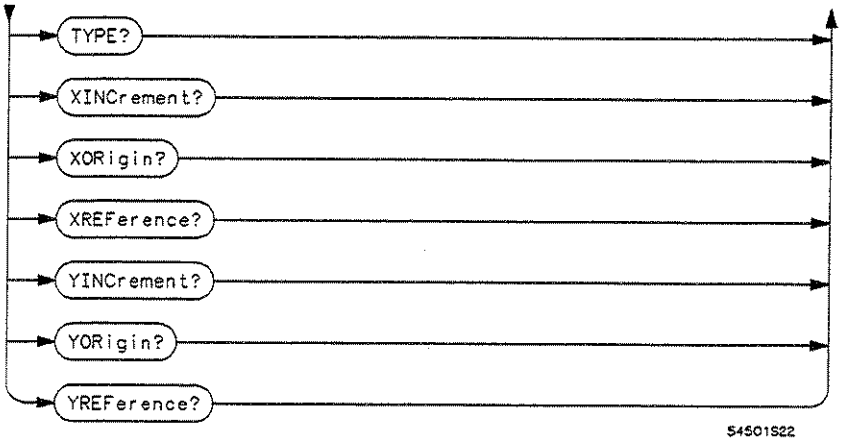


Figure 17-1. Waveform Subsystem Commands Syntax Diagram



channel_number = 1, 2, 3, or 4.

block_data_spec = A block of data in # format.

preamble_data = Refer to PREAMBLE command.

wmemory_number = An integer 1 through 4.

Figure 17-1. Waveform Subsystem Commands Syntax Diagram (continued)

COUNt query

The :WAVEFORM:COUNT query will always return a 1 in this instrument. This query is only included, in this instrument, for compatibility with other Hewlett-Packard instruments.

The value returned for this query has no meaning for the HP 54501A.

Query Syntax: :WAVeform:COUNt?

Returned Format: [:WAVeform:COUNt] 1<NL>

Example:
```
DIM Cnt$[50]
OUTPUT 707;":WAVEFORM:COUNT?"
ENTER 707;Cnt$
PRINT Cnt$
```

DATA

command/query

The :WAVEFORM:DATA command causes the instrument to accept a waveform data record over the HP-IB and store it in the previously specified waveform memory. The waveform memory is specified with a WAVEFORM:SOURCE command. Only waveform memories may have waveform data sent to them.

Note

The format of the data being sent must match the format previously specified by the waveforms preamble for the destination memory.

The DATA query tells the instrument to output the waveform record stored in the waveform memory or channel buffer, previously specified with a :WAVEFORM:SOURCE command, over the HP-IB.

Command Syntax: :WAVeform:DATA <binary block data in # format>

Example: OUTPUT 707;":WAV:DATA"

Query Syntax: :WAVeform:DATA?

Returned Format: [:WAVeform:DATA] <binary block length bytes> <binary block> <NL>

The following program moves data from the HP 54501A to the controller and then back to the HP 54501A with the :WAVEFORM:DATA query and command.

Example:

```
10 CLEAR 707
20                                          ! SET UP ACQUIRE SUBSYSTEM
30 OUTPUT 707;":ACQUIRE:TYPE NORMAL;COUNT 1;POINTS 512"
40 OUTPUT 707;":DIGITIZE CHANNEL1"          ! STORE CHAN 1 DISPLAY TO WMEM1
50 OUTPUT 707;":SYSTEM:HEADER OFF;:EOI ON"
60 OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1;FORMAT WORD" !SELECT WAVEFORM DATA
70                                          ! SOURCE AND OUTPUT FORMAT
80 OUTPUT 707;":WAVEFORM:DATA?"
90 ENTER 707 USING "#,2A,8D";Headers$,Bytes ! READ LENGTH BYTE
100 Length = Bytes
110 Length = Length/2
120 ALLOCATE INTEGER Waveform(1:Length)
130 ENTER 707 USING "#,W";Waveform(*)       ! ENTER WAVEFORM DATA TO INTEGER ARRAY
140 ENTER 707 USING "-K,B";End$             ! ENTER TERMINATOR
150 DIM Preamble$[200]
160 OUTPUT 707;":WAV:PREAMBLE?"             ! OUTPUT WAVE SOURCE PREAMBLE TO CONTROLLER
170 ENTER 707 USING "-K";Preamble$          ! ENTER PREAMBLE INTO CONTROLLER
180 OUTPUT 707;":WAV:SOURCE WMEMORY4"       !CHANGE SOURCE TO WMEMORY 4
190 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Preamble$! SEND PREAMBLE FROM
200                                         ! CONTROLLER TO WMEMORY 4
210 OUTPUT 707 USING "#,k";":WAVEFORM:DATA #800001024" !SEND HEADER
220 OUTPUT 707 USING "W";WAVEFORM(*)        ! SEND WAVEFORM DATA TO WMEMORY 4
220 OUTPUT 707;":BLANK CHANNEL1;VIEW WMEMORY4"! TURN CHAN 1 OFF - WMEM 4 ON
230 END
```

Note

In program line 190, the space after :WAVEFORM:PREAMBLE and before the quote mark is necessary.

FORMat command/query

The :WAVEFORM:FORMAT command sets the data transmission mode for waveform data output. This command controls how the data is formatted on the HP-IB when sent from the HP 54501A.

When the ASCII mode is selected, the data is ASCII digits with each data value separated by a comma.

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 transfered as 8-bit bytes.

The FORMAT query returns the current output format for transfer of waveform data.

Command Syntax: :WAVeform:FORMat {ASCii | WORD | BYTE | COMPressed}

Example:
```
OUTPUT 707;":WAV:FORMAT WORD"
```

Query Syntax: :WAVeform:FORMat?

Returned Format: [:WAVeform:FORMat] <mode> <NL>

Where:

<mode> ::= {ASCii | WORD | BYTE | COMPressed}

Example:
```
DIM Frmt$[30]
OUTPUT 707;":WAV:FORMAT?"
ENTER 707;Frmt$
PRINT Frmt$
```

POINts

query

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.

In most cases the number of time buckets actually acquired will be the number of points set in the ACQUIRE subsystem. There are some sweep speeds where the actual number of points will be less than requested. These are shown below.

With the sweep speed set to 2 ns per division the number of points actually acquired will be 32, 64, 128, or 200.

With the sweep speed set to 5 ns per division the number of points actually acquired will be 32, 64, 128, 256, or 500.

With the sweep speed set to 10 ns per division the number of points actually acquired will be 32, 64, 128, 256, 500, 512, or 1000.

Query Syntax: :WAVeform:POINts?

Returned Format: [:WAVeform:POINts] <value><NL>

Where:

<value> ::= number of acquired data points (integer - NR1 format)

Example:

```
Dim Pts$[50]
OUTPUT 707;":WAV:POINTS?"
ENTER 707;Pts$
PRINT Pts$
```

PREamble

command/query

The :WAVEFORM:PREAMBLE command sends a waveform preamble to the previously selected waveform memory in the instrument.

The PREAMBLE query sends a waveform preamble to the controller from the waveform source.

Command Syntax: :WAVeform:PREamble <preamble block>

Where:

<preamble block> ::= <format NR1>,<type NR1>,<points NR1>,<count NR1>,<xincrement NR3>,<xorigin NR3>,<xreference NR3>,<yincrement NR3>,<yorigin NR3>,<yreference NR3>

Query Syntax: :WAVeform:PREamble?

Returned Format: [:WAVeform:PREamble] <preamble block><NL>

Where:

<preamble block> ::= <format NR1>,<type NR1>, <points NR1>,<count NR1>,<xincrement NR3>,<xorigin NR3>,<xreference NR1>,<yincrement NR3>,<yorigin NR3>,<yreference NR1>

Where:

<format> ::= 0 for ASCII format
1 for BYTE format
2 for WORD format
4 for COMPRESSED format

<type> ::= 1 for NORMAL type
2 for AVERAGE type
3 for ENVELOPE type

Example: 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 the previously selected waveform memory.

```
10 DIM Pre$[120]
20 OUTPUT 707;"SYSTEM:HEADER OFF"
30 OUTPUT 707;"WAVEFORM:PREAMBLE?"
40 ENTER 707 USING "-K";Pre$
50 OUTPUT 707 USING "#,K";"WAV:PREAMBLE ";Pre$
60 END
```

Note

In line 50 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 will produce a word that is not a proper command word.

Example: The following program example brings the preamble in a numeric array.

```
10 DIM Preamble[1:10]
20 OUTPUT 707;"SYSTEM:HEADER OFF"
30 OUTPUT 707;"WAVEFORM:PREAMBLE?"
40 ENTER 707 ;Preamble(*)
50 OUTPUT 707;"WAV:PREAMBLE ";Preamble(*)
60 END
```

SOURce command/query

The :WAVEFORM:SOURCE command selects the channel or waveform memory to be used as the source for the waveform commands.

The SOURCE query returns the currently selected source for the waveform commands.

Command Syntax: :WAVeform:SOURce {CHANnel{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

Example:
```
OUTPUT 707;":WAV:SOURCE WMEMORY3"
```

Query Syntax: :WAVeform:SOURce?

Returned Format: [:WAVeform:SOURce] <source> <NL>

Where:

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

Example:
```
DIM Src$[30]
OUTPUT 707;":WAVEFORM:SOURCE?"
ENTER 707;Src$
PRINT Src$
```

TYPE

query

The :WAVEFORM:TYPE query returns the data type for the previously specified waveform source.

Query Syntax: :WAVeform:TYPE?

Returned Format: [:WAVeform:TYPE] <mode> <NL>

Where:

<mode> ::= {AVERage | ENVelope | NORMal}

Example:

```
DIM Typ$[30]
OUTPUT 707;":WAVEFORM:TYPE?"
ENTER 707;Typ$
PRINT Typ$
```

XINCrement

query

The :WAVEFORM:XINCREMENT query returns the x-increment value currently in the preamble for the current specified source. This value is the time difference between consecutive data points for NORMAL, AVERAGE, or ENVELOPE data.

Query Syntax: :WAVeform:XINCrement?

Returned Format: [:WAVeform:XINCrement] <value><NL>

Where:

<value> ::= x-increment in the current preamble (exponential - NR3 format)

Example:

```
DIM Xin$[50]
OUTPUT 707;":WAV:XINCREMENT?"
ENTER 707;Xin$
PRINT Xin$
```

XORigin

query

The :WAVEFORM:XORIGIN query returns the x-origin value currently in the preamble for the current specified source. This value is the time of the first data point in the memory with respect to the trigger point.

Query Syntax: :WAVeform:XORigin?

Returned Format: [:WAVeform:XORigin] <value> <NL>

Where:

<value> ::= x-origin value currently in preamble (exponential - NR3 format)

Example:

```
DIM Xr$[50]
OUTPUT 707;":WAV:XORIGIN?"
ENTER 707;Xr$
PRINT Xr$
```

XREFerence

query

The :WAVEFORM:XREFERENCE query returns the current x-reference value in the preamble for the current specified source. This value specifies the data point associated with the x-origin data value. For the HP 54501A this value is always zero.

Query Syntax: :WAVeform:XREFerence?

Returned Format: [:WAVeform:XREFerence] <value> <NL>

Where:

<value> ::= x-reference value in the current preamble, always 0
(integer - NR1 format)

Example:
```
DIM Xrf$[50]
OUTPUT 707;":WAV:XREFERENCE?"
ENTER 707;Xrf$
PRINT Xrf$
```

YINCrement query

The :WAVEFORM:YINCREMENT query returns the y-increment value currently in the preamble for the current specified source. This value is the voltage difference between consecutive data points.

Query Syntax: :WAVeform:YINCrement?

Returned Format: [:WAVeform:YINCrement] <value> <NL>

Where:

<value> ::= y-increment value in the current preamble (exponential - NR3 format)

Example:
```
DIM Yin$[50]
OUTPUT 707;":WAV:YINCREMENT?"
ENTER 707;Yin$
PRINT Yin$
```

YORigin

query

The :WAVEFORM:YORIGIN query returns the y-origin currently in the preamble for the current specified source. This value is the voltage at center screen.

Query Syntax: :WAVeform:YORigin?

Returned Format: [:WAVeform:YORigin] <value><NL>

Where:

<value> ::= y-origin in the current preamble (exponential -NR3 format)

Example:

```
Dim Yr$[50]
OUTPUT 707;":WAV:YORIGIN?"
ENTER 707;Yr$
PRINT Yr$
```

YREFerence query

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.

Query Syntax: :WAVeform:YREFerence?

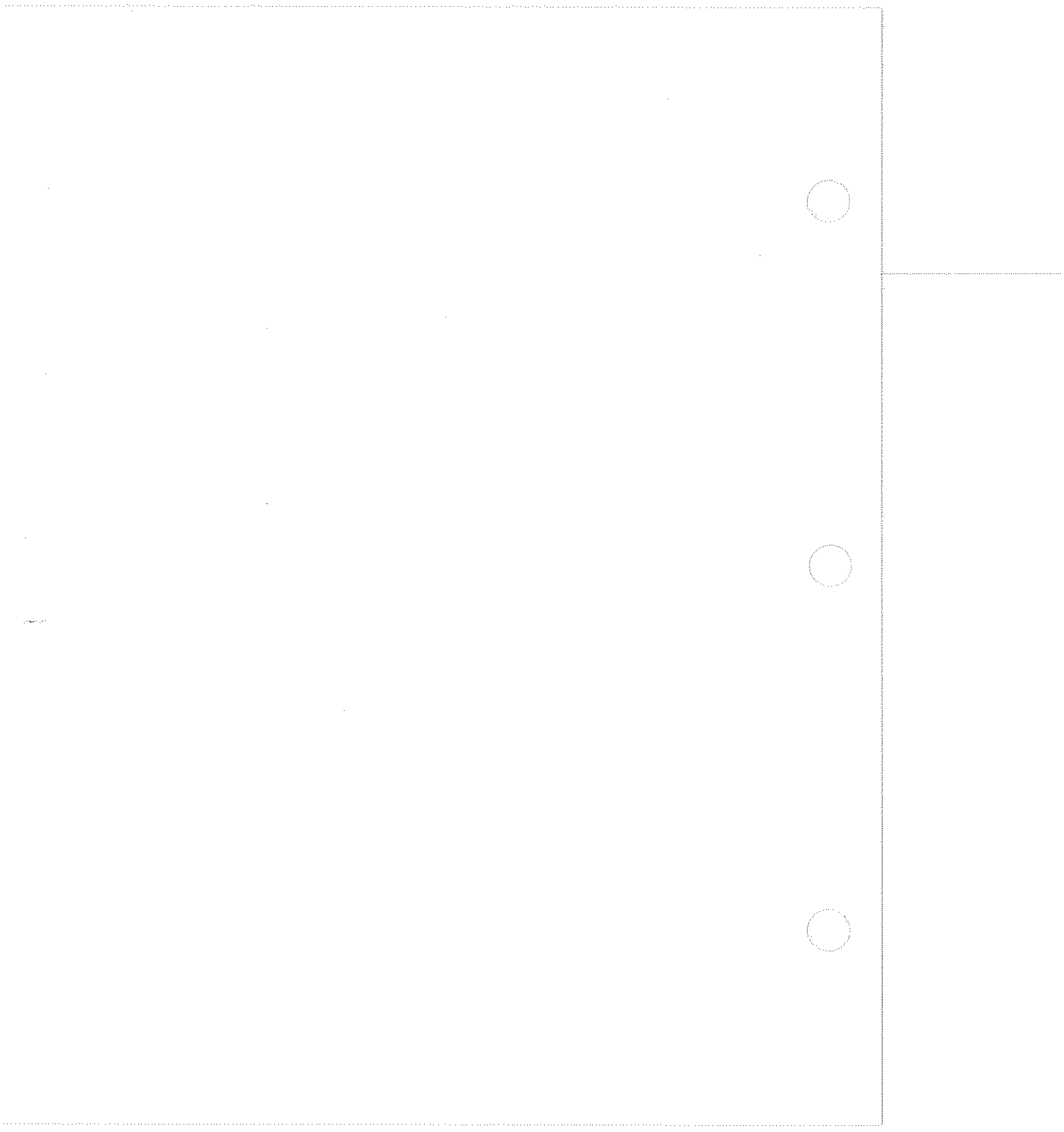
Returned Format: [:WAVeform:YREFerence] <value> <NL>

Where:

<value> ::= y-reference value in the current preamble (integer - NR1 format)

Example:

```
DIM Yrf$[50]
OUTPUT 707;"WAV:YREFERENCE?"
ENTER 707;Yrf$
PRINT Yrf$
```



Algorithms A

Introduction

One of the HP 54501A's primary features is its ability to make automatic measurements on displayed waveforms. This appendix 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. That is:

- 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 risetime measurements and all other edge measurements
- the trailing edge of the waveform must be displayed for falltime 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 will display ≤ 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 HP 54501A 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 risetime and falltime 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.

Automatic 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 the 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 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 crossing the upper threshold.

Note

Most time measurements are made based on the position of the first crossing of the middle threshold.

Algorithm Definitions

Following are the definitions that all measurements are based on:

delay There are three types of delay measurement:

- jitter
- standard
- user-defined

Jitter occurs only under the following circumstances:

- standard/user-defined key is set to standard
- two delay parameters are the same
- display mode is 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 when in the standard mode (not user-defined) and is not a jitter measurement.

standard delay = mid-threshold of the first edge of second parameter minus mid-threshold of the first edge of the first parameter

Note

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 - mid-threshold crossing of first rising edge

else
+ width = mid-threshold crossing of second falling edge - mid-threshold crossing of first rising edge

User-defined is the same as Standard definition except user-defined threshold.

- width The - width algorithm has standard and user-defined considerations:

if
first edge is rising

then
- width = second rising edge - first falling edge

else
- width = first rising edge - first falling edge

Period

if
first edge is rising

then
period = second rising edge - first rising edge

else
period = second falling edge - first falling edge

Frequency **frequency = 1/period**

Duty Cycle **duty cycle = (+ width/period) * 100**

Note

+ width is always calculated using mid-threshold.

Risetime **risetime = time at upper threshold - time at lower threshold**

Falltime **falltime = time at lower threshold - time at upper threshold**

V_{max} **V_{max} = voltage of the maximum point on screen**

V_{min} **V_{min} = voltage of the minimum point on screen**

V_{p-p} **$V_{p-p} = V_{max} - V_{min}$**

V_{top} **V_{top} = most prevalent point above waveform midpoint**

V_{base} V_{base} = **most prevalent point below waveform midpoint**

V_{amp} $V_{amp} = V_{top} - V_{base}$

V_{avg} Average voltage of the first cycle of the displayed signal is measured. If a complete cycle is not present the oscilloscope will average all data points.

V_{rms} The rms voltage of the first cycle of the displayed signal is measured. If a complete cycle is not present, the measurement will compute rms on all data points.

$$Vrms\ (ac) = \{ 1/n\ \Sigma^{n}_{j=1} V_j^2 - 1/n\ \Sigma^{n}_{j=1} Vj \}^{1/2}$$

Example Programs B

Introduction

This appendix contains example programs using the command set for the HP 54501A. In general, they use the long form of the command with alpha (as opposed to numeric) arguments with each command having a separate output statement for clarity. To optimize speed, switch to the concatenated short form numerics.

Throughout these examples, the HP 54501A is assumed to be at address 7, the hardcopy devices at address 1, and the system bus at 700. The input signal used is the AC CAL signal from the rear panel of the instrument. This signal is connected to channel 1 with a 10:1 probe.

All programs were developed on an HP Series 200/300 controller using HP BASIC 4.0. Several examples use the BASIC command "ENTER 2." This pauses program execution until the "ENTER" key is depressed on the controller. This is used to separate different blocks in the example to dramatize the features, allow for user interaction, or to wait for the HP 54501A to finish an operation such as a hardcopy output or an acquisition.

Vertical Channel Setup Program

This sample program demonstrates some of the commands used to set a vertical channel, in this case channel 1. The program assumes that the AC CAL signal from the rear panel of the instrument is connected to channel 1 through a 10:1 probe. The program sets the probe attenuation factor for channel 1 to 10:1.

```
10   CLEAR 707                                          !Device clear command.
20                                                      !initializes HP-IB registers.
30   !
40   OUTPUT 707;":EOI ON"                               !Turn on EOI
50   OUTPUT 707;"*RST"                                  !Reset instrument to
60                                                      !known state
70   OUTPUT 707;":AUTOSCALE"                            !Autoscales the unit
80   WAIT 3
90   OUTPUT 707;":BNC PROBE"                            !BNC output to probe mode
100  OUTPUT 707;":ACQUIRE:TYPE NORMAL"                  !Real time acquisition
110  OUTPUT 707;":CHANNEL1:PROBE 10"                    !Set probe attenuation to 10:1
120  OUTPUT 707;":CHANNEL1:RANGE .5"                    !Set vertical range to 500 mV
130  OUTPUT 707;":CHANNEL1:OFFSET .6"                   !Set offset to 0.6 V
140  OUTPUT 707;":CHANNEL1:COUPLING DC"                 !Set coupling to DC
150  REAL Offset,Range                                  !Set up Offset and Range
160                                                     !as variables
170  INTEGER J                                          !Set up J as variable
180  Offset=.6                                          !Set Offset variable to 0.6 V
190  OUTPUT 707;":BEEPER"
200  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - OFFSET POSITIONS WAVEFORM ON SCREEN'"
210  ENTER 2
220  OUTPUT 707;":SYSTEM:DSP ''"
230  !
240  !The following lines vary the channel 1 offset
250  !
260  FOR J=1 TO 18
270    OUTPUT 707;":CHANNEL1:OFFSET ";Offset"           !Sets next offset
280    Offset=Offset-.1
290    WAIT .5
300  NEXT J
310  !
320  OUTPUT 707;":BEEPER"
330  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - VERTICAL RANGE SCALES SIGNAL'"
340  ENTER 2
350  OUTPUT 707;":SYSTEM:DSP ''"
```

```
360  !
370  OUTPUT 707;":CHANNEL1:OFFSET -.4"                    !Center screen at -400 mV
380  OUTPUT 707;":CHANNEL1:RANGE .88"                     !Set vertical range to 880 mV
390  Range=.88                                            !Sets Range variable to 880 mV
400  !
410  !The following lines vary the vertical range
420  !
430  FOR J=1 TO 35
440    OUTPUT 707;":CHANNEL1:RANGE ";Range                !Sets new range
450    Range=Range+.05
460    WAIT .3
470  NEXT J
480  !
490  !
500  OUTPUT 707;":SYSTEM:DSP 'END OF PROGRAM'"
510  WAIT 15
520  OUTPUT 707;":SYSTEM:DSP ''"
530  LOCAL 707                                            !Returns instrument to local
540  END
```

Timebase Program

This sample program demonstrates some of the commands in the TIMEBASE subsystem. The program assumes that the AC CAL signal from the rear panel of the instrument is connected to channel 1 through a 10:1 probe.

```
10   CLEAR 707                                          !Device clear command.
20                                                      !initializes HP-IB registers.
30   !
40   OUTPUT 707;":EOI ON"                               !Turn on EOI
50   OUTPUT 707;"*RST"                                  !Reset instrument to
60                                                      !known state
70   OUTPUT 707;":AUTOSCALE"                            !Autoscales the unit
80   WAIT 3
90   REAL Sens,Tdelay                                   !Set up Sens and Tdelay
100                                                     !as variables
110  INTEGER J                                          !Set up J as variable
120  OUTPUT 707;":BNC PROBE"                            !BNC output to probe mode
130  OUTPUT 707;":ACQUIRE:TYPE NORMAL"                  !Real time acquisition
140  OUTPUT 707;":TIMEBASE:RANGE 5E-4"                  !Set timebase to 50 us/div
150  OUTPUT 707;":TIMEBASE:DELAY 0                      !Set delay to zero
160  OUTPUT 707;":TIMEBASE:REFERENCE CENTER"            !Delay reference at
170                                                     !center of graticule
180  !
190  OUTPUT 707;":BEEPER"
200  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - DELAY FROM TRIGGER EVENT WILL CHANGE'"
210  ENTER 2
220  OUTPUT 707;":SYSTEM:DSP ''"
230  Tdelay=0                                           !Set Tdelay to zero
240  !
250  !The following lines vary the timebase delay
260  !
270  FOR J=1 TO 18
280    OUTPUT 707;":TIMEBASE:DELAY ";Tdelay
290    Tdelay=Tdelay+4E-5
300    WAIT .3
310  NEXT J
320  !
330  !
340  OUTPUT 707;":BEEPER"
350  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - HORIZONTAL TIME WILL CHANGE'"
360  ENTER 2
```

```
370  OUTPUT 707;":SYSTEM:DSP ''"
380  !
390  Sens=2E-2                                    !Set Sens variable to 20 ms
400  OUTPUT 707;"TIMEBASE:DELAY 0"                !Set delay to zero
410  !
420  !The following lines vary the horizontal timebase range
430  !
440  FOR J=1 TO 15
450    OUTPUT 707;":TIMEBASE:RANGE ";Sens
460    Sens=Sens/1.7
470    WAIT .3
480  NEXT J
490  !
500  !
510  OUTPUT 707;":SYSTEM:DSP 'END OF PROGRAM'"
520  WAIT 15
530  OUTPUT 707;":SYSTEM:DSP ''"
540  LOCAL 707                                    !Returns instrument to local
550  END
```

Measurement Setup Program

This sample program demonstrates some of the commands in the MEASURE subsystem. The program assumes that the AC CAL signal from the rear panel of the instrument is connected to channel 1 through a 10:1 probe.

```
10   CLEAR 707                                          !Device clear command.
20                                                      !initializes HP-IB registers.
30   !
40   OUTPUT 707;":EOI ON"                               !Turn on EOI
50   OUTPUT 707;"*RST"                                  !Reset instrument to
60                                                      !known state
70   DIM Measure$[400]
80   OUTPUT 707;":SYSTEM:HEADER ON"                     !Turn headers on for queries
90   OUTPUT 707;":AUTOSCALE"                            !Autoscales the unit
100  WAIT 3
110  OUTPUT 707;":BNC PROBE"                            !BNC output to probe mode
120  OUTPUT 707;":ACQUIRE:TYPE NORMAL"                  !Set display mode to normal
130  OUTPUT 707;":TIMEBASE:RANGE 2E-3"                  !200 uS per division
140  !
150  OUTPUT 707;":CHANNEL1:PROBE 10"                    !Attenuation to 10:1
160  OUTPUT 707;":CHANNEL1:RANGE 1.2"                   !Channel 1 to 1.2 V full scale
170  OUTPUT 707;":CHANNEL1:OFFSET -.4"                  !Channel centered at -.4 V
180  !
190  OUTPUT 707;":TRIGGER:MODE EDGE"                    !Edge triggering
200  OUTPUT 707;":TRIGGER:SLOPE POSITIVE"               !Trigger on positive edge
210  !
220  OUTPUT 707;":DISPLAY:FORMAT 1"                     !Full screen display
230  OUTPUT 707;":VIEW CHANNEL1"
240  OUTPUT 707;":BLANK CHANNEL2"
250  OUTPUT 707;":DISPLAY:VMARKER ON"                   !Turn on Vmarkers
260  OUTPUT 707;":DOSPLAY:TMARKER ON"                   !Turn on Tmarkers
270  !
280  OUTPUT 707;":MEASURE:SOURCE CHANNEL1"              !Channel 1 is the
290                                                     !measurement source
300  OUTPUT 707;":MEASURE:VSTART -.4"                   !Sets voltage markers to
310  OUTPUT 707;":MEASURE:VSTOP -.4"                    !-0.4 V.  This will be used as
320                                                     !a reference for the edge
330                                                     !find function.
340  OUTPUT 707;":BEEPER"
350  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - TIME MARKERS MOVE TO SIGNAL EDGES'"
360  ENTER 2                                            !This causes a pause in
```

```
370                                                     !the program.  Press ENTER on
380                                                     !the controller to continue.
390  OUTPUT 707;":SYSTEM:DSP ''"
400  !
410  INTEGER J                                          !Set up J as variable
420  !
430  !The following lines move the time markers between
440  !signal edges
450  !
460  FOR J=1 TO 3
470    OUTPUT 707;":MEASURE:ESTART ";J                  !Find Jth positive edge
480    WAIT .75
490    OUTPUT 707;":MEASURE:ESTOP ";-J                  !Find Jth negative edge
500    WAIT .75
510  NEXT J
520  !
530  OUTPUT 707;":BEEPER"
540  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - INCREMENT VOLTAGE AND TIME MARKERS'"
550  ENTER 2
560  OUTPUT 707;":SYSTEM:DSP ''"
570  !
580  !
590  REAL Tdelay,Voffset                                !Set up Tdelay and Voffset
600                                                     !as variables
610  Tdelay=0                                           !Initialize Tdelay variable
620  Voffset=0                                          !Initialize Toffset variable
630  !
640  !The following lines move the time and voltage start and
650  !stop markers
660  !
670  FOR J=1 TO 21
680    OUTPUT 707;":MEASURE:TSTART ";-1.E-3-Tdelay      !Move time start marker
690    OUTPUT 707;":MEASURE:TSTOP ";1.E-3+Tdelay        !Move time stop marker
700    OUTPUT 707;":MEASURE:VSTART ";-.4-Voffset        !Move voltage start marker
710    OUTPUT 707;":MEASURE:VSTOP ";-.4+Voffset         !Move voltage stop marker
720    Tdelay=Tdelay-1.E-4
730    Voffset=Voffset-4E-2
740  NEXT J
750  !
760  OUTPUT 707;":BEEPER"
770  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER TO MAKE AUTOMATIC MEASUREMENTS'"
780  ENTER 2
790  OUTPUT 707;":SYSTEM:DSP ''"
800  !
```

```
810  OUTPUT 707;":MEASURE:ALL?"                                    !Measure all parameters
820  !
830  !The results of the MEASURE ALL query will be displayed on the scope,
840  !and are available over HP-IB
850  !
860  ENTER 707 USING "-K";Measure$
870  PRINT USING "K";Measure$
880  !
890  OUTPUT 707;":BEEPER"
900  OUTPUT 707;":SYSTEM"DSP 'PRESS ENTER TO ACCURATELY MEASURE RISE TIME'"
910  ENTER 2
920  OUTPUT 707;":SYSTEM:DSP ''"
930  !
940  OUTPUT 707;":TIMEBASE:RANGE 5E-4                              !Expand the display
950  WAIT 2                                                        !Must wait to acquire new data
960                                                                !after changing the timebase
970                                                                !range
980  OUTPUT 707;":MEASURE:RISETIME?"                               !Measure the rise time
990  !
1000 ENTER 707;Measure$
1010 PRINT Measure$
1020 !
1030 OUTPUT 707;":SYSTEM:DSP 'END OF PROGRAM'"
1040 WAIT 15
1050 OUTPUT 707;":SYSTEM:DSP ''"
1060 LOCAL 707                                                     !Returns instrument to local
1070 END
```

Digitize Program

This sample program demonstrates some of the commands used to digitize a waveform. The program assumes that the AC CAL signal from the rear panel of the instrument is connected to channel 1 through a 10:1 probe.

```
10   CLEAR 707                                          !Device clear command.
20   !initializes HP-IB registers.
30   !
40   OUTPUT 707;":EOI ON"                               !Turn on EOI
50   OUTPUT 707;"*RST"                                  !Reset instrument to
60                                                      !known state
70   OUTPUT 707;":AUTOSCALE"                            !Autoscales the unit
80   WAIT 3
90   OUTPUT 707;":BNC PROBE"                            !BNC output to probe mode
100  !
110  !
120  OUTPUT 707;":ACQUIRE:TYPE NORMAL"                  !Set display mode to normal
130  OUTPUT 707;":CHANNEL1:PROBE 10"                    !Set probe attenuation to 10:1
140  OUTPUT 707;":CHANNEL1:RANGE 1.6"                   Set vertical range to 1.6 V
150  OUTPUT 707;":CHANNEL1:OFFSET -.4"                  !Set offset to -0.4 V
160  OUTPUT 707;":CHANNEL1:COUPLING DC"                 !Set coupling to DC
170  OUTPUT 707;":TIMEBASE:RANGE 1E-7"                  !10 ns per division
180  !
190  !
200  OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"             !Selects channel 1 as the source
210  OUTPUT 707;":ACQUIRE:COMPLETE 30"                  !30% completion criteria for
220                                                     !each acquisition
230  OUTPUT 707;":ACQUIRE:POINTS 32"                    !32 points for
240                                                     !each acquisition record
250  OUTPUT 707;":DIGITIZE CHANNEL1"
260  REAL Pts,Cmp                                       !Set up Pts and Cmp
270                                                     !as variables
280  INTEGER J                                          !Set up J as a variable
290  OUTPUT 707;":BEEPER"
300  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - CHANGES NUMBER OF POINTS DISPLAYED'"
310  ENTER 2
320  OUTPUT 707;":SYSTEM:DSP ''"
330  Pts=32                                             !Set Pts variable to 32
340  !
350  !The following lines increase the number of points for
360  !each acquisition record
```

```
370  !
380  FOR J=1 TO 5
390    OUTPUT 707;":ACQUIRE:POINTS ";Pts                    !Sets new points
400    OUTPUT 707;":DIGITIZE CHANNEL1"
410    Pts=Pts*2
420    WAIT 2
430  NEXT J
440  !
450  OUTPUT 707;":BEEPER"
460  OUTPUT 707;":SYSTEM:DSP 'PRESS ENTER - CHANGE COMPLETION CRITERIA'"
470  ENTER 2
480  OUTPUT 707;":SYSTEM:DSP ''"
490  OUTPUT 707;":ACQUIRE:POINTS 128"                     !128 points for
500                                                       !each acquisition record
510  Cmp=100                                              !Sets Cmp variable to 100
520  !
530  !The following lines reduce the completion criteria for
540  !each acquisition
550  !
560  FOR J=1 TO 7
570    OUTPUT 707;":ACQUIRE:COMPLETE ";Cmp                !Sets new completion criteria
580    OUTPUT 707;":DIGITIZE CHANNEL1"
590    Cmp=CMP-15
600    WAIT 2
610  NEXT J
620  !
630  !
640  OUTPUT 707;":SYSTEM:DSP 'END OF PROGRAM'"
650  WAIT 15
660  OUTPUT 707;":SYSTEM:DSP ''"
670  LOCAL 707                                            !Returns instrument to local
680  END
```

Hardcopy Program (Service Request using OPC)

This sample program demonstrates some of the commands in the HARDCOPY subsystem. The service request is used to detect when the printing is complete. The program assumes that a graphics printer is used and its address is set to 1.

```
10   CLEAR 707                                          !Device clear command.
20                                                      !initializes HP-IB registers.
30   OUTPUT 707;":EOI ON"                               !Turn on EOI
40   ON INTR 7,5 GOTO 220                               !Exit printing routine
50                                                      !after SRQ
60   ENABLE INTR 7;2                                    !Enables SRQ on bus #7
70   OUTPUT 707;"*CLS"                                  !Clear status data structures
80   OUTPUT 707;"*ESE 1"                                !Enable OPC
90   OUTPUT 707;"*SRE 32"                               !Enable Event Status Register
100                                                     !Interrupt
110  OUTPUT 707;":HARDCOPY:PAGE AUTOMATIC"
120  OUTPUT 707;":HARDCOPY:LENGTH 12"
130  OUTPUT 707;":PRINT?;*OPC"                          !Set OPC when the
140                                                     !print is complete
150  SEND 7;UNT UNL                                     !Clears bus
160  SEND 7;TALK 7                                      !Puts the scope in talk mode
170  SEND 7;LISTEN 1                                    !Tells printer to listen
180  SEND 7;DATA                                        !Lowers ATN line @ controller
190  GOTO 190                                           !Loops until printing is
200                                                     !complete and interrupt
210                                                     !is generated
220  A=SPOLL(707)                                       !Clear service request
230  OUTPUT 707;":SYSTEM:DSP 'PRINT IS COMPLETE'"
240  WAIT 15
250  OUTPUT 707;":SYSTEM:DSP ''"
260  END
```

Waveform Template Program

This sample program demonstrates how to use some of the commands in the HP 54501A to make a waveform template for comparing waveforms. This program assumes that the AC CAL signal from the rear panel of the instrument is connected to channel 1 through a 10:1 probe.

```
10   CLEAR 707                                    !Device clear command.
20                                                !initializes HP-IB registers.
30   OUTPUT 707;":EOI ON"                         !Turn on EOI
40   OPTION BASE 0                                !Select default option base
50   OUTPUT 707;"*RST"                            !Reset instrument to
60                                                !known state
70   OUTPUT 707;":AUTOSCALE"                      !Autoscales the unit
80   WAIT 3
90   DIM Preamble$[200]
100  DIM Preamb(1:10)
110  !
120  OUTPUT 707;":BNC PROBE"                      !BNC output to probe mode
130  OUTPUT 707;":CHANNEL1:PROBE 10"              !Set probe attenuation to 10:1
140  OUTPUT 707;":CHANNEL1:RANGE 1.6"             !Set vertical range to 1.6 V
150  OUTPUT 707;":CHANNEL1:OFFSET -.4"            !Set offset to -0.4 V
160  OUTPUT 707;":CHANNEL1:COUPLING DC"           !Set coupling to DC
170  !
180  !
190  !
200  OUTPUT 707;":TIMEBASE:DELAY 0"               !Set delay to zero
210  OUTPUT 707;":TIMEBASE:MODE TRIGGERED"        !Triggered timebase mode
220  OUTPUT 707;":TIMEBASE:RANGE 5E-4"            !50 uS per division
230  OUTPUT 707;":TIMEBASE:REFERENCE CENTER"      !Display reference
240                                               !to center screen
250  !
260  OUTPUT 707;":TRIGGER:LEVEL -.4"              !Set trigger level to -.4
270  OUTPUT 707;":TRIGGER:MODE EDGE"              !Edge triggering
280  !
290  OUTPUT 707:":ACQUIRE:COMPLETE 100            !100% completion criteria
300  OUTPUT 707:":ACQUIRE:TYPE AVERAGE            !Set display mode to average
310  OUTPUT 707:":ACQUIRE:COUNT 4"                !4 values averaged
320  OUTPUT 707:":ACQUIRE:POINTS 512"             !512 points per
330                                               !acquisition record
340  !
350  OUTPUT 707;":DISPLAY:CONNECT ON"             !Turn on connect-the-dots
360                                               !function
```

```
370  OUTPUT 707;":DISPLAY:GRATICULE FRAME"
380  !
390  OUTPUT 707;":SYSTEM:HEADER OFF"                          !Turn headers off
400  OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"                   !Selects channel 1 as
410  !the waveform source
420  OUTPUT 707;":WAVEFORM:FORMAT WORD"                       !WORD format for data
430                                                           !transfers
440  OUTPUT 707;":DIGITIZE CHANNEL1"
450  OUTPUT 707;":WAVEFORM:PREAMBLE?"                         !Output waveform preamble
460                                                           !to controller
470  ENTER 707;Preamb(*)
480  OUTPUT 707;":WAVEFORM:PREAMBLE?"                         !Output waveform preamble
490                                                           !to controller
500  ENTER 707 USING "-K";Preamble$
510  OUTPUT 707;":WAVEFORM:DATA?"                             !Output waveform record
520                                                           !to controller
530  ENTER 707 USING "#,2A,8D";Header$,Bytes
540  Length=Bytes/2
550  !
560  ALLOCATE INTEGER Waveform(1:Length),Wavemax(1:Length),Wavemin(1:Length)
570  ENTER 707 USING "#,W";Waveform(*)
580  ENTER 707 USING "-K,B";End$
590  !
600  !The following lines set the voltage and time tolerance limits as
610  !a percent of full scale
620  !
630  Volt_tol=5                                               !5% of full scale voltage
640  Time_tol=5                                               !5% of full screen width
650  !
660  Time_tics=INT(512*Time_tol/100)
670  Volt_tics=INT(2*Preamb(10)*Volt_tol/100)
680  MAT Wavemin=Waveform                                     !Copy the waveform into the
690                                                           !min template memory
700  MAT Wavemax=Waveform                                     !Copy the waveform into the
710                                                           !max template memory
720  FOR Time_cntr=1 TO Length                                !This is the time bucket where
730                                                           !the center of the ellipse
740                                                           !is located
750    PRINT Time_cntr                                        !This counter tells you which
760                                                           !time bucket the program is
770                                                           !currently calculating
780    FOR Time_pt=(Time_cntr-Time_tics) TO (Time_cntr+Time_tics)   !This loop increments along the
790                                                           !time axis of the ellipse
800      IF Time_pt>0 AND Time_pt<=Length THEN
```

```
810         Volt_pt=Volt_tics*SQR(1-((Time_pt-Time_cntr)/Time_tics)^2
820         Volt_max=Waveform(Time_cntr)+Volt_pt
830         Volt_min=Waveform(Time_cntr)-Volt_pt
840         IF Wavemax(Time_pt)<Volt_max THEN
850            Wavemax(Time_pt)=Volt_max
860         END IF
870         IF Wavemin(Time_pt)>Volt_min THEN
880            Wavemin(Time_pt)=Volt_min
890         END IF
900      END IF
910    NEXT Time_pt
920  NEXT Time_cntr
930  OUTPUT 707;":WAVEFORM:SOURCE WMEMORY2"                    !Selects Wmemory 2
940                                                            !as source
950  OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Preamble$    !Send waveform
960                                                            !preamble to Wmemory 2
970  OUTPUT 707 USING "#,K";":WAVEFORM:DATA #800001024"        !Send waveform
980                                                            !record to Wmemory 2
990  OUTPUT 707 USING "W";Wavemin(*)
1000 OUTPUT 707;":VIEW WMEMORY2"
1010 OUTPUT 707;":WAVEFORM:SOURCE WMEMORY1"                    !Selects Wmemory 1
1020                                                           !as source
1030 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Preamble$    !Send waveform
1040                                                           !preamble to Wmemory 1
1050 OUTPUT 707 USING "#,K";":WAVEFORM:DATA #800001024"        !Send waveform
1060                                                           !record to Wmemory 1
1070 OUTPUT 707 USING "W";Wavemax(*)
1080 OUTPUT 707;":VIEW WMEMORY1"
1090 Funct_range=Range_value*2
1100 Range$=Val$(Funct_range)
1110 OUTPUT 707;":FUNCTION1:SUBSTRACT WMEMORY1,CHANNEL1"       !Set up function 1
1120 OUTPUT 707;":FUNCTION1:RANGE ";Range$
1130 OUTPUT 707;":FUNCTION2:SUBTRACT CHANNEL1,WMEMORY2"        !Set up function 2
1140 OUTPUT 707;":FUNCTION2:RANGE ";Range$
1150 OUTPUT 707;":VIEW FUNCTION1"
1160 OUTPUT 707;":VIEW FUNCTION2"
1170 OUTPUT 707;":RUN"
1180 OUTPUT 707;":MEASURE:STATISTICS ON"                       !Turn statistics on
1190 OUTPUT 707;":MEASURE:DESTINATION OFF"
1200 OUTPUT 707;":MEASURE:SOURCE FUNCTION1"
1210 WAIT 1
1220 OUTPUT 707;":MEASURE:VMIN"
1230 OUTPUT 707;":MEASURE:SOURCE FUNCTION2"
1240 WAIT 1
```

```
1250 OUTPUT 707;":MEASURE:VMIN"
1260 OUTPUT 707;":MEASURE:COMPARE VMIN,200,0"
1270 OUTPUT 707;":MEASURE:LIMITTEST MEASURE"
1280 OUTPUT 707;":MEASURE:POSTFAILURE STOP"
1290 END
```

Quick Reference Guide

The following section lists the commands and queries with their corresponding arguments and returned formats. The arguments for each command list the minimum argument required. The returned format listed assumes that HEADERS are off. The part of the command or query listed in uppercase letters refers to the short form of that command or query. The long form is the combination of uppercase and lowercase letters.

The following definitions are used:

<block data> ::= definite length block data
<integer> ::= integer - NR1 format
<real number> ::= exponential - NR3 format
<string> ::= string of alphanumeric characters
<measurement> ::= name of measurement
<upper> ::= high limit value
<lower> ::= low limit value

The suffix multipliers available for arguments are:

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

For more information on specific commands or queries, refer to the specific command or query in the programming reference manual.

Command	Argument	Returned Format
Common Commands		
*CLS	--	--
*ESE	{0 - 255}	--
*ESE?	--	{0 - 255}
*ESR?	--	{0 - 255}
*IDN?	--	<string>
*IST?	--	{1 or 0}
*LRN?	--	<block data>
*OPC	--	--
*OPC?	--	{1}
*OPT?	--	{0}
*PRE	{0 - 255}	--
*PRE?	--	{0 - 255}
*RCL	{0 - 4}	--
*RST	--	--
*SAV	{1 - 4}	--
*SRE	{0 - 255}	--
*SRE?	--	{0 - 255}
*STB?	--	{0 - 255}
*TRG	--	--
*TST?	--	<integer>
*WAI	--	--

Command	Argument	Returned Format
Root Level Commands		
:AUToscale	--	--
:BEEPer	--	--
	{ON or 1}	--
	{OFF or 0}	--
:BEEPer?	--	{1 or 0}
:BLANk	{CHANnel 1 - 4}	--
	{FUNCtion 1 or 2}	--
	{WMEMory 1 - 4}	--
	{PMEMory 1 or 2}	--
:DIGitize	{CHANnel 1 - 4}	--
:EOI	{ON or 1}	--
	{OFF or 0}	--
:EOI?	--	{1 or 0}
:ERASe	{PMEMory 0 - 2}	--
:LER?	--	{1 or 0}
:LTER?	--	{1 or 0}
:MENU	{TIMebase}	
	{CHANnel}	--
	{TRIGger}	--
	{DISPlay}	--
	{DELTa}	--
	{MATH}	--
	{SAVE}	--
	{MEASure}	--
	{UTILity}	--
	{SHOW}	--

Command	Argument	Returned Format
:MENU?	--	{CHANnel}
	--	{TRIGger}
	--	{DISPlay}
	--	{DELTa}
	--	{MATH}
	--	{SAVE}
	--	{MEASure}
	--	{UTILity}
	--	{SHOW}
:MERGe	{PMEMory1 or 2}	--
:PRINt?	--	--
:RUN	--	--
:SERial	<string>	--
:STOP	--	--
:STORe	{CHANnel 1 - 4},{WMEMory 1 - 4}	--
	{FUNCtion 1 or 2},{WMEMory 1 - 4}	--
	{WMEMory 1 - 4},{WMEMory 1 - 4}	--
:TER?	--	{1 or 0}
:VIEW	{CHANnel 1 - 4}	--
	{FUNCtion 1 or 2}	--
	{WMEMory 1 - 4}	--
	{PMEMory 1 or 2}	--

Command	Argument	Returned Format
System Subsystem Commands		
:SYSTem:DSP	<string>	--
:SYSTem:DSP?	--	<string>
:SYSTem:ERRor?	--	<integer>
	{NUMBer}	<integer>
	{STRing}	<integer>,<string>
:SYSTem:HEADer	{ON or 1}	--
	{OFF or 0}	--
:SYSTem:HEADer?	--	{1 or 0}
:SYSTem:KEY	{1 - 44}	--
:SYSTem:KEY?	--	{0 - 44}
:SYSTem:LONGform	{ON or 1}	--
	{OFF or 0}	--
:SYSTem:LONGform?	--	{1 or 0}
:SYSTem:SETup	<block data>	--
:SYSTem:SETup?	--	<block data>

Command	Argument	Returned Format
Acquire Subsystem Commands		
:ACQuire:COMPlete	{0 - 100}	--
:ACQuire:COMPlete?	--	{0 - 100}
:ACQuire:COUNt	{1 - 2048}	--
:ACQuire:COUNt?	--	{1 - 2048}
:ACQuire:POINts	{32 - 1024}	--
:ACQuire:POINts?	--	{32 - 1024}
:ACQuire:TYPE	{NORMal}	--
	{AVERage}	--
	{ENVelope}	--
ACQuire:TYPE?	--	{NORMal}
	--	{AVERage}
	--	{ENVelope}
Calibrate Subsystem Commands		
:CALibrate:TNUL1	<real number>,<real number>, <real number>	--
:CALibrate:TNUL1?	--	<real number>,<real number>, <real number>

Command	Argument	Returned Format
Channel Subsystem Commands		
:CHANnel{1 - 4}:COUPling	{AC}	--
	{DC}	--
	{DCFifty}	--
:CHANnel{1 - 4}:COUPling?	--	{AC}
	--	{DC}
	--	{DCFifty}
:CHANnel{1 - 4}:ECL	--	--
:CHANnel{1 - 4}:HFReject	{ON or 1}	--
	{OFF or 0}	--
:CHANnel{1 - 4}:HFReject?	--	{1 or 0}
:CHANnel{1 - 4}:OFFSet	<real number>	--
:CHANnel{1 - 4}:OFFSet?	--	<real number>
:CHANnel{1 - 4}:PROBe	{0.9 - 1000}	--
:CHANnel{1 - 4}:PROBe?	--	{0.9 - 1000}
:CHANnel{1 - 4}:RANGe	<real number>	--
:CHANnel{1 - 4}:RANGe?	--	<real number>
:CHANnel{1 - 4}:TTL	--	--

Command	Argument	Returned Format
Display Subsystem Commands		
:DISPlay:COLumn	{0 - 72}	--
:DISPlay:COLumn?	--	{0 - 72}
:DISPlay:CONNect	{ON or 1}	--
	{OFF or 0}	--
:DISPlay:CONNect?	--	{1 or 0}
:DISPlay:DATA	<block data>	--
:DISPlay:DATA?	--	<block data>
:DISPlay:FORMat	{1, 2, or 4}	--
:DISPlay:FORMat?	--	{1, 2, or 4}
:DISPlay:GRATicule	{OFF}	--
	{GRID}	--
	{AXES}	--
	{FRAMe}	--
:DISPlay:GRATicule?	--	{OFF}
	--	{GRID}
	--	{AXES}
	--	{FRAMe}
:DISPlay:INVerse	{ON or 1}	--
	{OFF or 0}	--
:DISPlay:INVerse?	--	{1 or 0}
:DISPlay:LINE	<string>	--
:DISPlay:MASK	{0 - 255}	--
:DISPlay:MASK?	--	{0 - 255}
:DISPlay:PERSistence	{INFinite}	--
	{SINGle}	--
	{0.1 - 11}	--
:DISPlay:PERSistence?	--	{INFinite}
	--	{SINGle}
	--	{0.1 - 11}
:DISPlay:ROW	{0 - 24}	--
:DISPlay:ROW?	--	{0 - 24}
:DISPlay:SCReen	{ON or 1}	--
	{OFF or 0}	--

Command	Argument	Returned Format
:DISPlay:SCReen?	--	{1 or 0}
:DISPlay:SOURce	{PMEMory 0 - 3}	--
:DISPlay:SOURce?	--	{PMEMory 0 - 3}
:DISPlay:STRing	<string>	--
:DISPlay:TEXT	{BLANk}	
:DISPlay:TMARker	{ON or 1}	--
	{OFF or 0}	--
:DISPlay:TMARker?	--	{1 or 0}
:DISPlay:VMARker	{ON or 1}	--
	{OFF or 0}	--
:DISPlay:VMARker?	--	{1 or 0}
Function Subsystem Commands		
:FUNCtion{1 or 2}:ADD	{CHANnel 1 - 4},{CHANnel 1 - 4}	--
	{CHANnel 1 - 4},{WMEMory 1 - 4}	--
	{WMEMory 1 - 4},{CHANnel 1 - 4}	--
	{WMEMory 1 - 4},{WMEMory 1 - 4}	--
:FUNCtion{1 or 2}:INVert	{CHANnel 1 - 4}	--
	{WMEMory 1 - 4}	--
:FUNCtion{1 or 2}:MULTiply	{CHANnel 1 - 4},{CHANnel 1 - 4}	--
	{CHANnel 1 - 4},{WMEMory 1 - 4}	--
	{WMEMory 1 - 4},{CHANnel 1 - 4}	--
	{WMEMory 1 - 4},{WMEMory 1 - 4}	--
:FUNCtion{1 or 2}:OFFSet	<real number>	--
:FUNCtion{1 or 2}:OFFSet?	--	<real number>
:FUNCtion{1 or 2}:ONLY	{CHANnel 1 - 4}	--
	{WMEMory 1 - 4}	--
:FUNCtion{1 or 2}:RANGe	<real number>	--
:FUNCtion{1 or 2}:RANGe?	--	<real number>
:FUNCtion{1 or 2}:SUBTract	{CHANnel 1 - 4},{CHANnel 1 - 4}	--
	{CHANnel 1 - 4},{WMEMory 1 - 4}	--
	{WMEMory 1 - 4},{CHANnel 1 - 4}	--
	{WMEMory 1 - 4},{WMEMory 1 - 4}	--

Command	Argument	Returned Format
:FUNCtion{1 or 2}:VERSus	{CHANnel 1 - 4},{CHANnel 1 - 4} {CHANnel 1 - 4},{WMEMory 1 - 4} {WMEMory 1 - 4},{CHANnel 1 - 4} {WMEMory 1 - 4},{WMEMory 1 - 4}	-- -- -- --
Hardcopy Subsystem Commands		
:HARDcopy:LENGth :HARDcopy:LENGth? :HARDcopy:PAGE :HARDcopy:PAGE?	{11 or 12} -- {MANual} {AUTomatic} -- --	-- {11 or 12} -- -- {MANual} {AUTomatic}
Measure Subsystem Commands		
:MEASure:ALL? :MEASure:COMPare :MEASure:COMPare? :MEASure:CURSor? :MEASure:DEFine	-- <measurement>,<upper>,<lower> <measurement> {DELTa} {STARt} {STOP} {DELay},<real number>,... {PWIDth},{MIDDle} {PWIDth},{UPPer} {PWIDth},{LOWer} {NWIDth},{MIDDle} {NWIDth},{UPPer} {NWIDth},{LOWer}	<real number>,... -- <real number>,<real number> <real number>,<real number> <real number>,<real number> <real number>,<real number> -- -- -- -- -- -- --

Command	Argument	Returned Format
:MEASure:DEFine?	{DELay}	<real number>,...
	{PWIDth}	{MIDDle}
	{PWIDth}	{UPPer}
	{PWIDth}	{LOWer}
	{NWIDth}	{MIDDle}
	{NWIDth}	{UPPer}
	{NWIDth}	{LOWer}
:MEASure:DELay	--	--
:MEASure:DELay?	--	<real number>
:MEASure:DESTination	{WMEMory 1 - 4}	--
	{PMEMory 1 or 2}	--
	{OFF}	--
:MEASure:DESTination?	--	{WMEMory 1 - 4}
	--	{PMEMory 1 or 2}
	--	{OFF}
:MEASure:DUTycycle	--	--
:MEASure:DUTycycle?	--	<real number>
:MEASure:ESTArt	<+ or -><integer>	--
:MEASure:ESTArt?	--	<integer>
:MEASure:ESTOp	<+ or -><integer>	--
:MEASure:ESTOp?	--	<integer>
:MEASure:FALLtime	--	--
:MEASure:FALLtime?	--	<real number>
:MEASure:FREQuency	--	--
:MEASure:FREQuency?	--	<real number>
:MEASure:LIMittest	{MEASure}	--
	{OFF}	--
:MEASure:LOWer	<real number>	--
:MEASure:LOWer?	--	<real number>
:MEASure:MODE	{STANdard}	--
	{USER}	--
:MEASure:MODE?	--	{STANdard}
	--	{USER}

Command	Argument	Returned Format
:MEASure:NWIDth	--	--
:MEASure:NWIDth?	--	<real number>
:MEASure:OVERshoot	--	--
:MEASure:OVERshoot?	--	<real number>
MEASure:PERiod	--	--
:MEASure:PERiod?	--	<real number>
:MEASure:POSTfailure	{CONTinue}	--
	{STOP}	--
:MEASure:POSTfailure?	--	{CONTinue}
	--	{STOP}
:MEASure:PRECision	{COARse}	--
:MEASure:PRECision?	--	{COARse}
:MEASure:PREShoot	--	--
:MEASure:PREShoot?	--	<real number>
:MEASure:PWIDth	--	--
:MEASure:PWIDth?	--	<real number>
:MEASure:RESults?	--	{1 - 8}<string>[;<string>]...
	--	{0}
:MEASure:RISetime	--	--
:MEASure:RISetime?	--	<real number>
:MEASure:SCRatch	--	--
:MEASure:SOURce	{CHANnel 1 - 4}	--
	{FUNCtion 1 or 2}	--
	{WMEMory 1 - 4}	--
:MEASure:SOURce?	--	{CHANnel 1 - 4}
	--	{FUNCtion 1 or 2}
	--	{WMEMory 1 - 4}
:MEASure:STATistics	{ON or 1}	--
	{OFF or 0}	--
:MEASure:STATistics?	--	{1 or 0}
:MEASure:TDELta?	--	<real number>
:MEASure:TMAX?	--	<real number>
:MEASure:TMIN?	--	<real number>

Command	Argument	Returned Format
:MEASure:TSTArt	<real number>	--
:MEASure:TSTArt?	--	<real number>
:MEASure:TSTOp	<real number>	--
:MEASure:TSTOp?	--	<real number>
:MEASure:TVOLt?	<real number>,{+ or -}<integer>	<real number>
:MEASure:UNITs	{PERCent}	--
	{VOLTs}	--
:MEASure:UNITs?	--	{PERCent}
	--	{VOLTs}
:MEASure:UPPer	<real number>	--
:MEASure:UPPer?	--	<real number>
:MEASure:VAMPlitude	--	--
:MEASure:VAMPlitude?	--	<real number>
:MEASure:VAVerage	--	--
:MEASure:VAVerage?	--	<real number>
:MEASure:VBASe	--	--
:MEASure:VBASe?	--	<real number>
:MEASure:VDELta?	--	<real number>
:MEASure:VFIFty	--	--
:MEASure:VMAX	--	--
:MEASure:VMAX?	--	<real number>
:MEASure:VMIN	--	--
:MEASure:VMIN?	--	<real number>
:MEASure:VPP	--	--
:MEASure:VPP?	--	<real number>
:MEASure:VRELative	{0 - 100}	--
:MEASure:VRELative?	--	{50 - 100}
:MEASure:VRMS	--	--
:MEASure:VRMS?	--	<real number>
:MEASure:VSTArt	<real number>	--
:MEASure:VSTArt?	--	<real number>
:MEASure:VSTOp	<real number>	--

Command	Argument	Returned Format
:MEASure:VSTOp?	--	<real number>
:MEASure:VTIMe?	<real number>	<real number>
:MEASure:VTOP	--	--
:MEASure:VTOP?	--	<real number>
Timebase Subsystem Commands		
:TIMebase:DELay	<real number>	--
:TIMebase:DELay?	--	<real number>
:TIMebase:MODE	{AUTO}	--
	{TRIGgered}	--
	{SINGle}	--
:TIMebase:MODE?	--	{AUTO}
	--	{TRIGgered}
	--	{SINGle}
:TIMebase:RANGe	<real number>	--
:TIMebase:RANGe?	--	<real number>
:TIMebase:REFerence	{LEFT}	--
	{CENTer}	--
	{RIGHt}	--
:TIMebase:REFerence?	--	{LEFT}
	--	{CENTer}
	--	{RIGHt}
:TIMebase:WINDow	{ON or 1}	--
	{OFF or 0}	--
:TIMebase:WINDow?	--	{1 or 0}
:TIMebase:WINDow:DELay	<real number>	--
:TIMebase:WINDow:DELay?	--	<real number>
:TIMebase:WINDow:RANGe	<real number>	--
:TIMebase:WINDow:RANGe?	--	<real number>

Command	Argument	Returned Format
Trigger Subsystem Commands		
:TRIGger:CONDition	{ENTer}	--
	{EXIT}	--
	{GT},<real number>	--
	{LT},<real number>	--
	{RANGe},<real number>,<real number>	--
	{TRUE}	--
	{FALSe}	--
:TRIGger:CONDition?	--	{ENTer}
	--	{EXIT}
	--	{GT},<real number>
	--	{LT},<real number>
	--	{RANGe},<real number>,<real number>
	--	{TRUE}
	--	{FALSe}
:TRIGger:DELay	{TIME},<real number>	--
	{EVENt},<real number>	--
:TRIGger:DELay?	--	{TIME},<real number>
	--	{EVENt},<real number>
:TRIGger:DELay:SLOPe	{POSitive}	--
	{NEGative}	--
:TRIGger:DELay:SLOPe?	--	{POSitive}
	--	{NEGative}
:TRIGger:DELay:SOURce	{CHANnel 1 - 4}	--
:TRIGger:DELay:SOURce?	--	{CHANnel 1 - 4}
:TRIGger:FIELd	{1 or 2}	--
:TRIGger:FIELd?	--	{1 or 2}
:TRIGger:HOLDoff	{TIME},<real number>	--
:TRIGger:HOLDoff?	--	<real number>

Command	Argument	Returned Format
:TRIGger:LEVel	<real number>	--
:TRIGger:LEVel?	--	<real number>
:TRIGger:LINE	{1 - 625}	--
:TRIGger:LINE?	--	{1 - 625}
:TRIGger:LOGic	{HIGH}	--
	{LOW}	--
	{DONTcare}	--
:TRIGger:LOGic?	--	{HIGH}
	--	{LOW}
	--	{DONTcare}
:TRIGger:MODE	{EDGE}	--
	{PATTern}	--
	{STATe}	--
	{DELay}	--
	{TV}	--
:TRIGger:MODE?	--	{EDGE}
	--	{PATTern}
	--	{STATe}
	--	{DELay}
	--	{TV}
:TRIGger:OCCurrence	{1 - 16000000}	--
:TRIGger:OCCurrence?	--	{1 - 16000000}
:TRIGger:OCCurrence:SLOPe	{POSitive}	--
	{NEGative}	--
:TRIGger:OCCurrence:SLOPe?	--	{POSitive}
	--	{NEGative}
:TRIGger:OCCurrence:SOURce	{CHANnel 1 - 4}	--
:TRIGger:OCCurrence:SOURce?	--	{CHANnel 1 - 4}
:TRIGger:PATH	{CHANnel 1 - 4}	--
:TRIGger:PATH?	--	{CHANnel 1 - 4}
:TRIGger:POLarity	{POSitive}	--
	{NEGative}	--
:TRIGger:POLarity?	--	{POSitive}
	--	{NEGative}

Command	Argument	Returned Format
:TRIGger:QUALity	{EDGE}	--
	{PATTern}	--
	{STATe}	--
	{LOW}	--
	{HIGH}	--
:TRIGger:QUALity?	--	{EDGE}
	--	{PATTern}
	--	{STATe}
	--	{LOW}
	--	{HIGH}
:TRIGger:SLOPe	{POSitive}	--
	{NEGative}	--
:TRIGger:SLOPe?	--	{POSitive}
	--	{NEGative}
:TRIGger:SOURce	{CHANnel 1 - 4}	--
:TRIGger:SOURce?	--	{CHANnel 1 - 4}
:TRIGger:STANdard	{525}	--
	{625}	--
	{USER}	--
:TRIGger:STANdard?	--	{525}
	--	{625}
	--	{USER}

Command	Argument	Returned Format
Waveform Subsystem Commands		
:WAVeform:COUNt?	--	{1}
:WAVeform:DATA	<block data>	--
:WAVeform:DATA?	--	<block data>
:WAVeform:FORMat	{ASCii}	--
	{WORD}	--
	{BYTE}	--
	{COMPressed}	--
:WAVeform:FORMat?	--	{ASCii}
	--	{WORD}
	--	{BYTE}
	--	{COMPressed}
:WAVeform:POINts?	--	<integer>
:WAVeform:PREamble	<real number>,...	--
:WAVeform:PREamble?	--	<real number>,...
:WAVeform:SOURce	{CHANnel 1 - 4}	--
	{WMEMory 1 - 4}	--
:WAVeform:SOURce?	--	{CHANnel 1 - 4}
	--	{WMEMory 1 - 4}
:WAVeform:TYPE?	--	{AVERage}
	--	{ENVelope}
	--	{NORMal}
:WAVeform:XINCrement?	--	<real number>
:WAVeform:XORigin?	--	<real number>
:WAVeform:XREFerence?	--	{0}
:WAVeform:YINCrement?	--	<real number>
:WAVeform:YORigin?	--	<real number>
:WAVeform:YREFerence?	--	<integer>

Index

A

AC RMS, 14-60
Acquire Subsystem, 8-1
  COMPlete Command/Query, 8-4
  COUNt Command/Query, 8-5
  POINts Command/Query, 8-6
  Syntax Diagram, 8-3
  TYPE Command/Query, 8-7
ADD Command, 12-4
Addressing, 2-1
Addressing the instrument, 1-3
advisory line, 7-3
Algorithms, A-1
ALL Query, 14-11
alpha argument, 4-1
Angular brackets, 1-3
arbitrary ASCII response data, 3-23
arbitrary block program data, 3-14
ASCII format, 17-5
AUTO timebase mode, 15-4
automatic measurements, A-1
Automatic Top-Base, A-2
AUToscale Command, 6-4
AVERAGE Type, 17-2
Averaging Mode, 8-2

B

BEEPer Command/Query, 6-5
BLANk Command, 6-6, 6-20
block data, 1-18, 3-23
Buffer Deadlock, 3-4
BYTE format, 17-5

C

Calibrate Subsystem, 9-1
  Syntax Diagram, 9-1
  TNULl Command/Query, 9-2
carrage return, 4-5
Channel Subsystem, 10-1
  COUPling Command/Query, 10-3
  ECL Command, 10-4
  HFReject Command/Query, 10-5
  OFFSet Command/Query, 10-6
  PROBe Command/Query, 10-7
  RANGe Command/Query, 10-8
  Syntax Diagram, 10-1
  TTL Command, 10-9
Character data, 1-9, 1-15
Character program data, 1-9, 1-15, 3-14
character response data, 3-23
CLEAR DISPLAY, 6-9
Clear measurement, 14-38
*CLS (clear status) Command, 5-4
CME - command error, 3-30

COLUMN Command/Query, 11-4
comma, 3-6
Command, 1-4, 1-14
Acquire Subsystem, 8-1
AUToscale, 6-4
BEEPer, 6-5
BLANk, 6-6
CALibrate, 9-2
Calibrate Subsystem, 9-1
Channel Subsystem, 10-1
CHANnelN:COUPling, 10-3
CHANnelN:ECL, 10-4
CHANnelN:HFReject, 10-5
CHANnelN:OFFSet, 10-6
CHANnelN:PROBe, 10-7
CHANnelN:RANGe, 10-8
CHANnelN:TTL, 10-9
*CLS, 5-1, 5-4
common, 4-4
Common Commands, 5-1
COMPlete, 8-4
COUNt, 8-5
Cross-Reference, 4-13
DIGitize, 6-7
Display Subsystem, 11-1
DISPlay:COLumn, 11-4
DISPlay:CONNect, 11-5
DISPlay:DATA, 11-6
DISPlay:FORMat, 11-8
DISPlay:GRATicule, 11-9
DISPlay:INVerse, 11-10
DISPlay:LINE, 11-11
DISPlay:MASK, 11-12
DISPlay:PERSistence, 11-14
DISPlay:ROW, 11-15
DISPlay:SCReen, 11-16
DISPlay:SOURce, 11-17
DISPlay:STRing, 11-18
DISPlay:TEXT, 11-19
DISPlay:TMARker, 11-20
DISPlay:VMARker, 11-21
EOI, 6-8
Command (continued)
ERASe, 6-9
*ESE, 5-5
*ESR, 5-7
Function Subsystem, 12-1
FUNCtionN:ADD, 12-4
FUNCtionN:INVert, 12-5
FUNCtionN:MULTiply, 12-6
FUNCtionN:OFFSet, 12-7
FUNCtionN:ONLY, 12-8
FUNCtionN:RANGe, 12-9
FUNCtionN:SUBTract, 12-10
FUNCtionN:VERSus, 12-11
Hardcopy Subsystem, 13-1
HARDcopy:LENGth, 13-2
HARDcopy:PAGE, 13-3
HEADer, 1-14
*IDN, 5-9
*IST, 5-10
LER, 6-10
LONGform, 1-14
*LRN, 5-11
LTER, 6-11
Measure Subsystem, 14-1
MEASure:ALL, 14-11
MEASure:COMPare, 14-12
MEASure:CURSor, 14-14
MEASure:DEFine, 14-15
MEASure:DELay, 14-17
MEASure:DESTination, 14-18
MEASure:DUTycycle, 14-19
MEASure:ESTArt, 14-20
MEASure:ESTOp, 14-22
MEASure:FALLtime, 14-24
MEASure:FREQuency, 14-25
MEASure:LIMittest, 14-26
MEASure:LOWer, 14-27
MEASure:MODE, 14-28
MEASure:NWIDth, 14-29
MEASure:OVERshoot, 14-30
MEASure:PERiod, 14-31
MEASure:POSTfailure, 14-32

Command (continued)
MEASure:PRECision, 14-33
MEASure:PREShoot, 14-34
MEASure:PWIDth, 14-35
MEASure:RESults, 14-36
MEASure:RISetime, 14-37
MEASure:SCRatch, 14-38
MEASure:SOURce, 14-39
MEASure:STATistics, 14-40
MEASure:TDELta, 14-41
MEASure:TMAX, 14-42
MEASure:TMIN, 14-43
MEASure:TSTArt, 14-44
MEASure:TSTOp, 14-45
MEASure:TVOLt, 14-46
MEASure:UNITs, 14-48
MEASure:UPPer, 14-49
MEASure:VAMPlitude, 14-50
MEASure:VAVerage, 14-51
MEASure:VBASe, 14-52
MEASure:VDELta, 14-53
MEASure:VFIFty, 14-54
MEASure:VMAX, 14-55
MEASure:VMIN, 14-56
MEASure:VPP, 14-57
MEASure:VRELative, 14-58
MEASure:VRMS, 14-60
MEASure:VSTArt, 14-61
MEASure:VSTOp, 14-62
MEASure:VTIMe, 14-63
MEASure:VTOP, 14-64
MENU, 6-12
MERGe, 6-13
*OPC, 5-12
*OPT, 5-13
parsing, 3-1
POINts, 8-6
*PRE, 5-14
PRINt, 6-14
*RCL, 5-15
Root Level, 4-4
Root Level Commands, 6-1

Command (continued)
*RST, 5-16
RUN, 6-15
*SAV, 5-18
SERial, 6-16
*SRE, 5-19
*STB, 5-21
STOP, 6-17
STORe, 6-18
System Subsystem, 7-1
SYSTem:DSP, 7-3
SYSTem:ERRor, 7-4
SYSTem:HEADer, 7-6
SYSTem:KEY, 7-7
SYSTem:LONGform, 7-9
SYSTem:SETup, 7-10
TER, 6-19
Timebase Subsystem, 15-1
TIMebase:DELay, 15-3
TIMebase:MODE, 15-4
TIMebase:RANGe, 15-5
TIMebase:REFerence, 15-6
TIMebase:WINDow, 15-7
TIMebase:WINDow:DELay, 15-8
TIMebase:WINDow:RANGe, 15-9
*TRG, 5-23
Trigger Mode, 16-1
Trigger Subsystem, 16-1
TRIGGER:CONDITION, 16-4 / 16-5, 16-13
TRIGGER:DELAY, 16-6, 16-16
TRIGger:DELay:SLOPe, 16-17
TRIGGER:DELAY:SOURCE, 16-6, 16-18
TRIGGER:FIELD, 16-7, 16-19
TRIGGER:HOLDOFF, 16-4, 16-20
TRIGGER:LEVEL, 16-1, 16-3 / 16-4, 16-21
TRIGger:LINE, 16-22
TRIGGER:LOGIC, 16-5, 16-23
TRIGger:MODE, 16-24
TRIGGER:OCCURRENCE, 16-6, 16-8, 16-25
TRIGGER:OCCURRENCE:SLOPE, 16-6, 16-8, 16-26

Command (continued)
TRIGGER:OCCURRENCE:SOURCE, 16-6, 16-27
TRIGGER:PATH, 16-4 / 16-5, 16-28
TRIGGER:POLARITY, 16-7, 16-29
TRIGGER:QUALIFY, 16-6, 16-8, 16-30
TRIGGER:SLOPE, 16-3, 16-32
TRIGGER:SOURCE, 16-3, 16-7, 16-33
TRIGGER:STANDARD, 16-7, 16-34
TRIGGER:TIME, 16-6
*TST, 5-24
TYPE, 8-7
VIEW, 6-20
*WAI, 5-25
Waveform Subsystem, 17-1
WAVeform:COUNt, 17-9
WAVeform:DATA, 17-10
WAVeform:FORMat, 17-12
WAVeform:POINts, 17-13
WAVeform:PREamble, 17-14
WAVeform:SOURce, 17-16
WAVeform:TYPE, 17-17
WAVeform:XINCrement, 17-18
WAVeform:XORigin, 17-19
WAVeform:XREFerence, 17-20
WAVeform:YINCrement, 17-21
WAVeform:YORigin, 17-22
WAVeform:YREFerence, 17-23
Command and Data Concepts, 2-1
command error, 3-4
Command header, 1-5
Command Table
TRIGger:MODE, 16-2
command tree, 4-1, 4-4, 4-9
Command Types, 4-4
Common command header, 1-6
Common Commands, 3-27, 4-4, 5-1
*CLS , 5-4
*ESE , 5-5
*ESR Query, 5-7
*IDN Query, 5-9
*IST Query, 5-10
Common Commands (continued)
*LRN Query, 5-11
*OPC Command/Query, 5-12
*OPT Query, 5-13
*PRE Command/Query, 5-14
*RCL, 5-15
*RST, 5-16
*SAV Command, 5-18
*SRE Command/Query, 5-19
*STB, 5-21
Syntax Diagram, 5-1
*TRG Command, 5-23
*TST, 5-24
*WAI Command, 5-25
COMPare Command/Query, 14-12
COMPlete Command/Query, 8-4
Compound command header, 1-5
compound header, 4-5
compound query, 3-3
COMPRESSED format, 17-5
CONDition Command/Query, 16-13
CONNect Command/Query, 11-5
Controllers, 1-2
COUNt Command/Query, 8-5
COUNt Query, 17-9
COUPling Command/Query, 10-3
CURSor Query, 14-14

D

Data Acquisition Types, 17-2
DATA Command/Query, 11-6, 17-10
DDE - device specific error, 3-30
deadlock, 3-4
decimal numeric program data, 3-14
DEFINE Command/Query, 14-15
definite length arbitrary block response data, 3-23
Definite-length block response data, 1-18
DELay Command/Query, 14-17, 15-3, 16-16
DELay SLOPe Command/Query, 16-17

DELay SOURce Command/Query, 16-18
Delay Trigger Mode, 16-6, 16-13
DESTination Command/Query, 14-18
Device address
  HP-IB, 1-3
device clear command, 3-2
Device Listening Syntax, 3-8
Device Talking Syntax, 3-19
device-specific error, 3-4
DIGitize Command, 6-7, 8-1, 17-2
Display Subsystem, 11-1
  COLumn Command/Query, 11-4
  CONNect Command/Query, 11-5
  DATA Command/Query, 11-6
  FORMat Command/Query, 11-8
  GRATicule Command/Query, 11-9
  INVerse Command/Query, 11-10
  LINE Command, 11-11
  MASK Command/Query, 11-12
  PERSistence Command/Query, 11-14
  ROW Command/Query, 11-15
  SCReen Command/Query, 11-16
  SOURce Command/Query, 11-17
  STRing Command, 11-18
  Syntax Diagram, 11-1
  TEXT Command, 11-19
  TMARker Command/Query, 11-20
  VMARker Command/Query, 11-21
DSP Command/Query, 7-3
duty cycle, A-5
DUTycycle Command/Query, 14-19

E

ECL Command, 10-4
Edge Definition, A-2
Enter statement, 1-2
Entered, 16-4
Envelope Mode, 8-2
ENVELOPE Type, 17-2
EOI, 1-9
EOI Command/Query, 6-8
ERASe Command, 6-9
ERASE PMEMORY0, 6-9
error number, 7-4
ERRor Query, 7-4
error queue, 5-4
ESB - event status bit, 3-30
*ESE Command/Query, 5-5
*ESR Query, 5-7
ESR, event summary, 5-5
ESTArt Command/Query, 14-20
ESTOp Command/Query, 14-22
Event Status Register, 5-8
Example program, B-1
  Channel Setup, B-2
  Digitize, B-9
  Hardcopy, B-11
  Measurement Setup, B-6
  Timebase Subsystem, B-4
  Waveform Template, B-12
EXE - execution error, 3-30
execution error, 3-4
Exited, 16-4
exponential - NR3, 3-23

F

falltime, A-5
FALLtime Command/Query, 14-24
falltime measurement, 14-1
falltime measurements, A-1
Field Command/Query, 16-19
FORMat Command/Query, 11-8, 17-12
frequency, A-5
FREQuency Command/Query, 14-25
frequency measurement, 14-1
Function Subsystem, 12-1
  ADD Command, 12-4
  INVert Command, 12-5

MULTiply Command, 12-6
OFFSet Command/Query, 12-7
ONLY Command, 12-8
RANGe Command/Query, 12-9
SUBTract Command, 12-10
Syntax Diagram, 12-2
VERSus Command, 12-11
Functional Elements
Input Buffer, 3-1
Output Queue, 3-2
Parser, 3-2

G

GRATicule Command/Query, 11-9
group execute trigger (GET), 3-3
GT, 16-4

H

Hardcopy Subsystem, 13-1
LENGth Command/Query, 13-2
PAGE Command/Query, 13-3
Syntax Diagram, 13-1
HEADer command, 1-14
HEADer Command/Query, 7-6
headers, 4-1
HFReject Command/Query, 10-5
HOLDoff Command/Query, 16-20
HP-IB, 1-3

I

*IDN Query, 5-9
IEEE 488.1, 3-1
IEEE 488.2, 3-5, 4-4
IEEE 488.2 defined syntax, 3-5
IEEE 488.2 instruments, 3-1
IEEE 488.2 standard, 3-1
Individual Status, 3-34
Infinity Representation, 4-6
Initialization, 1-11
input
vertical, 6-4
Input Buffer, 3-1 / 3-2, 3-4
instrument
serial number, 5-9
Instrument address
HP-IB, 1-3
integer - NR1, 3-23
Interface Capabilities, 2-1
Interface select code
HP-IB, 1-3
Interrupted Condition, 3-4
interrupted error, 3-26
INVerse Command/Query, 11-10
INVert Command, 12-5
*IST Query, 5-10

K

key codes, 7-7
KEY Command/Query, 7-7

L

LCL - local, 3-30
leading colon, 4-5 / 4-6
learn string, 5-11
LENGth Command/Query, 13-2
LER Query, 6-10
level
trigger, 6-4
LEVel Command/Query, 16-21
Limit Test Event Register, 6-11

LIMittest Command, 14-26
LINE Command, 11-11
LINE Command/Query, 16-22
linefeed, 4-4
linefeed (CRLF), 4-5
LOGic Command/Query, 16-23
Longform, 1-8
LONGform command, 1-14
LONGform Command/Query, 7-9
LOWer Command/Query, 14-27
Lowercase, 1-8
*LRN Query, 5-11
LT, 16-4
LTER Query, 6-11

M

Making Measurements, A-1, 14-2
MASK Command/Query, 11-12
MAV - message available, 3-30
Measure Subsystem, 14-1
  ALL Query, 14-11
  COMPare Command/Query, 14-12
  CURSor Query, 14-14
  DEFine Command/Query, 14-15
  DELay Command/Query, 14-17
  DESTination Command/Query, 14-18
  DUTycycle Command/Query, 14-19
  ESTArt Command/Query, 14-20
  ESTOp Command/Query, 14-22
  FALLtime Command/Query, 14-24
  FREQuency Command/Query, 14-25
  LIMittest Command, 14-26
  LOWer Command/Query, 14-27
  MODE Command/Query, 14-28
  NWIDth Command/Query, 14-29
  OVERshoot Command/Query, 14-30
  PERiod Command/Query, 14-31
  POSTfailure Command/Query, 14-32
  PRECision Command/Query, 14-33
Measure Subsystem (continued)
  PREShoot Command/Query, 14-34
  PWIDth Command/Query, 14-35
  RESults Query, 14-36
  RISetime Command/Query, 14-37
  SCRatch Command, 14-38
  SOURce Command/Query, 14-39
  STATistics Command/Query, 14-40
  Syntax Diagram, 14-4
  TDELta Query, 14-41
  TMAX Query, 14-42
  TMIN Query, 14-43
  TSTArt Command/Query, 14-44
  TSTOp Command/Query, 14-45
  TVOLt Query, 14-46
  UNITs Command/Query, 14-48
  UPPer Command/Query, 14-49
  VAMPlitude Command/Query, 14-50
  VAVerage Command/Query, 14-51
  VBASe Command/Query, 14-52
  VDELta Query, 14-53
  VFIFty Command, 14-54
  VMAX Command/Query, 14-55
  VMIN Command/Query, 14-56
  VPP Command/Query, 14-57
  VRELative Command/Query, 14-58
  VRMS Command/Query, 14-60
  VSTArt Command/Query, 14-61
  VSTOp Command/Query, 14-62
  VTIMe Query, 14-63
  VTOP Command/Query, 14-64
Measurement Error, 14-2
Measurement Setup, A-1, 14-1
MENU Command/Query, 6-12
MERGe Command, 6-13
message exchange protocols, 3-1
Message terminator, 1-3
- width, A-5
mnemonic, 4-1
Mode
  Delay Trigger, 16-6
  Pattern Trigger, 16-4

State Trigger, 16-5
TV Trigger, 16-7
MODE Command/Query, 14-28, 15-4, 16-24
MSG - message, 3-30
MSS (Master Summary Status), 5-21
MSS - master summary status, 3-30
Multiple data parameters, 1-8
Multiple numeric variables, 1-19
Multiple program commands, 1-10
Multiple queries, 1-19, 3-3
Multiple subsystems, 1-10
MULTiply Command, 12-6

N

- width, A-5
new line character, 6-8
NL, 1-9, 3-6, 4-4
Normal Persistence mode, 8-1
NORMAL Type, 17-2
Notation Conventions and Definitions, 4-7
nr1 numeric response data, 3-23
nr3 numeric response data, 3-23
Numeric data, 1-9
Numeric program data, 1-9
Numeric variables, 1-17
NWIDth Command/Query, 14-29

O

OCCurrence Command/Query, 16-25
OCCurrence:SLOPe Command/Query, 16-26
OCCurrence:SOURce Command/Query, 16-27
offset
vertical, 6-4
OFFSet Command/Query, 10-6, 12-7
ONLY Command, 12-8
*OPC Command/Query, 5-12
OPC - operation complete, 3-30
Operation Complete, 3-31
*OPT Query, 5-13
OUTPUT command, 1-3
Output Queue, 3-2, 3-4, 5-12
Output statement, 1-2
Overlapped Commands, 4-7
OVERshoot Command/Query, 14-30

P

PAGE Command/Query, 13-3
Parallel Poll, 3-34
Parallel Poll Configure, 3-34, 3-36
Parallel Poll Register Enable, 3-34
Parallel Poll Unconfigure, 3-37
Parser, 3-2, 3-9
parsing, 3-1 / 3-2
PATH Command/Query, 16-28
Pattern Trigger Mode, 16-4, 16-13
period, A-5
PERiod Command/Query, 14-31
period measurement, 14-1
PERSistence Command/Query, 11-14
+ width, A-4
POINts Command/Query, 8-6
POINts Query, 17-13
POLarity Command/Query, 16-29
PON - power on, 3-30
POSTfailure Command/Query, 14-32
*PRE command, 3-34, 5-14
PREamble Command/Query, 17-14
PRECision Command/Query, 14-33
PREShoot Command/Query, 14-34
PRINt Query, 6-14
PROBe Command/Query, 10-7
proggram message unit, 3-9
Program command, 1-4
Program data, 1-8, 3-14
Program example, B-1

Channel Setup, B-2
Digitize, B-9
Hardcopy, B-11
Measurement Setup, B-6
Timebase Subsystem, B-4
Waveform Template, B-12
Program message, 1-4, 3-2 / 3-3, 3-6, 3-9, 4-6
Program message syntax, 1-4
Program message terminator, 1-9, 3-2, 4-5
Program message unit, 1-4, 3-6
program message unit separator, 3-10, 4-5
Program query, 1-4
Program syntax, 1-4
Protocols, 3-1
pulse width measurement, 14-1
PWIDth Command/Query, 14-35

Q

QUALify Command/Query, 16-30
Query, 1-4, 1-7, 1-14
Query command, 1-7
query error, 3-3 / 3-4
query message, 3-2 / 3-3
query message unit, 3-21
query program header, 3-10
Query response, 1-13
query responses, 4-7
Question mark, 1-7
QYE - query error, 3-30

R

RANGE, 16-4
RANGe Command/Query, 10-8, 12-9, 15-5
*RCL Command, 5-15
real number - NR3, 3-23
REFerence Command/Query, 15-6
Response data, 1-18, 3-23
response data separator, 3-25
Response Generation, 4-7
Response header, 3-21
response header separator, 3-26
response message, 3-2 / 3-3, 3-21
response message terminator, 3-26
response message unit, 3-21
response message unit separator, 3-26
response messages, 3-2
RESults Query, 14-36
risetime, A-5
RISetime Command/Query, 14-37
risetime measurement, 14-1
risetime measurements, A-1
Root Level Command
ERASe Command, 6-9
PRINt Query, 6-14
Root Level commands, 4-4, 6-1
AUToscale Command, 6-4
BEEPer Command/Query, 6-5
BLANk Command, 6-6
DIGitize Command, 6-7
EOI Command/Query, 6-8
LER Query, 6-10
LTER Query, 6-11
MENU Command/Query, 6-12
MERGe Command, 6-13
RUN Command, 6-15
SERial Command, 6-16
STOP Command, 6-17
STORe Command, 6-18
Syntax Diagram, 6-1
TER Query, 6-19
VIEW Command, 6-20
ROW Command/Query, 11-15
RQC - request control, 3-30
RQS - request service, 3-30
*RST Command, 5-16
RUN Command, 6-15

S

*SAV Command, 5-18
save register, 5-18
save/recall register, 5-15
SCRatch Command, 14-38
SCReen Command/Query, 11-16
semicolon, 3-6
sensitivity
  vertical, 6-4
Separator, 1-5
Sequential commands, 4-6
SERial Command, 6-16
serial number, 5-9, 6-16
Serial Poll, 3-32
Service, 5-19
SETup Command/Query, 7-10
707, 1-16
Shortform, 1-8
Simple command header, 1-5
SINGLE timebase mode, 15-4
SLOPe Command/Query, 16-32
SOURce Command/Query, 11-17, 14-39, 16-33, 17-16
space, 3-6
*SRE Command/Query, 5-19
standard + width, A-4
standard - width, A-5
STANdard Command/Query, 16-34
Standard Event Status Enable Register, 5-5
Standard Event Status Register, 5-5, 5-12
State Trigger Mode, 16-5, 16-13
STATistics Command/Query, 14-40
Status, 1-19
Status Byte, 3-31
Status Byte , 5-19
status data structures, 5-4
status register, 5-1
Status registers, 1-19
Status Reporting, 3-28
*STB Query, 5-21
STOP command, 6-15, 6-17
STORe Command, 6-18
STRing Command, 11-18
string program data, 3-14
string response data, 3-23
String variables, 1-16
Subsystem
  Acquire, 8-1
  Calibrate, 9-1
  Channel, 10-1
  Display, 11-1
  Function, 12-1
  Hardcopy, 13-1
  Measure, 14-1
  System, 7-1
  Timebase, 15-1
  Trigger, 16-1
  Waveform, 17-1
Subsystem commands, 4-4
SUBTract Command, 12-10
suffix multipliers, 3-16
suffix program data, 3-14
suffix units, 3-16
sweep speed, 6-4
Syntax Diagram
  Acquire Subsystem, 8-3
  Calibrate Subsystem, 9-1
  Channel Subsystem, 10-1
  Common Commands, 5-1
  Display Subsystem, 11-1
  Function Subsystem, 12-2
  Hardcopy Subsystem, 13-1
  Measure Subsystem, 14-4
  Root Level Commands, 6-1
  System Subsystem, 7-2
  Trigger Subsystem, 16-9
  Waveform Subsystem, 17-7
syntax diagrams, 3-5
syntax error, 3-4
Syntax Overview, 3-5

System Subsystem, 7-1
DSP Command/Query, 7-3
ERRor Query, 7-4
HEADer Command/Query, 7-6
KEY Command/Query, 7-7
LONGform Command/Query, 7-9
SETup Command/Query, 7-10
Syntax Diagram, 7-2

T

Talking to the instrument, 1-2
TDELta Query, 14-41
TER Query, 6-19
Terminator, 1-3, 1-9
TEXT Command, 11-19
Timebase Subsystem, 15-1
DELay Command/Query, 15-3
MODE Command/Query, 15-4
RANGe Command/Query, 15-5
REFerence Command/Query, 15-6
WINDow Command/Query, 15-7
WINDow:DELay Command/Query, 15-8
WINDow:RANGe Command/Query, 15-9
Timebase Window mode, 15-1
TMARker Command/Query, 11-20
TMAX Query, 14-42
TMIN Query, 14-43
TNULl Command/Query, 9-2
*TRG Command, 5-23
TRG - trigger, 3-31
Trigger Bit, 3-31
Trigger Condition command, 16-4 / 16-5
Trigger Delay command, 16-6
Trigger Delay Source command, 16-6
Trigger Event, 16-6
Trigger Event command, 16-6
Trigger Field command, 16-7
Trigger Holdoff, 16-4
trigger level, 6-4
Trigger Level command, 16-1, 16-3
Trigger Logic command, 16-4 / 16-5
Trigger Mode, 16-1
EDGE, 16-1, 16-3
TV, 16-1
Trigger Occurrence command, 16-6, 16-8
Trigger Occurrence Slope command, 16-8
Trigger Path command, 16-4 / 16-5
Trigger Polarity command, 16-7
Trigger Qualify command, 16-6, 16-8
Trigger Slope command, 16-3
Trigger Source command, 16-3, 16-7
Trigger Standard command, 16-7
Trigger Subsystem, 16-1
CONDition Command/Query, 16-13
DELay Command/Query, 16-16
DELay SOURce Command/Query, 16-18
DELay:SLOPe Command/Query, 16-17
FIELd Command/Query, 16-19
HOLDoff Command/Query, 16-20
LEVel Command/Query, 16-21
LINE Command/Query, 16-22
LOGic Command/Query, 16-23
MODE Command/Query, 16-24
OCCurrence Command/Query, 16-25
OCCurrence:SLOPe Command/Query, 16-26
OCCurrence:SOURce Command/Query, 16-27
PATH Command/Query, 16-28
POLarity Command/Query, 16-29
QUALify Command/Query, 16-30
SLOPe Command/Query, 16-32
SOURce Command/Query, 16-33
STANdard Command/Query, 16-34
Syntax Diagram, 16-9
Trigger Time command, 16-6
TRIGger:MODE
Command Table, 16-2
TRIGGERED timebase mode, 15-4
Truncation Rules, 4-1
*TST Query, 5-24
TSTArt Command/Query, 14-44
TSTOp Command/Query, 14-45

TTL Command, 10-9
TV Trigger Mode, 16-1, 16-7, 16-13
TVOLT Query, 14-46
TYPE Command/Query, 8-7
TYPE Query, 17-17

U

UNITS Command/Query, 14-48
Unterminated Condition, 3-4
UPPer Command/Query, 14-49
Upper/Lower Case Equivalence, 3-8
Uppercase, 1-8
URQ - user request, 3-30
URQ, user request, 5-5
user-defined + width, A-4
User-Defined Measurements, 14-1

V

Vamp, A-6
VAMPlitude Command/Query, 14-50
VAVerage Command/Query, 14-51
Vavg, A-6
Vbase, A-6
VBASe Command/Query, 14-52
VDELta Query, 14-53
VERSus Command, 12-11
vertical input, 6-4
vertical offset, 6-4
vertical sensitivity, 6-4
VFIFty Command, 14-54
VIEW Command, 6-20
VMARker Command/Query, 11-21
Vmax, A-5
VMAX Command/Query, 14-55
Vmin, A-5
VMIN Command/Query, 14-56
Voltage measurements, 14-2
Vp-p, A-5
VPP Command/Query, 14-57
VRELative Command/Query, 14-58
Vrms, A-6
VRMS Command/Query, 14-60
VSTArt Command/Query, 14-61
VSTOp Command/Query, 14-62
VTIMe Query, 14-63
Vtop, A-5
VTOP Command/Query, 14-64

W

*WAI Command, 5-25
Waveform Data Conversion, 17-4
Waveform Subsystem, 17-1
  COUNt Query, 17-9
  DATA Command/Query, 17-10
  FORMat Command/Query, 17-12
  POINts Query, 17-13
  PREamble Command/Query, 17-14
  SOURce Command/Query, 17-16
  Syntax Diagram, 17-7
  TYPE Query, 17-17
  XINCrement Query, 17-18
  XORigin Query, 17-19
  XREFerence Query, 17-20
  YINCrement Query, 17-21
  YORigin Query, 17-22
  YREFerence Query, 17-23
white space, 3-8
WINDow Command/Query, 15-7
WINDow:DELay Command/Query, 15-8
WINDow:RANGe Command/Query, 15-9
WORD format, 17-5

X

XINCrement Query, 17-18
XORigin Query, 17-19
XREFerence Query, 17-20

Y

YINCrement Query, 17-21
YORigin Query, 17-22
YREFerence Query, 17-23