

---

# **Programming Reference Manual**

**HP 54121T Digitizing Oscilloscope  
Consisting of: HP 54120B  
HP 54121A**

---





---

## **Product Warranty**

This Hewlett-Packard product has a warranty against defects in material and workmanship for a period of 1 year 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 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 Buyer's facility only upon Hewlett-Packard's prior agreement and 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. Addresses are provided at the back of this operating note.

### **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 operating note must be heeded.



# 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 and/or firmware code may be printed before the date; this indicates the version level of the software and/or firmware of this product at the time of 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

July 1989

54121-90907

# 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 Printing History and on the title page.

| Pages         | Effective Date |
|---------------|----------------|
| all . . . . . | July 1989      |

# Contents

---

## Chapter 1

|      |  |
|------|--|
|      | <b>Introduction to Programming an Oscilloscope</b> |
| 1-1  | Introduction                                       |
| 1-2  | Programming Syntax                                 |
| 1-2  | Talking to the Instrument                          |
| 1-3  | Addressing the Instrument and the I/O Interface    |
| 1-3  | Initialization                                     |
| 1-4  | Program Syntax                                     |
| 1-4  | Separator  |
| 1-4  | Command Header                                     |
| 1-6  | Query Command                                      |
| 1-7  | Longform/Shortform                                 |
| 1-7  | Program Data                                       |
| 1-8  | Program Message Terminator                         |
| 1-8  | Selecting Multiple Subsystems                      |
| 1-9  | Summary  |
| 1-9  | <b>Programming an Oscilloscope</b>                 |
| 1-9  | Autoscale  |
| 1-9  | Setting Up the Oscilloscope                        |
| 1-10 | Receiving Information from the Instrument          |
| 1-13 | String Variables                                   |
| 1-14 | Numeric Variables                                  |
| 1-14 | Multiple Queries                                   |
| 1-15 | Status   |
| 1-16 | Definite-Length Block Response Data                |
| 1-17 | Digitize Command                                   |
| 1-19 | Example Program                                    |
| 1-20 | Program Overview                                   |

---

## Chapter 2

### Interface Functions

- 2-1 Introduction
  - 2-1 Interface Capabilities
  - 2-1 Command and Data Concepts
  - 2-2 Addressing
  - 2-3 Remote, Local and Local Lockout
  - 2-3 Bus Commands
    - 2-3 Device Clear
    - 2-3 Group Execute Trigger (GET)
    - 2-4 Interface Clear (IFC)
  - 2-4 Status Annunciators
- 

## Chapter 3

### Message Communication and System Functions

- 3-1 Introduction
- 3-1 Protocols
  - 3-1 Functional Elements
- 3-2 Protocol Overview
- 3-2 Protocol Operation
- 3-3 Protocol Exceptions
- 3-5 Syntax Diagrams
- 3-5 Syntax Overview
- 3-8 Device Listening Syntax
- 3-21 Device Talking Syntax
- 3-27 Common Commands
- 3-28 Status Reporting
  - 3-29 Bit Definitions
  - 3-30 Key Features
  - 3-31 Serial Poll
  - 3-32 Parallel Poll

---

## Chapter 4

### Programming and Documentation Conventions

- 4-1 Introduction
  - 4-1 Truncation Rule
  - 4-2 The Command Tree
    - 4-2 Command Types
    - 4-4 Tree Traversal Rules
    - 4-4 Examples
  - 4-5 Infinity Representation
  - 4-6 Sequential and Overlapped Commands
  - 4-6 Response Generation
  - 4-6 Notation Conventions and Definitions
  - 4-7 Syntax Diagrams
  - 4-8 Command Structure
    - 4-8 Common Commands
    - 4-8 Root Level Commands
    - 4-8 Subsystem Commands
  - 4-10 Program Examples
  - 4-11 Command Set Organization
- 

## Chapter 5

### Common Commands

- 5-1 Introduction
- 5-3 \*CLS
- 5-4 \*ESE
- 5-6 \*ESR?
- 5-8 \*IDN?
- 5-9 \*LRN?
- 5-10 \*OPC
- 5-11 \*RCL
- 5-12 \*RST
- 5-14 \*SAV
- 5-15 \*SRE
- 5-17 \*STB?
- 5-19 \*TRG
- 5-20 \*TST?
- 5-21 \*WAI

---

## Chapter 6

### Root Level Commands

|      |              |
|------|--------------|
| 6-1  | Introduction |
| 6-4  | AUToscale    |
| 6-5  | BLANk        |
| 6-6  | DIGitize     |
| 6-8  | EOI          |
| 6-9  | ERASe        |
| 6-10 | LER?         |
| 6-11 | MENU         |
| 6-12 | MERGe        |
| 6-13 | PLOT?        |
| 6-14 | PRINT?       |
| 6-15 | RUN          |
| 6-16 | SER          |
| 6-17 | STOP         |
| 6-18 | STORe        |
| 6-19 | TER?         |
| 6-20 | VIEW         |

---

## Chapter 7

### System Subsystem

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

---

## **Chapter 8**

### **Acquire Subsystem**

- 8-1 Introduction
  - 8-1 Persistence (Normal) Mode
  - 8-2 Averaging Mode
  - 8-4 BANDwidth
  - 8-5 COMPlete
  - 8-6 COUNT
  - 8-8 POINTs
  - 8-9 TYPE
    - 8-9 Type Average or Normal
    - 8-9 Type Histogram
    - 8-9 Type Envelope
- 

## **Chapter 9**

### **Calibrate Subsystem**

- 9-1 Introduction
  - 9-2 DATA
  - 9-4 GAIN
- 

## **Chapter 10**

### **Channel < N > Subsystems**

- 10-1 Introduction
- 10-3 OFFSet
- 10-4 PROBe
- 10-5 RANGe

---

## Chapter 17

### Timebase Subsystem

- 17-1 Introduction
  - 17-3 DELay
  - 17-4 GRATe
  - 17-5 MODE
  - 17-6 RANGe
  - 17-7 REFerence
- 

## Chapter 18

### Trigger Subsystem

- 18-1 Introduction
  - 18-3 HFRejct
  - 18-4 LEVel
  - 18-5 MODE
  - 18-6 PROBe
  - 18-7 SENSitivity
  - 18-8 SLOPe
  - 18-9 SOURce
- 

## Chapter 19

### Waveform Subsystem

- 19-1 Introduction
- 19-2 Data Acquisition Types
  - 19-2 Normal
  - 19-2 Average
  - 19-2 Histogram
- 19-3 Envelope
- 19-5 Data Conversion
  - 19-5 Conversion from Data Value to Voltage
  - 19-5 Conversion from Data Value to Time
- 19-6 Data Format for HP-IB Transfer
  - 19-6 WORD Format
  - 19-6 ASCII Format
  - 19-6 LONG Format
- 19-9 COUNT?
- 19-10 DATA



19-12 FORMat  
19-13 POINts?  
19-14 PREamble  
19-16 SOURce  
19-17 TYPE  
19-18 VALid?  
19-19 XINCrement?  
19-20 XORigin?  
19-21 XREFerence?  
19-22 YINCrement?  
19-23 YORigin?  
19-24 YREFerence?



# 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 18. 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.

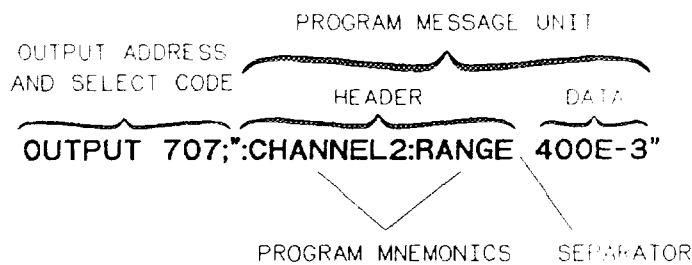
Chapter 1 deals mainly with how to set up the instrument, how to retrieve setup information and measurement results, how to digitize a waveform, and how to pass data to the controller. This chapter is divided into two sections. The first section (page 1-2) concentrates on program syntax, and the second section (page 1-9) 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.*

## Program Syntax

To program the instrument over the HP-IB, you must have some understanding of the command format and structure expected by the instrument. The instrument is remotely programmed with HP-IB 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, and program data. 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 Header

The command header is the program mnemonic or mnemonics that represent the operation to be performed by the instrument. The different types of command 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 oscilloscopes. The syntax is:

<program mnemonic> <terminator>

When program data must be included with the simple command header (for example, :DIGITIZE CHAN1, 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 that appear between the subsystem mnemonic and the function mnemonic select subsystems of subsystems. 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 CHANNEL1:RANGE 0.4)

- To use a subsystem within a subsystem:

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

- To execute more than one function within the same subsystem:

: <subsystem> : <function> <separator> <data> ; <function> <separator> <data> <terminator>  
(For example CHANNEL1:RANGE 0.4;OFFSET 0.5)

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 generic functions within the instrument (such as reset, etc.). Their syntax is:

\* <command header> <terminator>

No space or separator is allowed between the asterisk and the command header. **\*RST** 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 HP-IB output queue. The output message remains in the queue until it is read or another command is issued. When read, the message 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 command:

ENTER <device address> ;Range

passes the value across the HP-IB 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 HP-IB 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 command **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 message to appear on screen.*

## Longform/ Shortform

Program commands can be a combination of uppercase or lowercase ASCII characters, whereas the instrument responses are always returned in uppercase.

Both program commands and queries may be used in either longform (complete spelling) or in shortform (abbreviated spelling). Either of the following examples changes the timebase to 0.1 seconds per division and delays the on-screen presentation (with respect to the trigger event) by 1 second in time.

:TIMEBASE:RANGE 1;DELAY 1 - longform  
:TIM:RANG 1;DEL 1 - 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, :MEASURE:SOURCE CHAN1,CHAN2 has two data parameters: CHAN1 and CHAN2.

**Character Program Data.** Character program data is used to convey parameter information as short alpha or alphanumeric strings. For example, the display **GRATICULE** can be set to grid axes, frame, or off. The character program data in this case may be **GRID**, **AXES**, **FRAME**, or **OFF**. **:DISPLAY:GRATICULE OFF** turns the display graticule off.

**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.

## 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-Of-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 relates directly to the **EOS** (End Of String) and **EOT** (End Of Text) terminators.*

## 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>

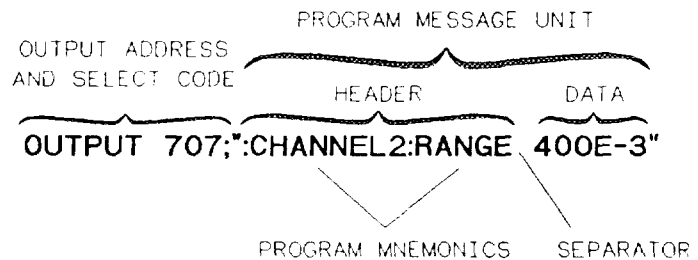
**:CHANNEL1:RANGE 0.4;;TIMEBASE:RANGE 1**

### Note

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



**Summary** The following illustration summarizes the syntax for programming over the HP-IB.



---

## Programming an Oscilloscope

**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 Oscilloscope** 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 m s 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 HP-IB output queue. The output message remains in the queue until it is read or another command is issued. When read, the message is transmitted across the bus to the designated listener (typically a controller). You read the bus by using an enter command and passing the device address. For example, to read the result of the query command **:MEASURE:RISETIME?** you would execute the statement:

```
ENTER <device address>;Risetime
```

where **<device address>** represents the address of your device. This would enter the value from your risetime measurement in the variable **Risetime**.

### 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 by the command **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 message to appear on screen.*

*The actual **ENTER COMMAND** you use when programming is dependent on the programming language you are using.*

The format of the returned ASCII string depends on the current settings of the system subsystem **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 **:MEAS:FREQ?** 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

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

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 > ;Risetime$
```

places the output of the query in the string variable Risetime\$.

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 address 707. This 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$[30]
20 OUTPUT 707;":CHANNEL1:RANGE?"
30 ENTER 707;Rang$
40 PRINT Rang$
50 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 oscilloscope, then you can use a numeric variable. When you are receiving numeric data, turn the headers 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;";CHAN1:RANGE?"  
30 ENTER 707;Rang  
40 PRINT Rang  
50 END
```

After running this program, the controller displays:

.1

## 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 **:MEASURE:RISETIME?;VPP?** into the string variable **Mresults\$** with the command:

```
ENTER 707;Mresults$
```

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 **:MEASURE:RISETIME?;VPP?** into multiple numeric variables:

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

## 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.

## 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:

NUMBER OF DIGITS  
THAT FOLLOW

/

ACTUAL DATA

#280<eighty bytes of data><terminator>

NUMBER OF BYTES  
THAT FOLLOW

The "2" states the number of digits that follow, and "80" states the number of bytes to be transmitted.

## 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 and placed in waveform memory. To obtain waveform data, you must specify the waveform source for the waveform data prior to sending the :WAVEFORM:DATA? query. For example:

```
DIGITIZE CHAN1::WAVEFORM:SOURCE WMEM1;FORMAT WORD;DATA? < terminator >
```

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 through the program and may be selected as **ASCII**, **WORD**, or **LONG**.

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 32.767. 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" and the programming examples in Appendix B.

< signed 6 digit integer > , < signed 6 digit integer > ... < terminator >

The number of data points comprising a waveform varies according to the number requested in the ACQUIRE subsystem. This ACQUIRE subsystem determines the number of data points, type of acquisition, and number of averages used by the DIGITIZE command. This allows you to specify exactly what the digitized information will contain. A typical acquisition setup is:

ACQUIRE:TYPE AVERAGE;COUNT 4;POINTS 500<terminator>

This setup places the instrument into the averaged mode with four averages and defines the data record to be 500 points. This means that when the DIGITIZE command is received, the waveform will not be stored into memory until 500 points have been averaged at least four times.

### Example Program

This program demonstrates the basic command structure used to program the instrument. The HP 54121A Four Channel Test Set is used as the input. Make sure the coaxial short is on the CHANNEL 1 input prior to running this program.

```
10  CLEAR 707                                !Initialize instrument interface
20  OUTPUT 707;"*RST"                         !Reset Scope to known default state
30  OUTPUT 707;":DISPLAY:GRATICULE FRAME;FORMAT 1" !Setup display
40  DIM Mresults$(60)                        !Reserve memory for string variable
50  OUTPUT 707;":VIEW CHANNEL1"              !Turn on channel 1
60  OUTPUT 707;":NETWORK:REFLECTION:PRESET"   Turn TDR step generator on
70                                           !get known waveform on screen
80  OUTPUT 707;":TIMEBASE:RANGE 2.0E-7"       !Setup Timebase Range
90  OUTPUT 707;":CHANEL1:RANGE 0.4;OFFSET 0.133" !Setup Channel Range and Offset
100
110 OUTPUT 707;":SYSTEM:HEADER OFF"           !Headers must be off for numeric!
120                                           !variables
130 OUTPUT 707;":TIMEBASE:RANGE?;:CHANNEL 1:RANGE?;OFFSET"
140 ENTER 707;Setup1, Setup2, Setup3          !Enter instrument setup in numeric
150                                           !variables
160 PRINT Setup1, Setup2, Setup3              !Verify instrument setup
170 OUTPUT 707;":MEASURE;PWIDTH?"            !Measure Pulse Width
180 Enter 707;Pwid                           !Enter result in a numeric variable
190
200 PRINT Pwid                               !Print result of measurement
210 OUTPUT 707;":SYSTEM:HEADER ON;LONGFORM ON" !Turn headers and longform on
220 OUTPUT 707;":MEASURE:RISETIME?;VPP?"      !Measure Risetime and peak-to-peak
230                                           !voltage
240 ENTER 707;Mresults$                      !Enter results in a string variable
250
260 PRINT Mresults$                          Print results of measurements
270  END
```



## Program Results

While running the program, the following information will appear on screen after executing line 160:

2.0E-7 .4 .133

- This first message confirms that the timebase range has been set to 2.0E-7 s full scale (20 ns/division) and the channel 1 range has been set to 0.4 V full scale (50 mV/division) with an offset of 0.133 V.
- The result of the pulse width measurement appears on screen when line 200 is executed:

5.004E – 10

- Since the results of the query on line 220 were placed in a string variable with the headers on, the result for risetime and peak-to-peak voltage measurements are displayed on screen with the appropriate headers.

:MEASURE:RISETIME + 4.24000E-11;:MEASURE:VPP + 2.07813E-01

### Note

*The actual returned values may vary slightly due to variations in the input signal.*

## Program Overview

The first thing the program does is to reset the instrument to a known state. Then the program sets up the display for measurements and instrument responses to queries.

The program uses a signal generated by the HP 54121A as an input and sets up the instrument to look at this signal on Channel 1. Then it queries the instrument's configuration to confirm the instrument's setup parameters. At this point the program queries the pulse width of the signal and places the result in the numeric variable Pwid. The result is then printed on screen.

Next, the program queries the risetime and peak-to-peak voltage value of the signal and places the result in the string variable Mresults\$. Since the HEADER and LONGFORM commands are on the results are printed on screen in longform with the appropriate headers.



# 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 54121T, as defined by IEEE 488.1 are SH1, AH1, T5, L4, SR1, RL1, PP0, DC1, DT1, C0, and E1.

---

## 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. These device-dependent messages include all of the instrument commands and responses found in chapters 5 through 19 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 plotter or 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 54121T is in remote mode, all front-panel input is disabled except for the local key and the line switch. In remote with local lockout mode, all controls (except the power switch) are entirely locked out. Local control can only be restored by the controller.

### 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).

---

## 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 an 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 command will cause the same action as the RUN command: the instrument will acquire data for the active waveform display.

**Interface Clear (IFC)** This 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 54121T 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, no message is displayed.

# Message Communication and System Functions

---

## Introduction

This chapter describes the operation of instruments that operate in compliance with the IEEE 488.2 standard. The IEEE 488.2 standard is a new 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 54121T's input buffer will hold 274 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 54121T's output queue will hold 510 characters, however the instrument will handle block data of greater than 510 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 powered 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. A group execute trigger (GET) sent in the middle of a < program message > will also cause a command error.

**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 are 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.

---

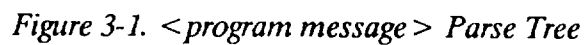
## **Syntax Overview**

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.



## 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.

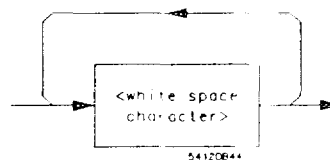


Figure 3-2. <white space>

**< 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.

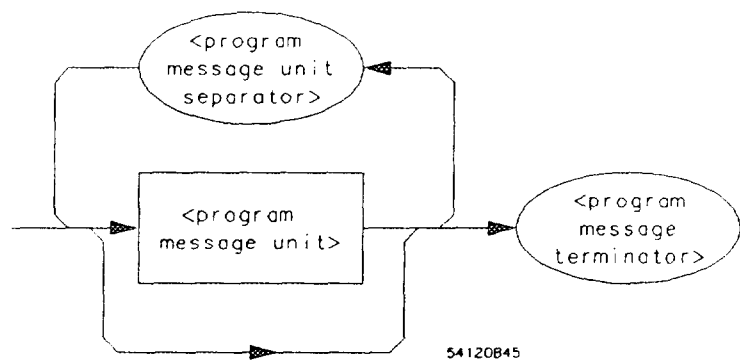


Figure 3-3. **< program message >**

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

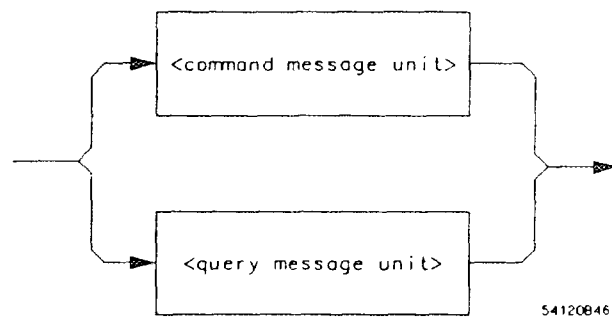


Figure 3-4. **< program message unit >**

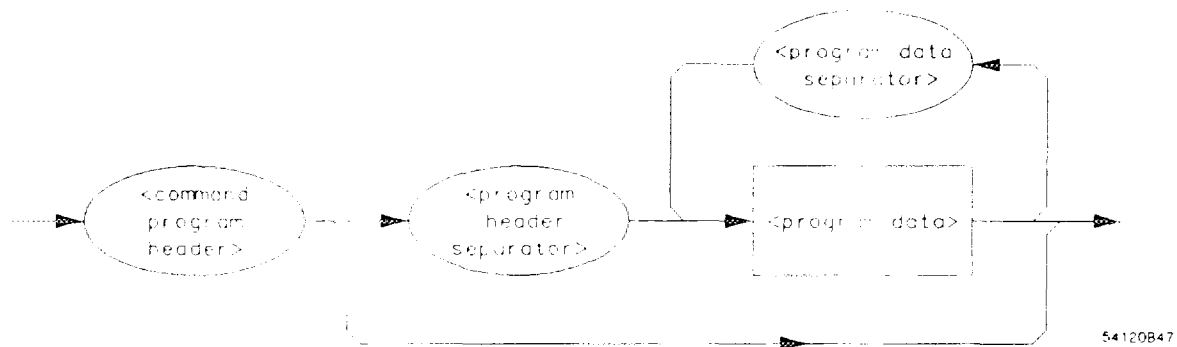


Figure 3-5. <command message unit>

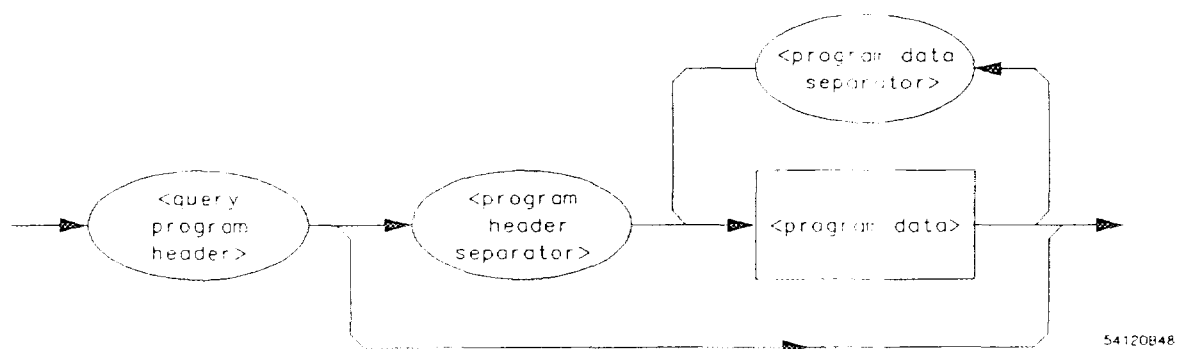


Figure 3-6. <query message unit>



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

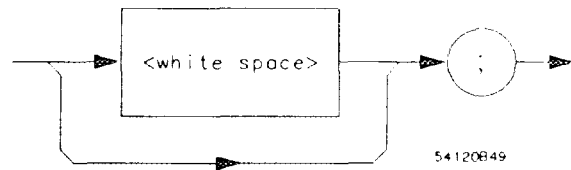


Figure 3-7. <program message unit separator>

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

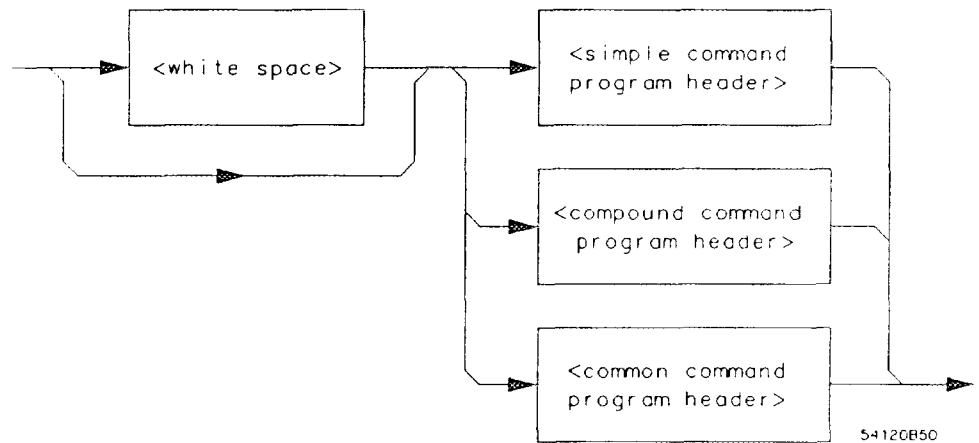
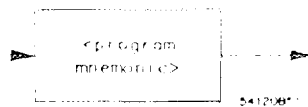
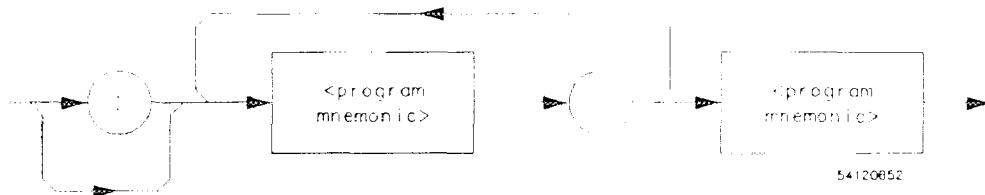


Figure 3-8. <command program header>

Where < simple command program header > is defined as



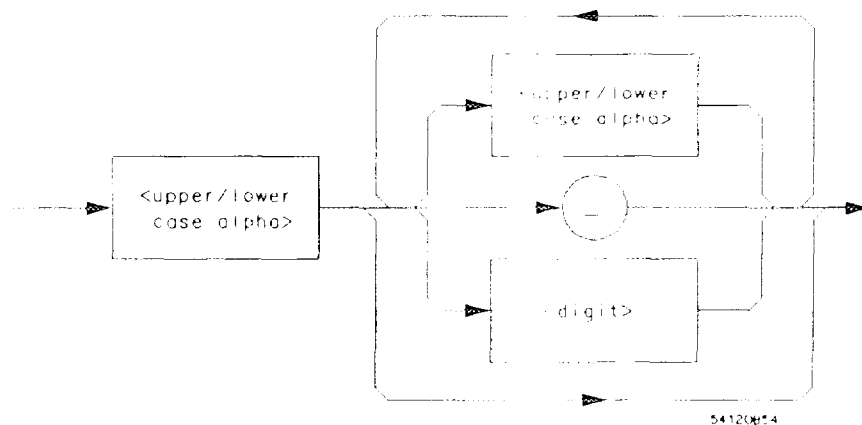
Where < compound command program header > is defined as



Where < common command program header > is defined as



Where < program mnemonic > is defined as

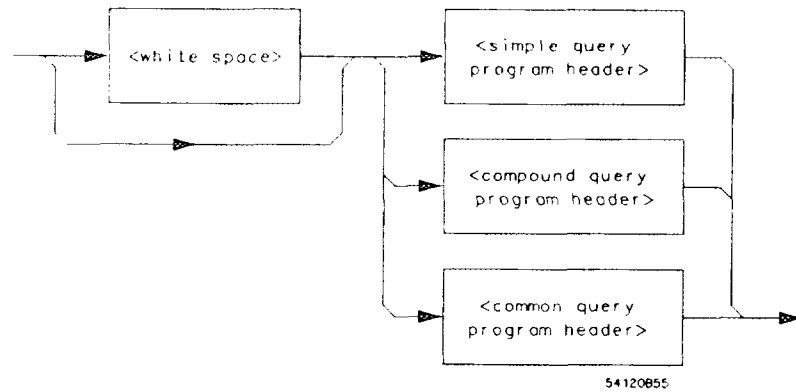


Where < upper / lower case alpha > is defined as a single ASCII encoded byte in the range 41 - 5A, 61 - 7A (65 - 90, 97 - 122 decimal).

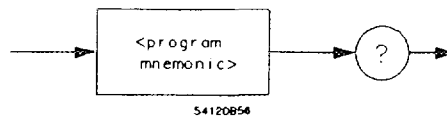
Where < digit > is defined as a single ASCII encoded byte in the range 30 - 39 (48 - 57 decimal).

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

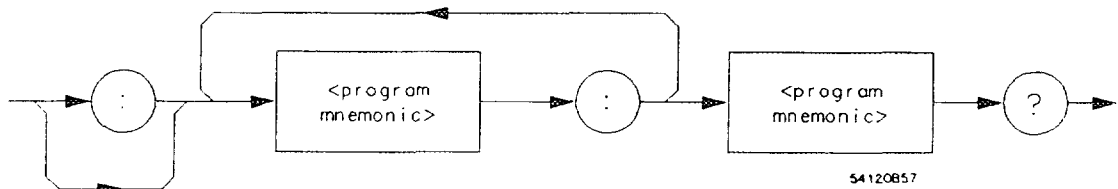
Figure 3-8. < command program header > (continued)



Where < simple query program header > is defined as



Where < compound query program header > is defined as



Where < common query program header > is defined as

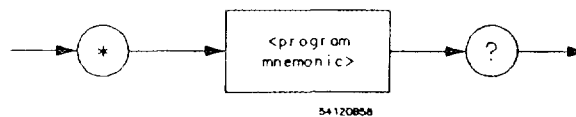
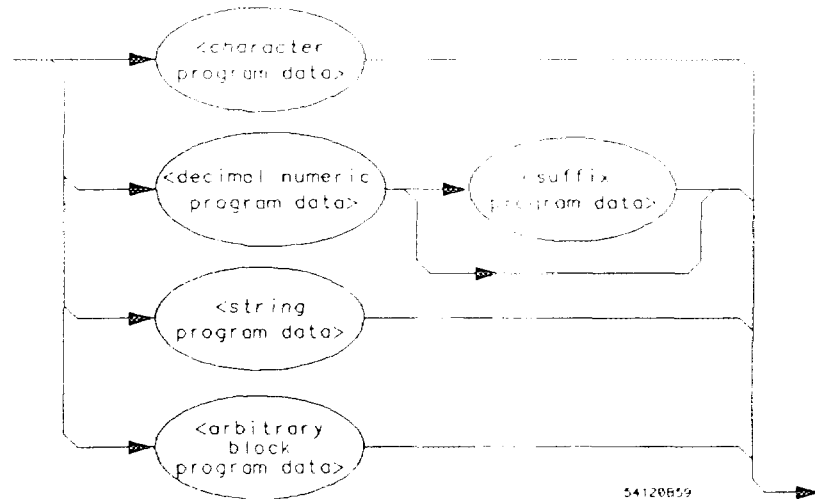
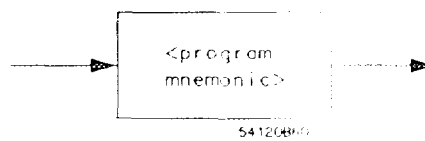


Figure 3-9. < query program header >

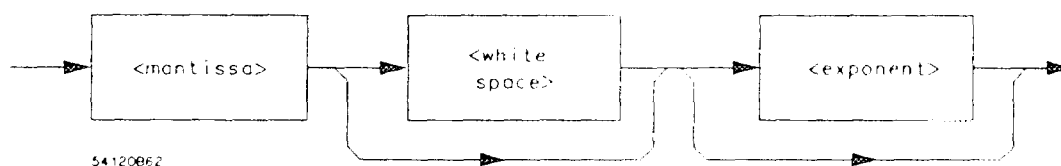
**< program data >**. The **< program data >** element represents the possible types of data which may be sent to the instrument. The HP 54121T will accept the following data types: **< character program data >**, **< decimal numeric program data >**, **< suffix program data >**, **< string program data >**, and **< arbitrary block program data >**.



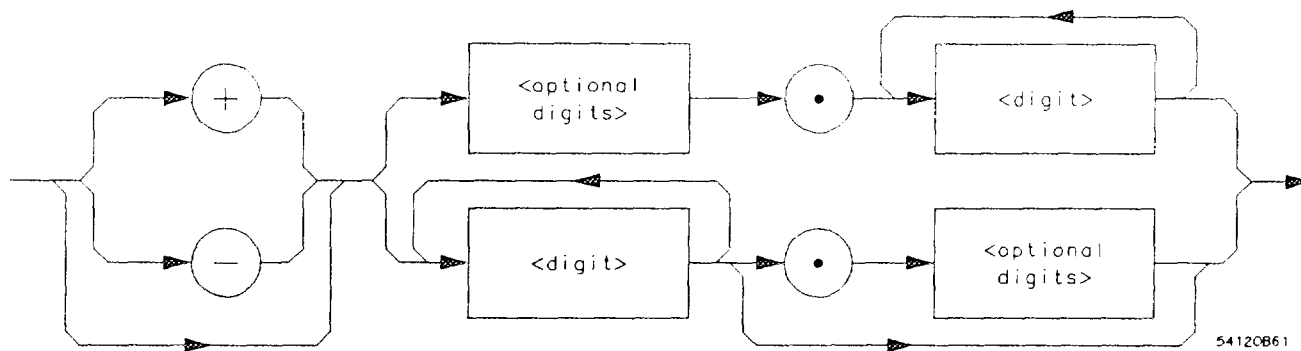
*Figure 3-10. <program data>*



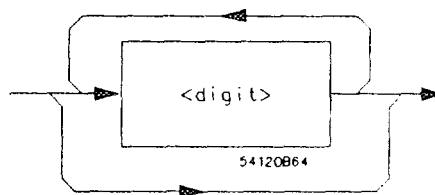
*Figure 3-11. <character program data>*



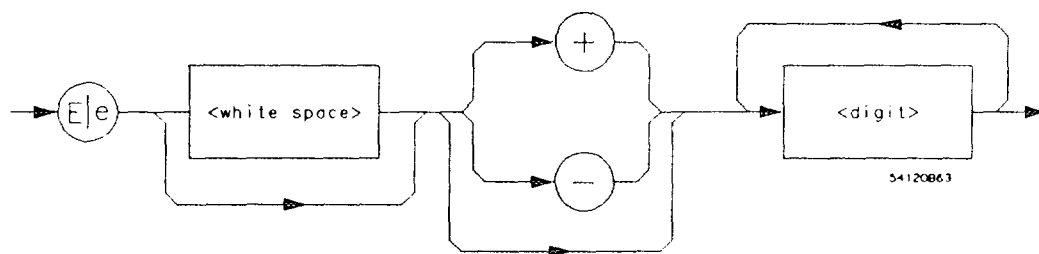
Where **< mantissa >** is defined as



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.

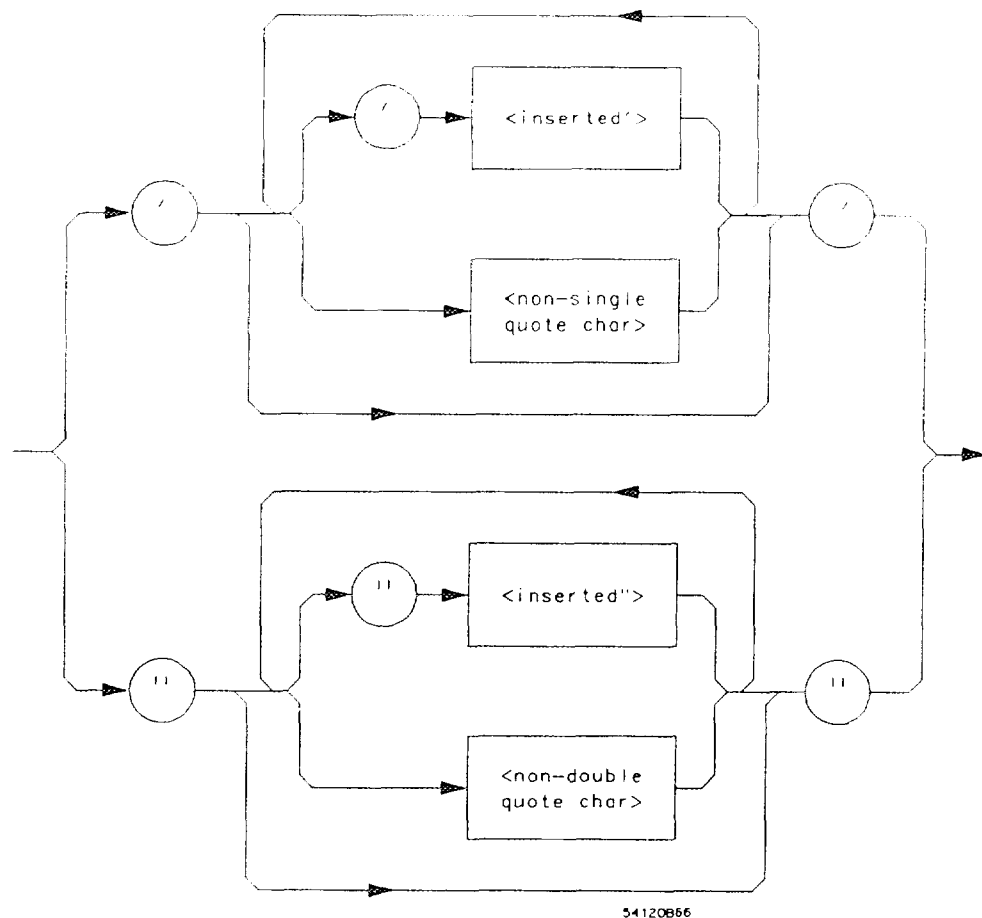
Table 3-1. <suffix mult>

| 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        |

**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            |



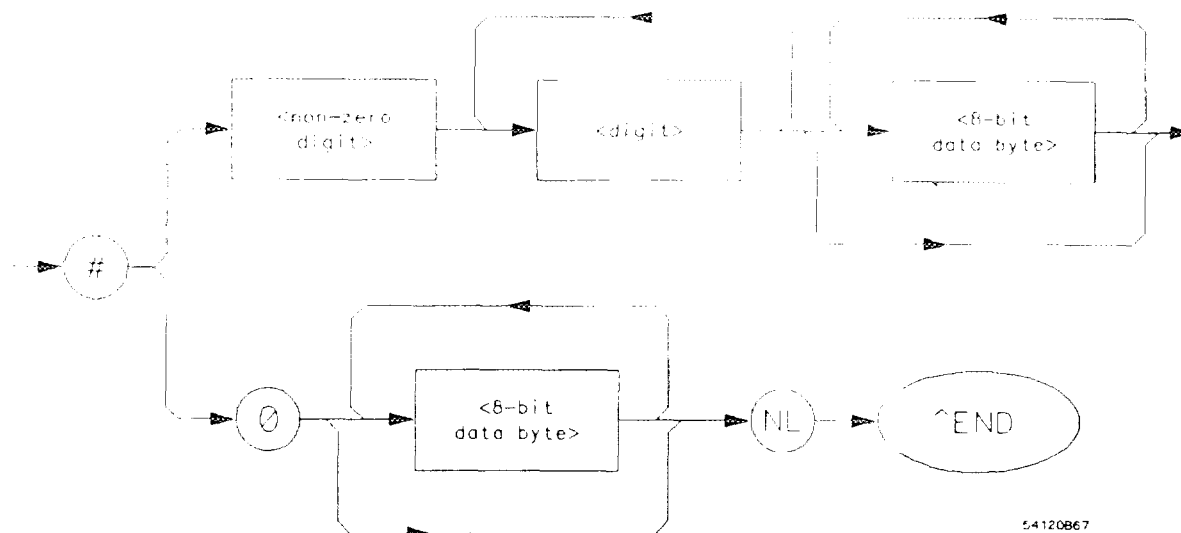
Where <inserted ' > is defined as a single ASCII character with the value 27 (39 decimal).

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.

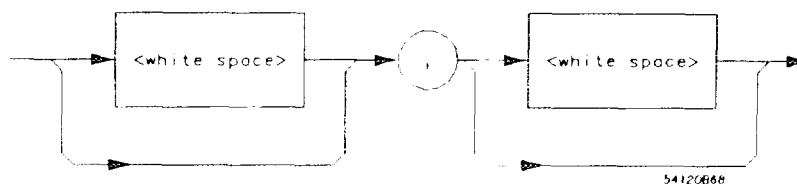


Figure 3-16. <program data separator>



**< program header separator >** . A space separates the header from the first or only parameter of the command.

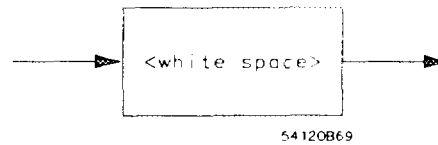
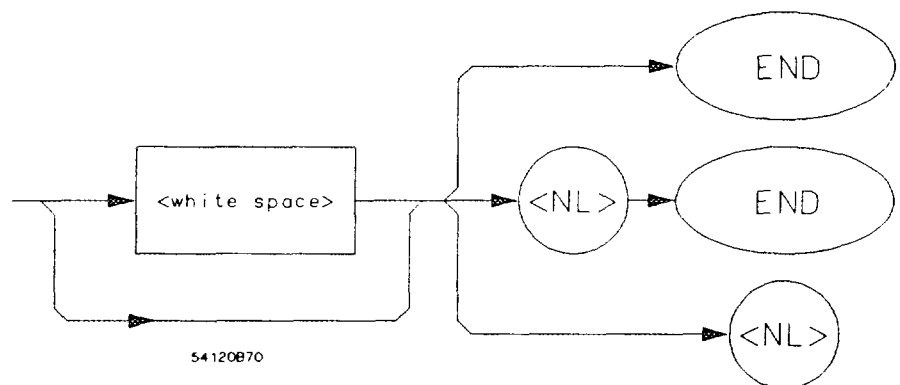


Figure 3-17. <program header separator>

**< 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.



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

Figure 3-18. <program message terminator>



## 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's 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.

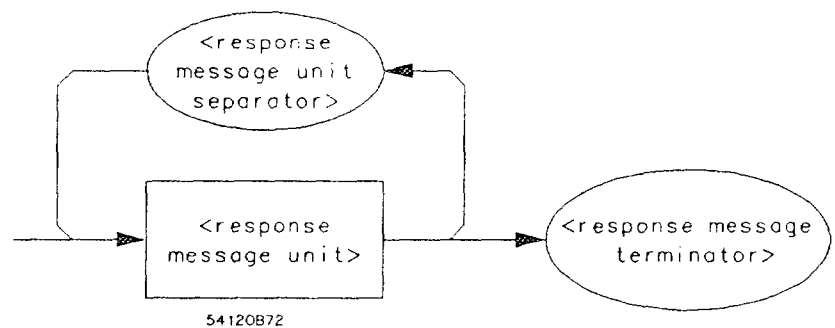
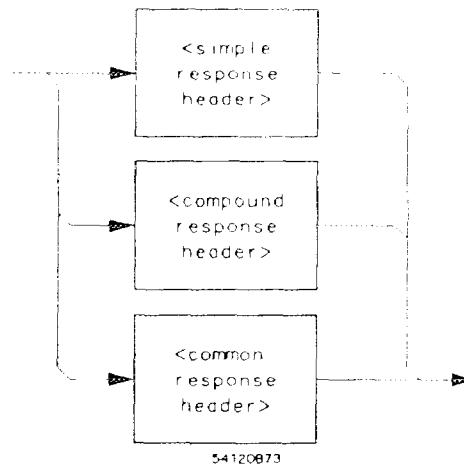


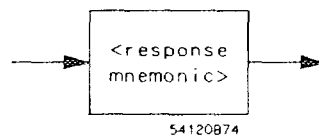
Figure 3-20. <response message>

**<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.

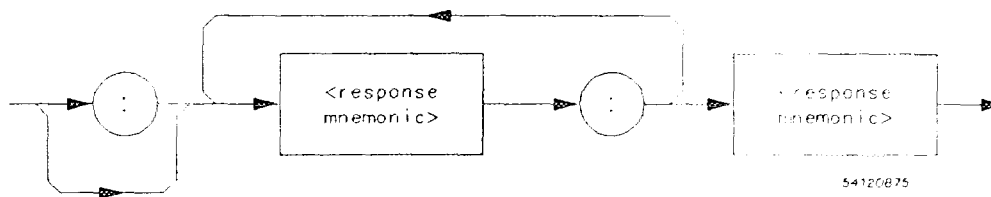
**<response header>**. The <response header>, when returned indicates what the response data represents.



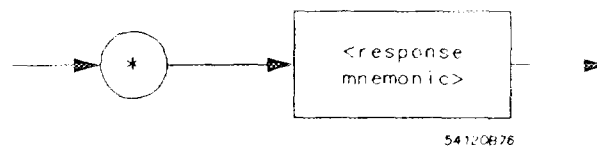
Where **< simple response mnemonic >** is defined as



Where **< compound response header >** is defined as

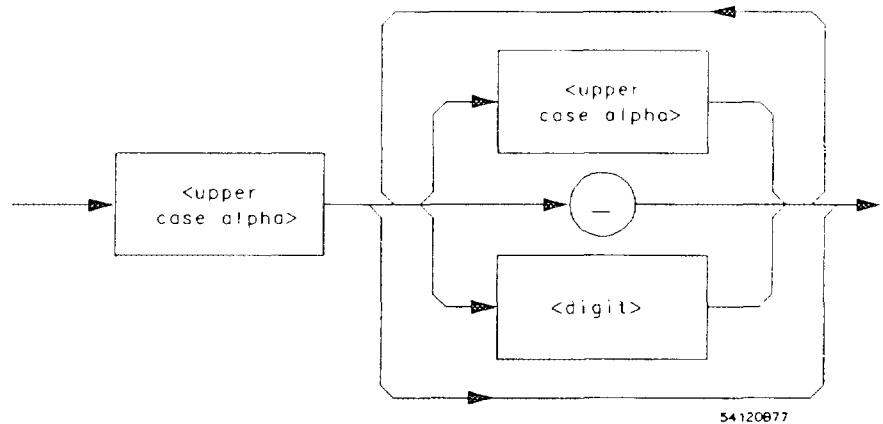


Where **< common response header >** is defined as



*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 >, < nr3 numeric response data >, < string response data >, < definite length arbitrary block response data >, and < arbitrary ASCII response data >.

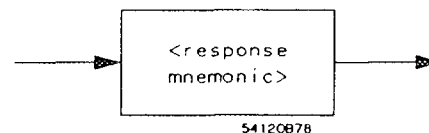


Figure 3-22. <character response data>

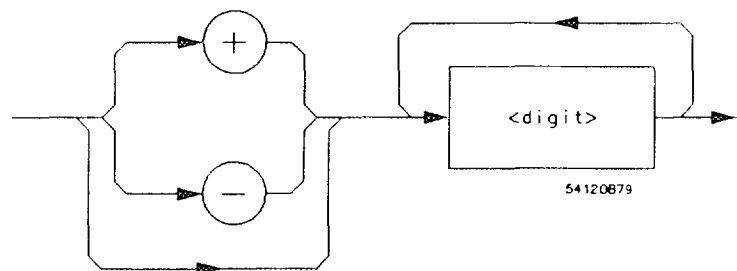


Figure 3-23. <nr1 numeric response data>

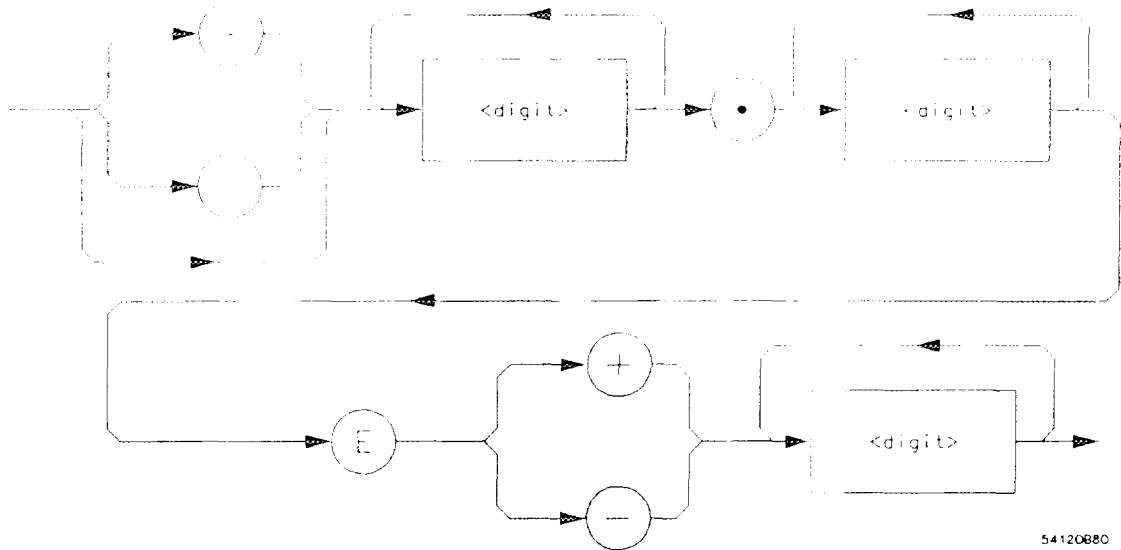


Figure 3-24. <nr3 numeric response data>

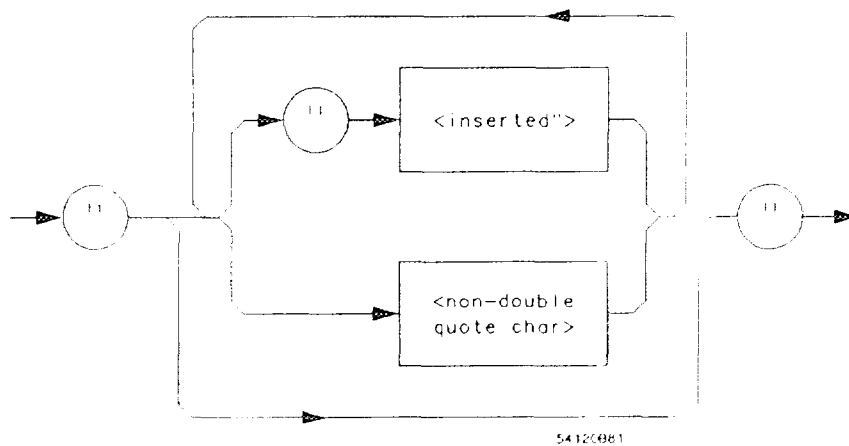


Figure 3-25. <string response data>

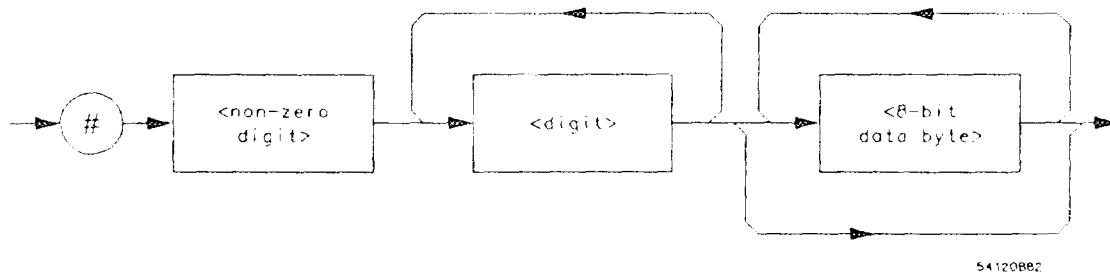
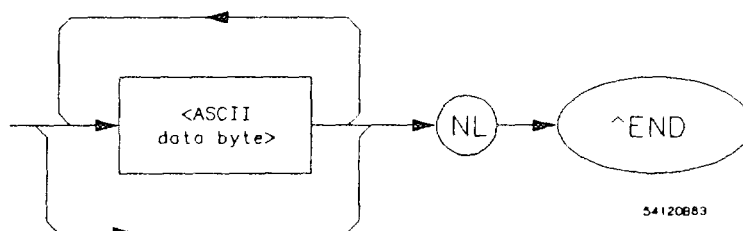


Figure 3-26. <definite length arbitrary block response data>



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

Notes:

1. The END message provides an unambiguous termination to an element that contains arbitrary ASCII characters.
2. The IEEE 488.1 END message serves the dual function of 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 present 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>.

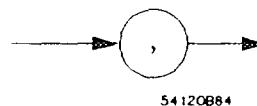


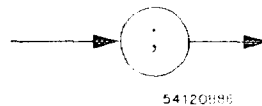
Figure 3-28. <response data separator>

**<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 54121T will produce an interrupted error.*



## 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 54121T'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           |
| *LRN?   | Learn Device Setup Query       |
| *OPC    | Operation Complete Command     |
| *OPC?   | Operation Complete 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.

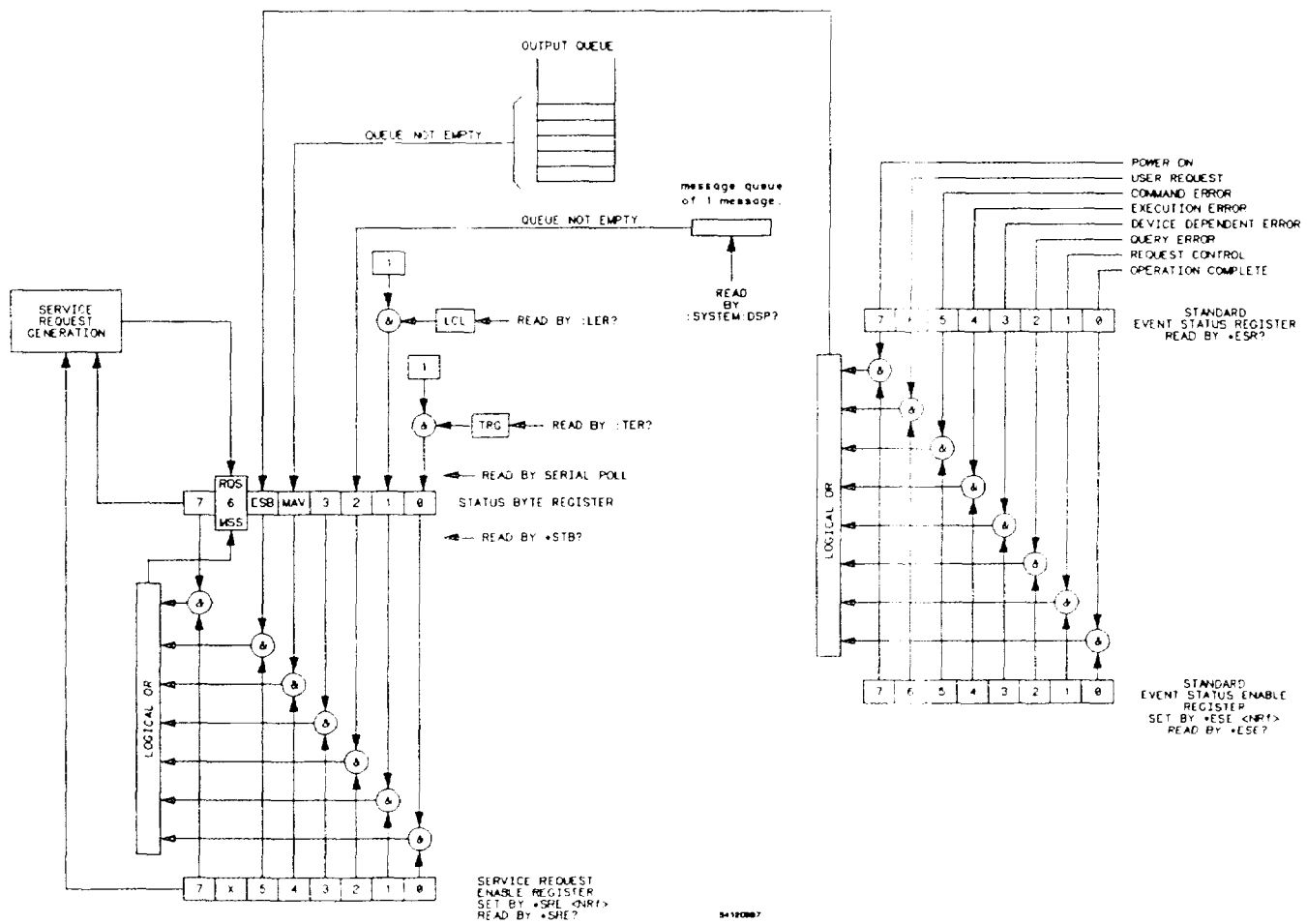


Figure 3-31. Status Byte Structures and Concepts

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.

## 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** - 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** - Indicates whether there is a message in the message queue.
- PON** - power on. Always 0 in the HP 54121T.
- 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 54121T will never request control.
- OPC** - operation complete. Indicates whether the device has completed all ending operations.
- LCL** - Indicates whether a remote to local transition has occurred.
- TRG** - Indicates whether a trigger has been received.

## 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,<br>!will generate a SRQ when the<br>!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.

|                     |   |
|---------------------|---|
| OUTPUT 707;"*SRE 1" | !enables a trigger service request.<br>!the next trigger will generate an SRQ . |
|---------------------|---|

|                    |  |
|--------------------|--|
| OUTPUT 707;"*TER?" | !queries the TRG event register, thus                          |
| ENTER 707;A\$      | !clearing it.<br>!the next trigger can now generate an<br>!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 54121T 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 54121T 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.

**Parallel Poll**    The HP 54121T does not support the parallel poll feature.

# Programming and Documentation Conventions

---

## 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.

---

## Truncation Rule

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 command are shown in table 4-1.

*Table 4-1. Mnemonic Truncation*

| Longform | Shortform |
|----------|-----------|
| RANGE    | RANG      |
| PATTERN  | PATT      |
| TIME     | TIME      |
| DELAY    | DEL       |

---

## The Command Tree

The command tree (figure 4-1) shows all commands in the HP 54121T and the relationship of the commands to each other. You should notice that the IEEE 488.2 common commands are not actually included with the command 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.

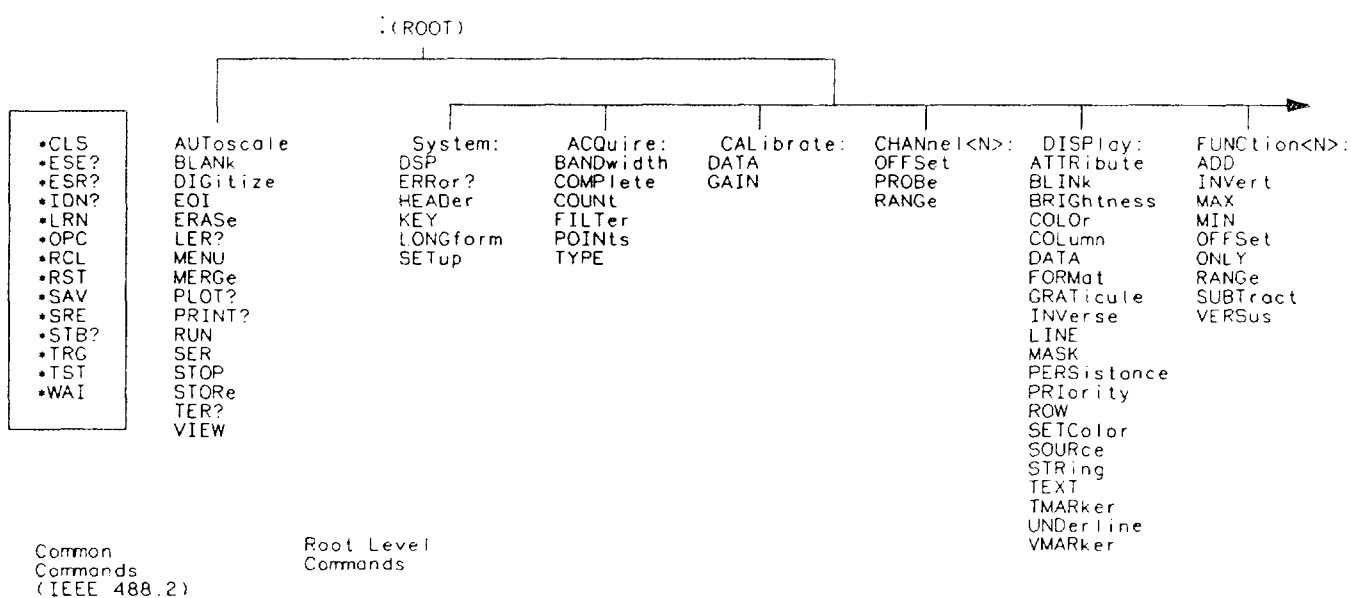
Example: **"\*RST"** .

**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.

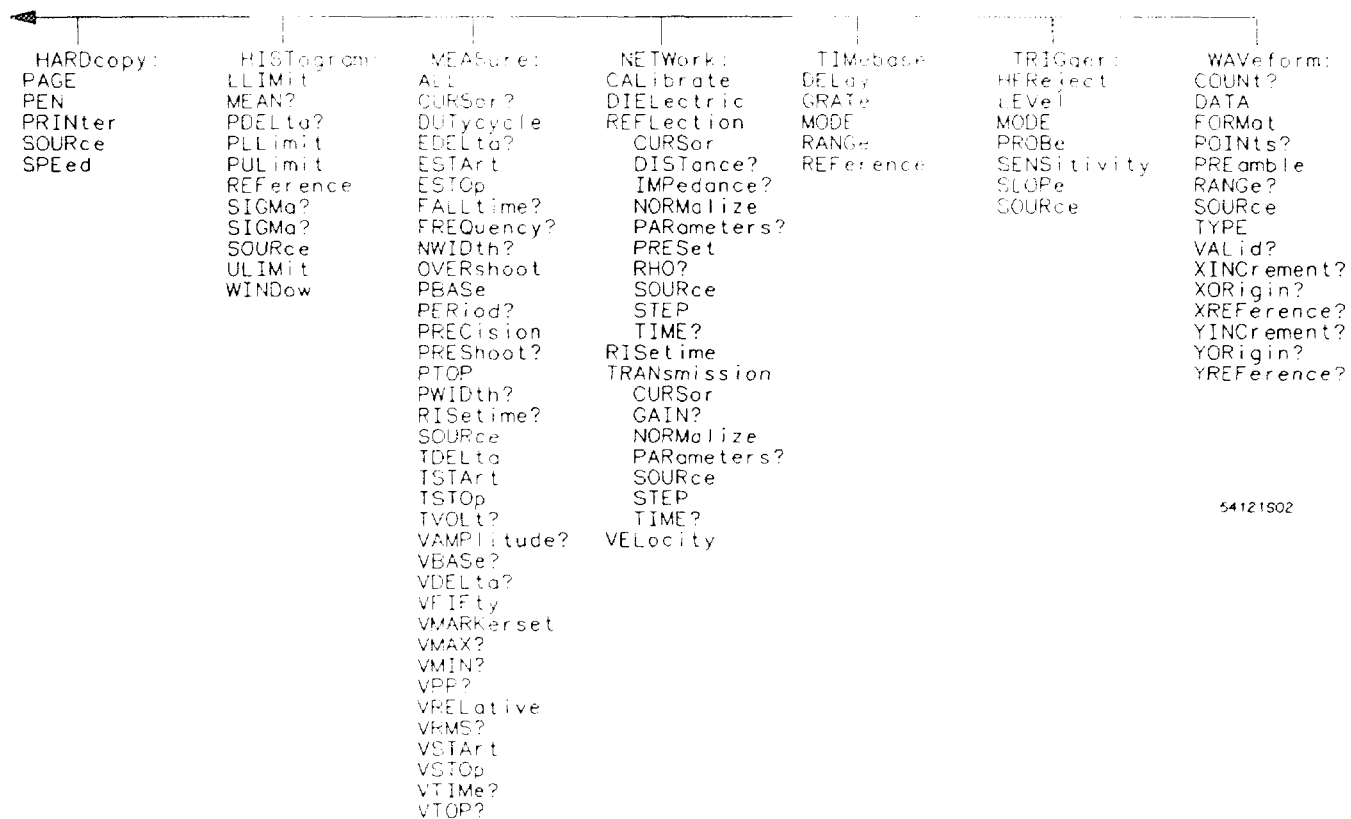




This instrument contains four identical channel subsystems and two identical function subsystems. The "N" in the Channel header must be 1 through 4, and the Function header must be 1 or 2.

54121S01

*Figure 4-1. The HP 54121T Command Tree*



54121502

Figure 4-1. HP 54121T Command Tree (Continued)

## 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 ":CHAN1: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 (CHAN1:), 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;"CHAN1:RANGE 0.5 ; OFFSET 0"
```

comments: The colon between CHAN1 and RANGE is necessary. CHAN1: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 CHAN1 preceding it, since the CHAN1:RANGE command set the parser to the CHAN1 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 the fourth line of 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 54121T 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 54121T will buffer responses to a query when it 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> ::` = Linefeed (ASCII decimal 10).

`<sp> ::` = `<white space>`.

---

## 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 54121T 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 and a programming command cross-reference is shown in table 4-2. Table 4-3 contains a summary of the commands with the required syntax to use them.

### 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 13 subsystems in the HP 54121T are:

|                  |  |
|------------------|--|
| <b>System</b>    | controls some basic functions of the oscilloscope.   |
| <b>Acquire</b>   | allows the parameters for acquiring and storing data to be set.  |
| <b>Calibrate</b> | allows the channel skews and vertical gain to be calibrated and stored for specific setups.  |
| <b>Channel</b>   | controls all Y-axis oscilloscope functions.  |
| <b>Display</b>   | controls how waveforms, voltage and time markers graticule, and text are displayed and written on the screen.                      |
| <b>Function</b>  | controls the waveform math functions of the oscilloscope.  |
| <b>Hardcopy</b>  | controls the parameters used during the plotting or printing of waveforms.   |
| <b>Histogram</b> | controls the parameters required for the oscilloscope to produce histograms.   |
| <b>Measure</b>   | selects the automatic measurements to be made.   |
| <b>Timebase</b>  | controls all X-axis oscilloscope functions.  |
| <b>Trigger</b>   | controls the trigger modes and parameters for each trigger mode.   |
| <b>Network</b>   | controls the time domain reflectometry and time domain transmission functions.   |
| <b>Waveform</b>  | 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 HP BASIC 4.0 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. If a plotter is used, it is always assumed to be at address 705.

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 (0.1), 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 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 54121T, 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 54121T is divided into 15 separate groups: Common commands, root level commands and 13 sets of subsystem commands. Each of the 15 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 of the commands contain a description of the command and its arguments the command syntax, and a programming example.

Table 4-2. Alphabetic Command Cross Reference

| Command     | Where Used          | Command     | Where Used          | Command       | Where Used          |
|-------------|---------------------|-------------|---------------------|---------------|---------------------|
| :ACQuire:   | Subsystem Selector  | :MEAN?      | Histogram Subsystem | SOURce        | Display Subsystem   |
| ADD         | Function Subsystem  | :MEASure:   | Subsystem Selector  | SOURce        | Hardcopy Subsystem  |
| ALL?        | Measure Subsystem   | :MENU       | Root Level Command  | SOURce        | Histogram Subsystem |
| ATTRibute   | Display Subsystem   | :MERGe      | Root Level Command  | SOURce        | Measure Subsystem   |
| :AUToscale  | Root Level Command  | MIN         | Function Subsystem  | SOURce        | Trigger Subsystem   |
|             |                     | MODE        | Timebase Subsystem  | SOURce        | Waveform Subsystem  |
|             |                     | MODE        | Trigger Subsystem   | SPEEd         | Hardcopy Subsystem  |
| BANDwidth   | Acquire Subsystem   |             |                     | *SRE          | Common Command      |
| :BLANK      | Root Level Command  | :NETWork:   | Subsystem Selector  | *STB?         | Common Command      |
| BLInk       | Display Subsystem   | NWIDth?     | Measure Subsystem   | :STOP         | Root Level Command  |
| BRIgHtness  | Display Subsystem   |             |                     | :STORe        | Root Level Command  |
|             |                     | OFFSet      | Channel Subsystem   | STRing        | Display Subsystem   |
| :CALibrate: | Subsystem Selector  | OFFSet      | Function Subsystem  | SUBTract      | Function Subsystem  |
| CALibrate   | Network Subsystem   | ONLY        | Function Subsystem  | :SYSTem:      | Subsystem Selector  |
| :CHANnel:   | Subsystem Selector  | *OPC        | Common Command      |               |                     |
| *CLS        | Common Command      | OVERshoot?  | Measure Subsystem   | TDElta?       | Measure Subsystem   |
| COLOR       | Display Subsystem   |             |                     | :TER          | Root Level Command  |
| COLUmN      | Display Subsystem   | PAGE        | Hardcopy Subsystem  | TEXT          | Display Subsystem   |
| COMPLetE    | Acquire Subsystem   | PBAsE       | Measure Subsystem   | :TiMEbase:    | Subsystem Selector  |
| COUNT       | Acquire Subsystem   | PDELta      | Histogram Subsystem | TMARker       | Display Subsystem   |
| COUNT       | Waveform Subsystem  | PEN         | Hardcopy Subsystem  | TRANsmission: | Network Subsystem   |
| CURSor?     | Measure Subsystem   | PERiod?     | Measure Subsystem   | CURSor        |                     |
|             |                     | PERSistence | Display Subsystem   | GAIN?         |                     |
| DATA        | Calibrate Subsystem | PLLimit     | Histogram Subsystem | NORMAlize     |                     |
| DATA        | Display Subsystem   | :PLOT       | Root Level Command  | PARAmeters?   |                     |
| DATA        | Waveform Subsystem  | POINTs      | Acquire Subsystem   | SOURce        |                     |
| DElay       | Timebase Subsystem  | POINTs?     | Waveform Subsystem  | STEP          |                     |
| DIElectric  | Network Subsystem   | PREAmble    | Waveform Subsystem  | TIME?         |                     |
| :DIGitize   | Root Level Command  | PRECision   | Measure Subsystem   | :TRIGger:     | Subsystem Selector  |
| :DISPlay:   | Subsystem Selector  | PREShoot?   | Measure Subsystem   | *TRG          | Common Command      |
| DSP         | System Subsystem    | :PRINt      | Root Level Command  | *TST?         | Common Command      |
| DUTYcycle?  | Measure Subsystem   | PRINter     | Hardcopy Subsystem  | TSTAr         | Measure Subsystem   |
|             |                     | PRiority    | Display Subsystem   | TSTOp         | Measure Subsystem   |
| EDELta?     | Measure Subsystem   | PROBe       | Channel Subsystem   | TVOLt?        | Measure Subsystem   |
| :EOI        | Root Level Command  | PROBe       | Trigger Subsystem   | TYPE          | Acquire Subsystem   |
| :ERASe      | Root Level Command  | PTOP        | Measure Subsystem   | TYPE          | Waveform Subsystem  |
| ERRor?      | System Subsystem    | PULimit     | Histogram Subsystem |               |                     |
| *ESE        | Common Command      | PWIDth?     | Measure Subsystem   | ULIMit        | Histogram Subsystem |
| *ESR?       | Common Command      |             |                     | UNDERline     | Display Subsystem   |
| ESTAr       | Measure Subsystem   | RANGE       | Channel Subsystem   |               |                     |
| ESTOp       | Measure Subsystem   | RANGE       | Function Subsystem  | VALId?        | Waveform Subsystem  |
|             |                     | RANGE       | Timebase Subsystem  | VAMPliTude?   | Measure Subsystem   |
| FALLtime?   | Measure Subsystem   | RANGE       | Waveform Subsystem  | VBAsE?        | Measure Subsystem   |
| FORMat      | Display Subsystem   | *RCL        | Common Command      | VDELta?       | Measure Subsystem   |
| FORMat      | Waveform Subsystem  | REFerence   | Histogram Subsystem | VELocity      | Network Subsystem   |
| FREQuency?  | Measure Subsystem   | REFerence   | Timebase Subsystem  | VERsus        | Function Subsystem  |
| :FUNCTion:  | Subsystem Selector  | REFlection: | Network Subsystem   | :VIEW         | Root Level Command  |
|             |                     | CURSor      |                     | VFIfty        | Measure Subsystem   |
| GAIN        | Calibrate Subsystem | DISTAnce?   |                     | VMARker       | Display Subsystem   |
| GRATe       | Timebase Subsystem  | IMPedance?  |                     | VMARkerset    | Measure Subsystem   |
| GRATicule   | Display Subsystem   | NORMAlize   |                     | VMAX?         | Measure Subsystem   |
|             |                     | PARAmeters? |                     | VMIN?         | Measure Subsystem   |
| :HARDcopy:  | Subsystem Selector  | PRESet      |                     | VPP?          | Measure Subsystem   |
| HEADer      | System Subsystem    | RHO?        |                     | VRELAtive     | Measure Subsystem   |
| HFRReject   | Trigger Subsystem   | SOURce      |                     | VRMS?         | Measure Subsystem   |
| :HISTogram: | Subsystem Selector  | STEP        |                     | VSTAr         | Measure Subsystem   |
|             |                     | TIME?       |                     | VSTOp         | Measure Subsystem   |
| *IDN?       | Common Command      | RISetime?   | Measure Subsystem   | VTIME?        | Measure Subsystem   |
| INVerse     | Display Subsystem   | RISetime    | Network Subsystem   | VTOP?         | Measure Subsystem   |
| INVer       | Function Subsystem  | ROW         | Display Subsystem   |               |                     |
|             |                     | *RST        | Common Command      | *WAI          | Common Command      |
| KEY         | System Subsystem    | :RUN        | Root Level Command  | :WAVEform:    | Subsystem Selector  |
|             |                     |             |                     | WINDow        | Histogram Subsystem |
| :LER?       | Root Level Command  | *SAV        | Common Command      |               |                     |
| LEVel       | Trigger Subsystem   | :SER        | Root Level Command  | XINCrement?   | Waveform Subsystem  |
| LINE        | Display Subsystem   | SENSitivity | Trigger Subsystem   | XORigin?      | Waveform Subsystem  |
| LJIMit      | Histogram Subsystem | SETColor    | Display Subsystem   | XREFerence?   | Waveform Subsystem  |
| LONGform    | System Subsystem    | SETup       | System Subsystem    |               |                     |
| *LRN?       | Common Command      | SIGMA?      | Histogram Subsystem | YINCrement?   | Waveform Subsystem  |
|             |                     | SLOPe       | Trigger Subsystem   | YORigin?      | Waveform Subsystem  |
| MASK        | Display Subsystem   |             |                     | YREFerence?   | Waveform Subsystem  |
| MAX         | Function Subsystem  |             |                     |               |                     |

Table 4-3. Command Summary

| KEYWORD                    | PARAMETER FORM  | TYPE    |
|----------------------------|---|---------|
| <b>COMMON COMMANDS</b>     |   |         |
| *CLS                       |   | command |
| *ESE <mask>                | <mask> ::= 0 through 255  | command |
| *ESE?                      | (integer - NR1) 0 through 255   | query   |
| *ESR?                      | (integer - NR1) 0 through 255   | query   |
| *IDN?                      | HEWLETT-PACKARD,54120A,<serial #>,<rev-date>                                | query   |
| *LRN?                      | block data in # format  | query   |
| *OPC                       |   | command |
| *OPC?                      | ASCII "1" in output queue   | query   |
| *RCL <recall_reg>          | <recall_reg> ::= 0 through 10   | command |
| *RST                       |   | command |
| *SAV <save_reg>            | <save_reg> ::= 0 through 9  | command |
| *SRE <mask>                | <mask> ::= 0 through 255  | command |
| *SRE?                      | (integer - NR1) 0 through 255   | query   |
| *STB?                      | (integer - NR1) 0 through 255   | query   |
| *TRG                       |   | command |
| *TST?                      | (integer - NR1) 0, -321, or -340  | query   |
| <b>ROOT LEVEL COMMANDS</b> |   |         |
| :AUTbscale                 |   | command |
| :BLANk <source>            | <source> ::= {CHANnel<N>   FUNction<N>   WMEMory<N>}<br><N> ::= 1 through 5 | command |
| :DIGitize <source>         | <source> ::= CHANnel<N>[,CHANnel<N>]<br><N> ::= 1 through 4                 | command |
| :EOI <state>               | <state> ::= {{ON   1 }   {OFF   0}}   | command |
| :EOI?                      | { 1   0 }   | query   |
| :ERASE <source>            | <source> ::= PMEMory { 0   1   2 }  | command |
| :LER?                      | (integer - NR1) { 1   0 }   | query   |

## ROOT LEVEL COMMANDS (Continued)

|                               |   |         |
|-------------------------------|---|---------|
| :MENU <menu>                  | <menu> ::= {CHANnel   DISPlay   HISTogram   MEASure  <br>MATH   NETWork   PLOT   PRINt   SAVE   TDELta  <br>TIMEbase   TRIGger   UTILity   VDELta } | command |
| :MENU?                        | {CHANnel   DISPlay   HISTogram   MEASure   MATH  <br>NETWork   PLOT   PRINt   SAVE   TDELta  <br>TIMEbase   TRIGger   UTILity   VDELta }            | query   |
| :MERGe <source>               | <source> ::= {PMEMory1   PMEMory2}  | command |
| :PLOT?                        | initiates plot of display   | command |
| :PRINt?                       | initiates print of display  | command |
| :RUN                          |   | command |
| :SER <string>                 | <string> ::= serial number in string format   | command |
| :STOP                         |   | command |
| :STORE <source>,<destination> | <source> ::= {CHANnel{1   ...   4}  <br>FUNction{1   2}  <br>WMEMory{1   ...   4}}<br><destination> ::= WMEMory{1   ...   4}                        | command |
| :TER?                         | {1   0}   | query   |
| :VIEW <source>                | <source> ::= {CHANnel<N>   FUNction<N>   WMEMory<N>}<br><N> ::= 1 through 5   | command |

## SYSTEM SUBSYSTEM

|                  |  |         |
|------------------|--|---------|
| :SYSTem:         |  |         |
| DSP <string>     | <string> ::= quoted string             | command |
| DSP?             | last string displayed on advisory line | query   |
| ERRor?           | (integer - NR1) next error # in queue  | query   |
| HEADer <state>   | <state> ::= {{ON   1}   {OFF   0}}     | command |
| HEADer?          | { 1   0 }                              | query   |
| KEY <code>       | <code> ::= KEYCODE 1 - 63              | command |
| KEY?             | (integer - NR1) 0 - 63                 | query   |
| LONGform <state> | <state> ::= {{ON   1}   {OFF   0}}     | command |
| LONGform?        | { 1   0 }                              | query   |

## SYSTEM SUBSYSTEM (Continued)

|               |                                    |         |
|---------------|------------------------------------|---------|
| SETup <block> | <block> ::= block data in # format | command |
| SETup?        | block data in # format             | query   |

## ACQUIRE SUBSYSTEM

|                   |  |         |
|-------------------|--|---------|
| <b>:ACquire:</b>  |  |         |
| BANDwidth <state> | <state> ::= {HIGH   LOW}                               | command |
| BANDwidth?        | {HIGH   LOW}   | query   |
| COMplete?         | (integer - NR1 format) 100                             | query   |
| COUNt <argument>  | <argument> ::= 1 through 655000000                     | command |
|                   | valid values depend on :ACQ:TYPE                       |         |
| COUNt?            | (integer - NR1) count value                            | query   |
| POINts <argument> | <argument> ::= 100 - 1024 valid values depend on s/div | command |
| POINts?           | (integer - NR1 format) point value                     | query   |
| TYPE <type>       | <type> ::= {NORMAL   AVERage   ENVELOpe   HISTogram}   | command |
| TYPE?             | {NORMAL   AVERage   ENVELOpe   HISTogram}              | query   |

## CALIBRATE SUBSYSTEM

|                    |                                    |         |
|--------------------|------------------------------------|---------|
| <b>:CALibrate:</b> |                                    |         |
| DATA <block>       | <block> ::= block data in # format | command |
| DATA?              | block data in # format             | query   |
| GAIN               |                                    | command |

## CHANNEL SUBSYSTEM

|                           |  |         |
|---------------------------|--|---------|
| <b>:CHANnel&lt;N&gt;:</b> |  |         |
| <N>                       | <N> ::= {1   2   3   4}                              |         |
| OFFSet <voltage>          | <voltage> ::= - 500 mV to +500 mV                    | command |
| OFFSet                    | (exponential - NR3) offset value                     | query   |
| PROBe <atten>             | <atten> ::= 1 through 1000                           | command |
| PROBe?                    | (exponential - NR3) probe attenuation                | query   |
| RANGe <voltage>           | <voltage> ::= .008 V through .640 V - with 1:1 probe | command |
| RANGe?                    | (exponential - NR3) current range                    | query   |

## DISPLAY SUBSYSTEM

|  |   |                  |
|--|---|------------------|
| <b>:DISPlay:</b>                                 |   |                  |
| ATTRibute <state><br>ATTRibute?                  | <state> ::= {ENABLE   DISable}<br>{ENABLE   DISable}                                  | command<br>query |
| BLINk <state><br>BLINk?                          | <state> ::= {{ON   1}   {OFF   0}}<br>{ 1   0 }                                       | command<br>query |
| BRIGhtness <state><br>BRIGhtness?                | <state> ::= {LOW   HIGH}<br>{LOW   HIGH}  | command<br>query |
| COLOr <argument><br>COLOr?                       | <argument> ::= 0 through 15<br>(integer - NR1) current color                          | command<br>query |
| COLumn <argument><br>COLumn?                     | <argument> ::= 0 through 71<br>(integer - NR1) current column                         | command<br>query |
| DATA <block><br>DATA?                            | <block> ::= block data in # format<br>block data in # format                          | command<br>query |
| FORMat <argument><br>FORMat                      | <argument> ::= { 1   2   3 }<br>(integer - NR1) current format                        | command<br>query |
| GRATicule <mode><br>GRATicule?                   | <mode> ::= OFF   GRID   AXES   FRAME<br>{OFF   GRID   AXIS  FRAME}                    | command<br>query |
| INVerse <state><br>INVerse                       | <state> ::= {{ON   1}   {OFF   0}}<br>{ 1   0 }                                       | command<br>query |
| LINE <string>                                    | <string> ::= quoted string  | command          |
| MASK <mask><br>MASK?                             | <mask> ::= 0 through 255<br>(integer - NR1) current mask                              | command<br>query |
| PERSistence <argument><br>PERSistence?           | <argument> ::= {INFinite   0.3 through 11}<br>(exponential - NR3) current persistence | command<br>query |
| PRlority <state><br>PRlority?                    | <state> ::= {{ON   1}   {OFF   0}}<br>{ 1   0 }                                       | command<br>query |
| ROW <argument><br>ROW?                           | <argument> ::= 0 through 21<br>(integer - NR1) current row                            | command<br>query |
| :SETColor {<color>,<hue>,<sat>,<lum>   DEFault } | <color> 1 through 15<br><hue>,<sat>,<lum> ::= 0 through 100                           | command          |

## DISPLAY SUBSYSTEM (Continued)

|                                  |  |                  |
|----------------------------------|--|------------------|
| :SETColor <N>?                   | <N> ::= color number 0 through 15<br>(3 integers - NR3) current value of hue, saturation, luminosity | query            |
| :SOURce:PMEMory <N><br>:SOURce?  | <N> ::= 0 through 8<br>(integer - NR3) current source  | command<br>query |
| :STRing <string>                 | <string> ::= quoted string   | command          |
| :TEXT BLANK                      |  | command          |
| :TMArker <state><br>:TMArker?    | <state> ::= {{ON   1 }   {OFF   0 }}<br>{ 0   1 }  | command<br>query |
| :UNDerline<state><br>:UNDerline? | <state> ::= {{ON   1 }   {OFF   0 }}<br>{ 1   0 }  | command<br>query |
| :VMArker <state><br>:VMArker?    | <state> ::= {{ON   1 }   {OFF   0 }}<br>{ 1   0 }  | command<br>query |

## FUNCTION SUBSYSTEM

<N> ::= 1 or 2  
<operand> ::= {CHANnel{1 ... 4} | WMEMory{1 ... 4}}

|   |  |                  |
|---|--|------------------|
| :FUNctioN<N>:<br>ADD <operand>, <operand> |  | command          |
| INVert <operand>                          |  | command          |
| MAX <operand>                             |  | command          |
| MIN <operand>                             |  | command          |
| OFFSet <volts><br>OFFSet?                 | + or - volts full screen maximum<br>(exponential - NR3) current offset     | command<br>query |
| ONLY <operand>                            |  | command          |
| RANGe <volts><br>RANGe?                   | 8 mV to 640 mV maximum with 1:1 probe<br>(exponential - NR3) current range | command<br>query |
| SUBTract<operand>,<operand>               |  | command          |
| VERSus <operand>,<operand>                |  | command          |



## HARDCOPY SUBSYSTEM

### :HARDcopy:

|                                |   |                  |
|--------------------------------|---|------------------|
| PAGE <mode><br>PAGE?           | <mode> ::= {MANual   AUTomatic}<br>{MANual   AUTomatic} current setting | command<br>query |
| PEN <mode><br>PEN?             | <mode> ::= {MANual   AUTomatic}<br>{MANual   AUTomatic} current setting | command<br>query |
| PRINter <argument><br>PRINter? | <argument> ::= {DEFault   OTHer}<br>{DEFault   OTHer} current setting   | command<br>query |
| SOURce <source>[,<source>,...] | <source> ::= {PMEMory{0 ... 2}   FACTors WMEMory{0 ... 4}}              | command          |
| SPEed <mode><br>SPEed?         | <mode> ::= {SLOW   FAST}<br>{SLOW   FAST} current setting               | command<br>query |

## HISTOGRAM SUBSYSTEM

### :HISTogram:

|   |   |                  |
|---|---|------------------|
| LLIMit <argument><br>LLIMit?                  | time or voltage within screen limits<br>(exponential - NR3) current position                | command<br>query |
| MEAN?   | statistical mean time or voltage<br>(exponential - NR3)                                     | query            |
| PDELta?                                       | difference in cumulative percentage between upper and lower<br>limit<br>(exponential - NR3) | query            |
| PLLimit <argument><br>PLLimit?                | cummulative percentage value<br>(exponential - NR3) current value                           | command<br>query |
| PULimit <argument><br>PULimit?                | cummulative percentage value<br>(exponential - NR3) current value                           | command<br>query |
| REFeRence <0% reference>,<br><100% reference> | <0% reference> and <100% reference> ::= any on<br>screen time or voltage                    | command          |
| REFeRence?                                    | (exponential - NR3) current 0% reference,100% reference                                     | query            |
| SIGMa?  | (exponential - NR3) standard deviation on the histogram                                     | command          |
| SOURce <source><br>:SOURce?                   | <source> ::= CHANnel{1   2   3   4}<br>CHANnel{1   2   3   4}                               | command<br>query |
| ULIMit <argument><br>ULIMit?                  | time or voltage within screen limits<br>(exponential - NR3) current position                | command<br>query |

## HISTOGRAM SUBSYSTEM (Continued)

|                                    |   |         |
|------------------------------------|---|---------|
| WINDow <sec-or-volt>,<sec-or-volt> | <sec-or-volt> ::= any on screen time or voltage                   | command |
| WINDow?                            | {TIME   VOLTs},<value>,<value><br><value> ::= (exponential - NR3) | query   |

## MEASURE SUBSYSTEM

|   |  |         |
|---|--|---------|
| MEASure:  |  |         |
| ALL?  | (16 exponential values - NR3)  | query   |
| CURSor? <cursor>                                  | <cursor> ::= {DELTA   START   STOP}<br>RESPONSE: <time>,<voltage> current values<br><time> and <voltage> ::= (exponential - NR3) | query   |
| DUTycle?  | (exponential - NR3) dutycycle ratio  | query   |
| EDELta? <slope & occurrence>,<slope & occurrence> | <slope & occurrence> ::= sign and edge number<br>RESPONSE: (exponential - NR3) difference value                                  | query   |
| ESTArt <edge>                                     | <edge> ::= sign and number   | command |
| ESTArt?   | (exponential - NR3) current edge   | query   |
| ESTOp <edge>                                      | <edge> ::= sign and number   | command |
| ESTOp?  | (exponential - NR3) current edge   | query   |
| FALLtime?   | (exponential - NR3) falltime value   | query   |
| FREQuency?  | (exponential - NR3) frequency value  | query   |
| NWIDth?   | (exponential - NR3) negative pulse width   | query   |
| OVERshoot?  | (exponential - NR3) overshoot ratio  | query   |
| PBAsE <percent>                                   | - 25% through 125%   | command |
| PBAsE?  | (integer - NR1) current setting  | query   |
| PERIOD?   | (exponential - NR3) period value   | query   |
| PRECision <mode>                                  | {COARse   FINE}  | command |
| PRECision?  | {COARse   Fine} current setting  | query   |
| PREShoot?   | (exponential - NR3) preshoot ratio   | query   |
| PTOP <percent>                                    | - 25% through 125%   | command |
| PTOP?   | (integer - NR1) current setting  | query   |
| PWIDth?   | (exponential - NR3) positive pulse width   | query   |

## MEASURE SUBSYSTEM (Continued)

|                                     |   |         |
|-------------------------------------|---|---------|
| RISetime?                           | (exponential - NR3) risetime value  | query   |
| SOURce <source>[,<source>]          | <source> ::= {CHANnel{1 ... 4}  <br>FUNCTION{1   2}  <br>WMEMory{1 ... 4}}  | command |
| SOURce?                             | <source>[,<source>] current source(s)   | query   |
| TDELta?                             | (exponential - NR3) current Delta time  | query   |
| TSTArt <time>                       | <time> ::= time at start marker   | command |
| TSTArt?                             | (exponential - NR3) start marker time   | query   |
| TSTOp <time>                        | <time> ::= time at stop marker  | command |
| TSTOp?                              | (exponential - NR3) stop marker time  | query   |
| TVOLt <voltage>,<slope><occurrence> | <voltage> ::= voltage level at crossing<br><slope> ::= +(pos) or -(neg) edge<br><occurrence> ::= number of crossing | command |
| TVOLt?                              | (exponential - NR3) time at crossing  | query   |
| VAMPlitude?                         | (exponential - NR3) top-base voltage  | query   |
| VBASe?                              | (exponential - NR3) base voltage  | query   |
| VDELta?                             | (exponential - NR3) Delta Voltage   | query   |
| VFIFty                              | (sets markers to 50% voltage of source)   | command |
| VMARKerset                          | (sets markers to VREL positions)  | command |
| VMAX?                               | (exponential - NR3) maximum voltage   | query   |
| VMIN?                               | (exponential - NR3) minimum voltage   | query   |
| VPP?                                | (exponential - NR3) peak to peak voltage  | query   |
| VRELative <percent>                 | <percent> ::= {0   10   20   50}  | command |
| VRELative?                          | (integer - NR1) current position  | query   |
| VRMS?                               | (exponential - NR3) rms voltage   | query   |
| VSTArt <voltage>                    | <voltage> ::= position of the cursor in volts   | command |
| VSTArt?                             | (exponential - NR3) voltage at VMarker1   | query   |
| VSTOp <voltage>                     | <voltage> ::= position of the cursor in volts   | command |
| VSTOp?                              | (exponential - NR3) voltage at VMarker2   | query   |

|        |   |       |
|--------|---|-------|
| VTIME? | (exponential - NR3) voltage at specified time | query |
| VTOP?  | (exponential - NR3) voltage at waveform top   | query |

## NETWORK SUBSYSTEM

### :NETWork:

|                       |  |                  |
|-----------------------|--|------------------|
| CALibrate <argument>  | {SHORT   FIFTy   OPEN   THRU}  | command          |
| CALibrate             | {SHORT   FIFTy   OPEN   THRU}  | command          |
| DIElectric <constant> | <constant> ::= 1.0 through 100.0   | command          |
| DIElectric?           | (exponential - NR3) dielectric constant  | query            |
| REFlection:           |  |                  |
| CURSor <deltat>       | <deltat> ::= any on screen time  | command          |
| CURSor?               | (exponential - NR3) current position   | query            |
| DISTance?             | (exponential - NR3) distance in metres   | query            |
| IMPedance?            | (exponential - NR3) impedance at cursor  | query            |
| NORMALize             |  | command          |
| PARameters?           | (exponential - NR3) <rho__max>,<rho__min>  | query            |
| PRESet                |  | command          |
| RHO?                  | (exponential - NR3) percent reflection   | query            |
| SOURce <source>       | {CHANnel1   CHANnel3   WMEMory1}   | command          |
| SOURce?               | current source   | query            |
| STEP <channel>        | {CHANnel1   CHANnel3   OFF}  | command          |
| STEP?                 | currently selected step channel  | query            |
| TIME?                 | (exponential - NR3) time from trigger<br>minimum = timebase range/125 or 10 ps<br>maximum = timebase range/2 | query<br>command |
| RISetime <risetime>   |  |                  |
| RISetime?             | (exponential - NR3) filter risetime  | query            |
| TRANsmission:         |  |                  |
| CURSor <deltat>       | cursor position in time  | command          |
| CURSor?               | (exponential - NR3) current position   | query            |
| GAIN?                 | (exponential - NR3) gain at cursor   | query            |
| NORMALize             |  | command          |

## NETWORK SUBSYSTEM (Continued)

|                     |  |         |
|---------------------|--|---------|
| PARAmeters?         | (exponential - NR3) <prop_delay>,<dist>,<gain> | query   |
| SOURce <source>     | {CHANnel4   WMEMory2}                          | command |
| SOURce?             | {CHANnel4   WMEMory2} current source           | query   |
| STEP <source>       | {CHANnel4   OFF}                               | command |
| STEP?               | {CHANnel4   OFF} current step source           | query   |
| VELocity <constant> | 3.3356 ns/m to 33.356 ns/m                     | command |
| VELocity            | (exponential - NR3) current velocity constant  | query   |

## TIMEBASE SUBSYSTEM

|                      |                                       |         |
|----------------------|---------------------------------------|---------|
| <b>TIMEbase:</b>     |                                       |         |
| DELAy <delay>        | delay time in seconds (16 ns minimum) | command |
| DELAy?               | (exponential - NR3) time in seconds   | query   |
| GRATe <frequency>    | Rate Generator rate in hertz          | command |
| GRATe?               | (exponential - NR3) current rate      | query   |
| MODE <mode>          | {FREerun   TRIGgered}                 | command |
| MODE?                | {FREerun   TRIGgered} current mode    | query   |
| RANGE <range>        | 100 ps to 10 s                        | command |
| RANGE?               | (exponential - NR3) current range     | query   |
| REFerence <position> | {LEFT   CENTer}                       | command |
| REFerence?           | {LEFT   CENTer} current reference     | query   |

## TRIGGER SUBSYSTEM

|                  |  |         |
|------------------|--|---------|
| <b>TRIGger:</b>  |  |         |
| HFRejeCt <state> | {{ON   1}   {OFF   0}}                       | command |
| HFRejeCt?        | { 1   0 } current state                      | query   |
| LEVel <voltage>  | <voltage> ::= trigger level setting in volts | command |
| LEVel?           | (exponential - NR3) current setting          | query   |
| MODE EDGE        | ALWAYS SET TO EDGE                           | command |
| MODE?            | ALWAYS RETURNS EDGE                          | query   |
| PROBe <atten>    | 1 through 1000                               | command |
| PROBe?           | (exponential - NR3) probe attenuation factor | query   |

## TRIGGER SUBSYSTEM (Continued)

|                                    |  |                  |
|------------------------------------|--|------------------|
| SENSitivity <mode><br>SENSitivity? | {NORMal   HIGH}<br>{NORMal   HIGH} current mode                      | command<br>query |
| SLOPe <slope><br>SLOPe?            | {POSitive   NEGative}<br>{POSitive   NEGative} current trigger slope | command<br>query |
| SOURce EXTernal1<br>SOURce?        | ALWAYS SET TO EXTERNAL1<br>ALWAYS RETURNS EXTERNAL1                  | command<br>query |

## WAVEFORM SUBSYSTEM

|                               |  |                  |
|-------------------------------|--|------------------|
| <b>:WAVEform:</b><br>COUNT?   | (integer - NR1) current count value                                      | query            |
| DATA <block><br>DATA?         | block data in # format<br>block data in # format                         | command<br>query |
| FORMat <mode><br>FORMat?      | <mode> ::= {ASCii   WORD   LONG}<br>{ASCii   WORD   LONG} current format | command<br>query |
| POINTs?                       | (integer - NR1) number of acquired data points                           | query            |
| PREamble <block><br>PREamble? | block data in # format<br>block data in # format                         | command<br>query |
| SOURce <source><br>SOURce?    | WMEMory{1   2   3   4   5}<br>current source WMEMory{1 ... 5}            | command<br>query |
| TYPE <type><br>TYPE?          | {NORMal   AVERage   ENVELOpe   HISTogram}<br>current type (as above)     | command<br>query |
| VALid?                        | {1   0} current data valid   | query            |
| XINCrement?                   | (exponential - NR3) current value  | query            |
| XORigin?                      | (exponential - NR3) current value  | query            |
| XREFerence?                   | (integer - NR1) always 0   | query            |
| YINCrement?                   | (exponential - NR3) current value  | query            |
| YORigin?                      | (exponential - NR3) current value  | query            |
| YREFerence?                   | (integer - NR1) current value  | query            |

## Introduction

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

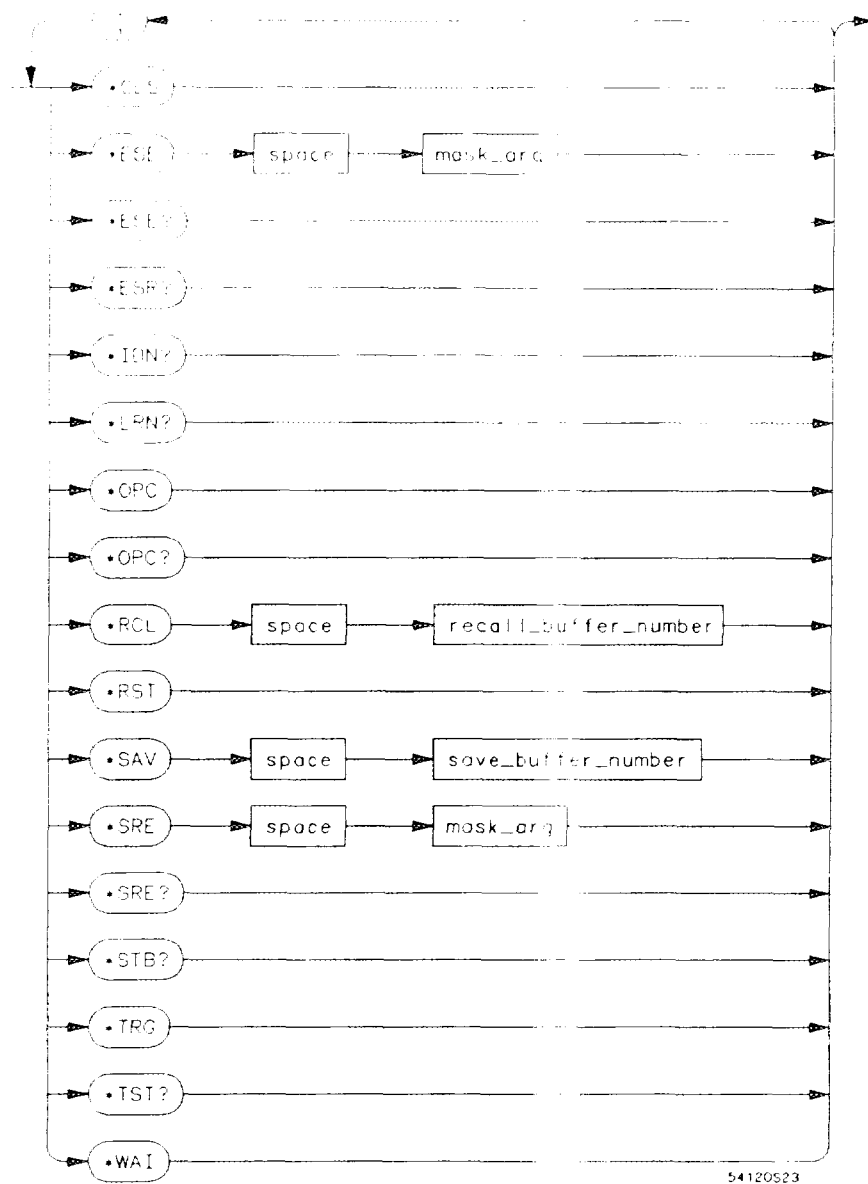
The common commands control some of the basic instrument functions, such as instrument identification and reset, reading the learn string (instrument setup), how status is read and cleared, and how commands and queries are received and processed by the instrument.

Common commands can be received and processed by the HP 54121T 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 chapter 3 for a complete discussion of how to read the status registers and how to use the status information available from this instrument.

Each of the status registers has a parallel status enable (mask) register. By setting the bits in the mask value you can select the status information you wish to use. All status bits that have not been unmasked (enabled in the enable register) will return to a value of zero when that status register is read.

Refer to figure 5-1 for the 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.

**recall\_buffer\_number** = An integer, 0 through 10.

**save\_buffer\_number** = An integer, 0 through 9.

*Figure 5-1. Common Commands Syntax Diagram*



---

**\*CLS****(Clear Status)****command**

The \*CLS (clear status) common command clears the Event Status Register, the Status Byte Register, the trigger bit, the local bit, and the error queue.

The Event Status Register is read by the \*ESR? command. The Status Byte Register is read by the \*STB? command or a serial poll.

**Command Syntax:** \*CLS

**Example:** OUTPUT 707;"\*CLS"<NL>

**Note**

*Refer to chapter 3 for a complete discussion of status.*

## **\*ESE**

---

|             |                              |                      |
|-------------|------------------------------|----------------------|
| <b>*ESE</b> | <b>(Event Status Enable)</b> | <b>command/query</b> |
|-------------|------------------------------|----------------------|

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 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>  
                          <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, event summary bit (ESB) in the Status Byte Register will be set also.

*Table 5-1. Standard Event Status Enable Register*

| Event Status Enable Register<br>(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     |

**Query Syntax:** \*ESE?

**Returned Format:** <mask> <NL>

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

**Note**

*Refer to chapter 3 for a complete discussion of status.*

## **\*ESR?**

---

### **\*ESR?**

**(Event Status Register)**

**query**

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

#### **Note**

*Reading the register clears the Standard Event Status Register and the ESB bit in the SBR.*

**Query Syntax:** \*ESR?

**Returned Format:** <status> <NL>  
<status> :: = 0 to 255

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

With the example (\*ESE = 64), if a front-panel key has been pressed the variable "event" will contain 64, the URQ (User Request bit).

Table 5-2 shows the Standard Event Status Register. The table shows each bit in the Standard Event Status Register, and the bit weight. When you read Standard 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. The Standard Event Status Register.*

| BIT | BIT WEIGHT | BIT NAME | CONDITION  |
|-----|------------|----------|--|
| 7   | 128        | PON      | 0 = not used - always zero.  |
| 6   | 64         | URQ      | 0 = no front-panel key has been pressed<br>1 = front-panel key has been pressed  |
| 5   | 32         | CME      | 0 = no command errors<br>1 = a command error has been detected                   |
| 4   | 16         | EXE      | 0 = no execution errors<br>1 = an execution error has been detected              |
| 3   | 8          | DDE      | 0 = no device dependent errors<br>1 = a device dependent error has been detected |
| 2   | 4          | QYE      | 0 = no query errors<br>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<br>1 = operation is complete                       |
|     |            |          | 0 = False = Low<br>1 = True = High   |

## **\*IDN?**

---

|              |                                |              |
|--------------|--------------------------------|--------------|
| <b>*IDN?</b> | <b>(Identification Number)</b> | <b>query</b> |
|--------------|--------------------------------|--------------|

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

"HEWLETT-PACKARD,54121A,<XXXXAYYYYY>,<ZZZZ>."

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

<ZZZZ> :: = the software revision code.

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

**Query Syntax:** \*IDN?

**Returned Format:** HEWLETT-PACKARD,54121A,XXXXAYYYYY,ZZZZ < NL >

**Example:** DIM Id\$(50)  
OUTPUT 707;"\*IDN?"  
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 does the same thing 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.*

**Query Syntax:** \*LRN?

**Returned Format:** :SYSTem:SETup <block\_data>  
<block\_data> :: = # 3272 <learn string> <NL>

The learn string is 272 data bytes in length. The 272 data bytes do not include the header or " # 3272", therefore those bytes must also be read.

**Example:** DIM Lrn\$(300)  
OUTPUT 707;"\*LRN?"  
ENTER 707 USING " — K";Lrn\$

## **\*OPC**

---

| <b>*OPC</b> | <b>(Operation Complete)</b> | <b>command/query</b> |
|-------------|-----------------------------|----------------------|
|-------------|-----------------------------|----------------------|

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;"\*OPC?"  
ENTER 707;Opc  
PRINT Opc



---

**\*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 0 through 9 are general purpose and can be used with the \*SAV command. Register 10 is special, it recalls the state that existed before the last AUTOSCALE, entry to the HISTOGRAM menu, or a Channel Skew calibration.

**Command Syntax:**    \*RCL <rcl\_register>  
                          <rcl\_register> :: = 0 through 10

**Example:**    OUTPUT 707;"\*RCL 3"

## \*RST

### \*RST

(Reset)

command

The \*RST command places the instrument in a known state. Refer to tables 5-3 and 5-4 for the reset conditions.

**Command Syntax:** \*RST

**Example:** OUTPUT 707;"\*RST"

*Table 5-3. Reset Conditions for the HP 54121T*

|                         |            |
|-------------------------|------------|
| Channel 1,2,3,4Display  | On         |
| Channel 1,2,3,4Range    | 640 mv/div |
| Channel 1,2,3,4OFFSET   | 0.00 V     |
| Channel 1,2,3,4Probe    | 1.0        |
| Timebase Range          | 10 ns/div  |
| DELAY                   | 16.00 ns   |
| Delay Ref at            | Left       |
| Freerun/Triggered Sweep | Trg'd      |
| Freerun Rate            | 500 KHz    |
| TRIGGER LEVEL           | 0.00 volts |
| Trigger Slope           | Positive   |
| Trigger Probe           | 1.0        |
| Averaging               | On         |
| NUMBER OF AVERAGES      | 1          |
| Screen                  | Quad       |
| Graticule               | Axes       |
| Voltage Markers         | Off        |
| V Marker 1 Position     | 0 mV       |
| V Marker 2 Position     | 0 mV       |
| Topbase Reference       | 50 - 50%   |
| Variable Levels         | 0 - 100%   |

Table 5-3. Reset Conditions for the HP 54121T (continued)

|                               |                   |
|-------------------------------|-------------------|
| Time Markers                  | Off               |
| Start Marker Position         | 18.5 ns           |
| Stop Marker Position          | 23.5 ns           |
| Start Marker Edge Slope       | Positive          |
| Start Marker Number           | 1                 |
| Stop Marker Edge Slope        | Negative          |
| Stop Marker Number            | 1                 |
| Waveform Memories             | Off               |
| Source for Store              | Channel 1         |
| Selected Memory               | Waveform Memory 1 |
| Waveform Data in all memories | 0 volts           |
| Pixel Memories                | Off               |
| Function 1 and 2              | Off               |
| Functions Definition          | Chan 1 - Chan 2   |
| Functions Scaling             |                   |
| Offset                        | 0.0 V             |
| Volts/division                | 80.00 mV/div      |
| Measure Source                | Channel 1         |
| Precision                     | Fine              |

Table 5-4. HP-IB Reset Conditions for the HP 54121T

|                         |           |
|-------------------------|-----------|
| Service Request Enable  | Decimal 0 |
| Serial Poll Status Byte | Clear     |
| Error Queue             | Empty     |
| WAVEform Format         | Word      |
| EOI                     | On        |
| Longform                | Off       |
| Header                  | Off       |

## **\*SAV**

---

|             |               |                |
|-------------|---------------|----------------|
| <b>*SAV</b> | <b>(Save)</b> | <b>command</b> |
|-------------|---------------|----------------|

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 0 through 9 are valid for this command.

**Command Syntax:**    \*SAV < save\_register >  
                          < save\_register > :: = 0 through 9

**Example:**    OUTPUT 707;"\*SAV 6"

---

**\*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>  
                          <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>  
                          <mask> :: = sum of all bits that are set - 0 through 255

**Example:**    OUTPUT 707;"\*SRE?"  
                  ENTER 707;Sre  
                  PRINT Sre

## \*SRE

---

*Table 5-5. Service Request Enable Register*

| Service Request Enable Register |        |                             |
|---------------------------------|--------|-----------------------------|
| Bit                             | Weight | Enables                     |
| 7                               | 128    | not used                    |
| 6                               | 64     | RQS - Request Service       |
| 5                               | 32     | ESR - Event Status Register |
| 4                               | 16     | MAV - Message Available     |
| 3                               | 8      | SDS - Sub-Device Status     |
| 2                               | 4      | MSG - Message               |
| 1                               | 2      | LCL - Local                 |
| 0                               | 1      | TRG - Trigger               |

**\*STB?**

**\*STB?**

**(Status Byte)**

**query**

The \*STB query returns the current value of the instrument's status byte. The MSS (Master Summary Status) bit and not RQS is reported on bit 6. 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>  
<value> :: = 0 through 255 (integer - NR1)

**Example:** OUTPUT 707;"\*STB?"  
ENTER 707;Stb  
PRINT Stb

## \*STB?

Table 5-6. The Status Byte Register.

| BIT | BIT<br>WEIGHT | BIT<br>NAME | CONDITION  |
|-----|---------------|-------------|--|
| 7   | 128           | ---         | 0 = not used   |
| 6   | 64            | RQS/M<br>SS | 0 = instrument has no reason for service<br>1 = instrument is requesting service                   |
| 5   | 32            | ESR         | 0 = no event status conditions have occurred<br>1 = an enabled event status condition has occurred |
| 4   | 16            | MAV         | 0 = no output messages are ready<br>1 = an output message is ready                                 |
| 3   | 8             | SDS         | 0 = not used - always 0  |
| 2   | 4             | MSG         | 0 = no message has been displayed<br>1 = message has been displayed                                |
| 1   | 2             | LCL         | 0 = a remote-to-local transition has not occurred<br>1 = a remote-to-local transition has occurred |
| 0   | 1             | TRG         | 0 = no trigger has occurred<br>1 = a trigger has occurred  |

0 = False = Low  
1 = True = High



---

**\*TRG****(Trigger)****command**

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

**Command Syntax:** \*TRG

**Example:** OUTPUT 707;"\*TRG"

## \*TST?

---

### \*TST?

(Test)

query

The function of the \*TST? command is to run and report the results of the tests in the test menu. The test menu setup actually controls how the \*TST? command is implemented. If the test menu is set to REPEAT LOOP the selected loop will be tested once and the results reported. If the test menu is in the RUN FROM LOOP mode, the \*TST? command will begin its test at the currently selected loop increment the loop number by one, test that loop, and repeat this sequence until the highest loop number has been tested. The results are then reported via the HP-IB.

Three values can be reported as a result of the \*TST? command. They are:

- 321 if a ROM test failed or loop 1 failed
- 340 for any other failure; test or loop
- 0 if no failure occurred

#### Note

*The Test menu (within the Utility menu) must be set up prior to sending the \*TST? command. The menu can be set up from the front panel or by using the :SYSTEM:KEY commands.*

**Query Syntax:** \*TST?

**Returned Format:** <result> <NL>  
<result> :: =    – 321 (ROM test failure)  
                  – 340 (other failure)  
                  0 (successful completion of test)

**Example:** OUTPUT 707;"\*TST?"  
              ENTER 707;Tst  
              PRINT Tst

---

**\*WAI****(Wait)****command**

The \*WAI command causes the device to wait until the completion of one command before executing any further command or queries. At present this command has no effect since all commands defined for this instrument will be completed before the next command is started.

**Command Syntax:** \*WAI

**Example:** OUTPUT 707;"\*WAI"



# Root Level Commands

6

## Introduction

Root Level commands control the basic operations of the oscilloscope.

Refer to figure 6-1 for root level commands syntax diagram.

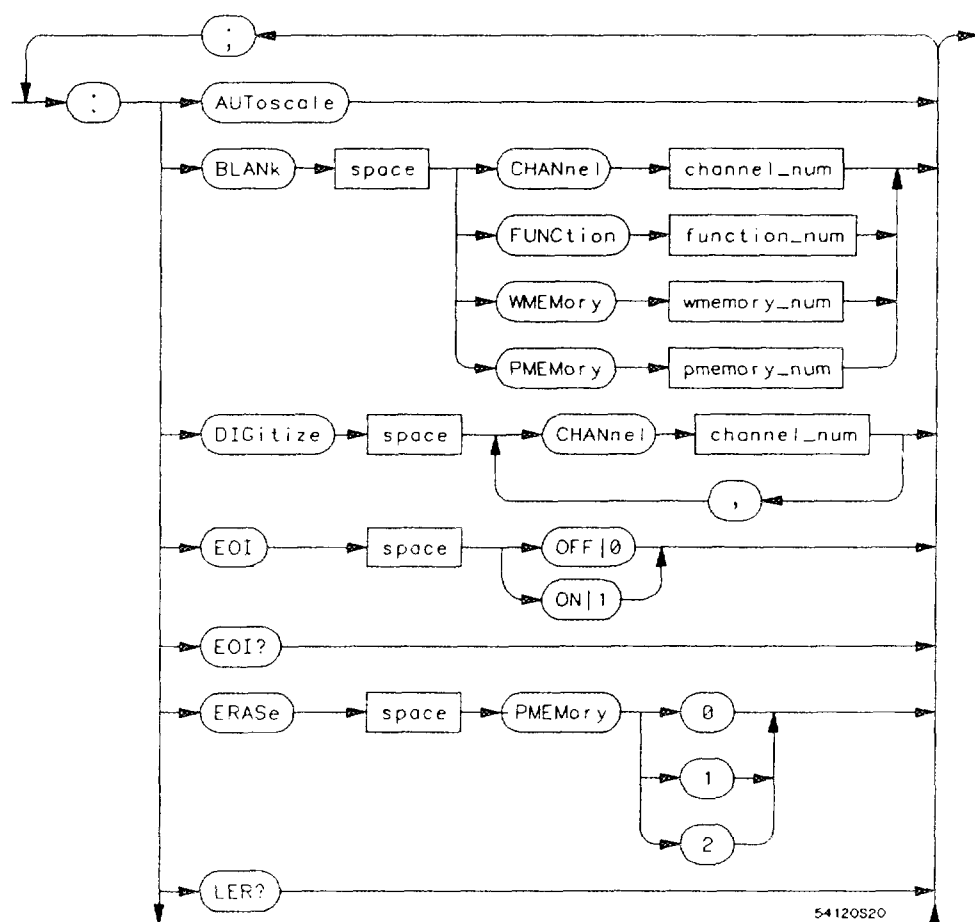


Figure 6-1. Root Level Commands Syntax Diagram

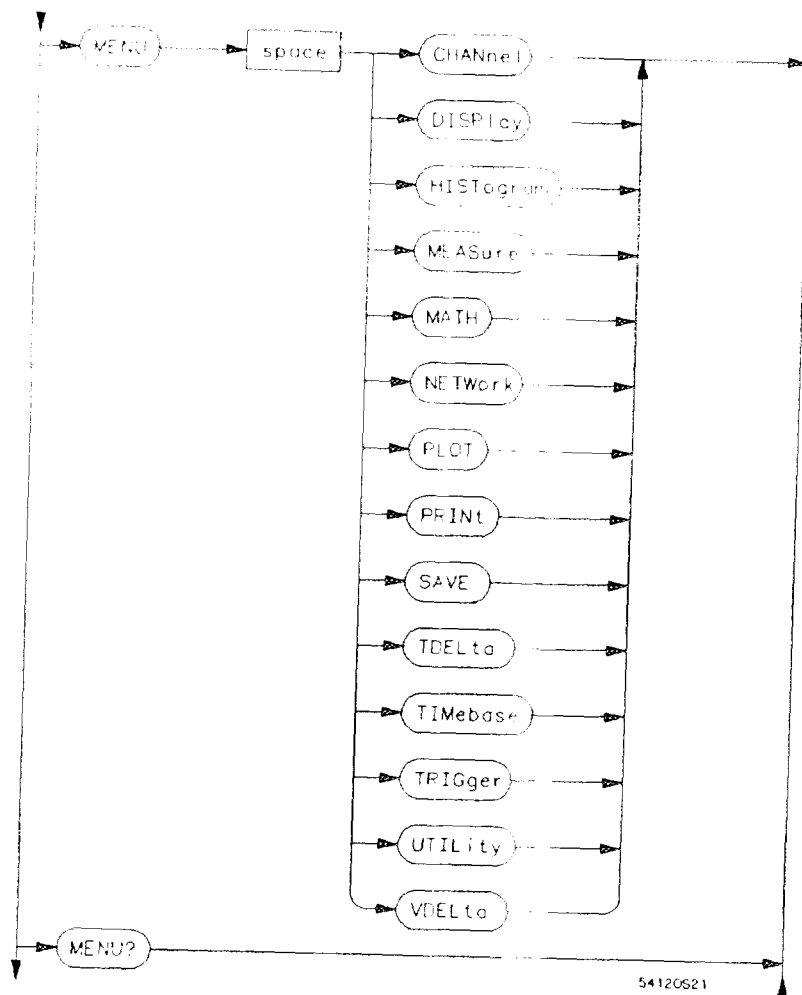
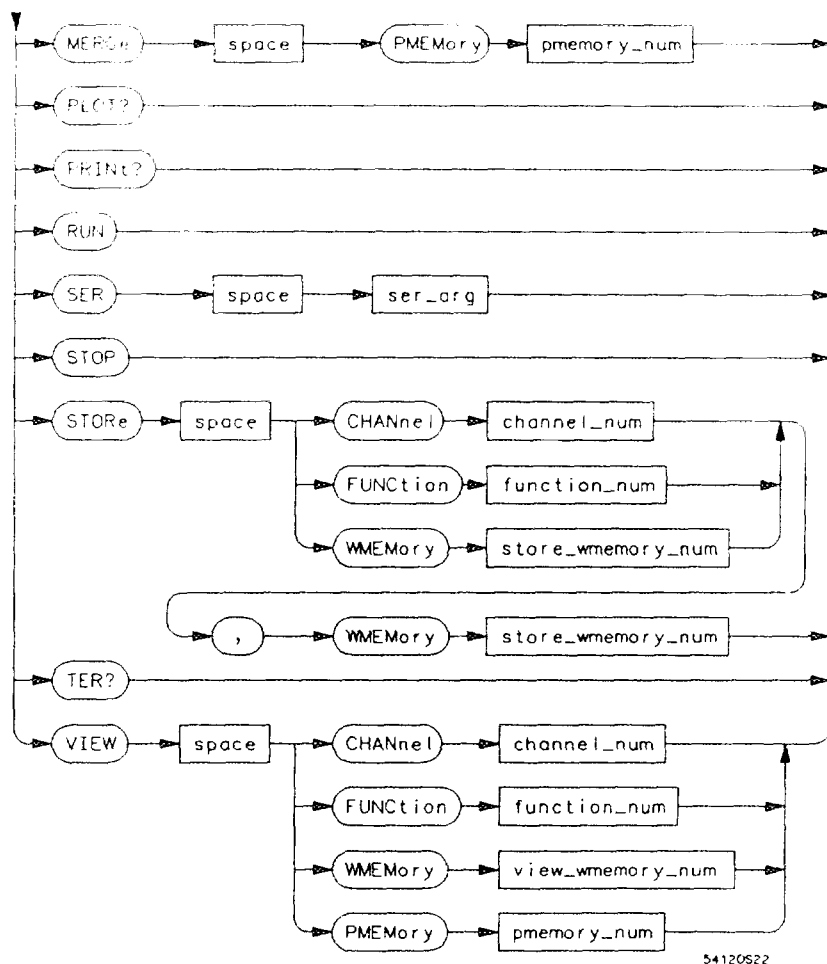


Figure 6-1. Root Level Commands Syntax Diagram (continued)



**channel\_num** = An integer 1 through 4.

**function\_num** = An integer 1 or 2.

**store\_wmemory\_num** = An integer 1 through 4.

**pmemory\_num** = An integer 1 or 2.

**view\_wmemory\_num** = An integer 1 through 5.

**ser\_arg** = A 10 character string

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

## AUToscale

---

### AUToscale

### command

The AUTOSCALE command automatically selects the vertical sensitivity, vertical offset, trigger level, and sweep speed for a display of the input signal. For proper AUTOSCALE operation and input signal requirements, see "Operating Characteristics" in Chapter 19 of the *Front-Panel Reference*. When the AUTOSCALE operation is complete, the Timebase menu will be selected, and the entry devices will be assigned to the TIME/DIV function.

**Command Syntax:** AUToscale

**Example:** OUTPUT 707;":AUTOSCALE"



---

**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 | 5}. To blank a pixel memory display, use :BLANK PMEMORY{1 | 2}, and to blank a function, use :BLANK FUNCTION{1 | 2}.

Function 1 should be blanked before the :VIEW CHANNEL1 command is sent, and Function 2 should be blanked before the :VIEW CHANNEL2 command is sent.

**Command Syntax:** BLANK <display>  
<display> :: = {CHANnel{1 | 2 | 3 | 4} | FUNCTION{1 | 2} |  
WMEMory{1 | 2 | 3 | 4 | 5} | PMEMory{1 | 2}}

**Example:** OUTPUT 707;"BLANK CHANNEL1"

## DIGitize

---

### 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 stored in waveform memory. The ACQUIRE subsystem commands are used to set up conditions such as TYPE, number of POINTS, and the average COUNT for the next DIGITIZE command. See the ACQUIRE subsystem, Chapter 7, for a description of these commands. Data for types normal and average are placed in the same number waveform memory as the channel being digitized. Envelope data is placed in waveform memories 1 and 3 for channels 1 and 3, and in waveform memories 2 and 4 for channels 2 and 4. Histogram data is placed in waveform memory 5.

The sources for the :DIGITIZE command are channels 1 through 4 (see table 6-2).

**Command Syntax:** :DIGitize CHANnel < N > [,CHANnel < N > ,...]  
< N > :: = 1, 2, 3, or 4.

#### Note

*For histogram mode, the digitize source must be the same as the source in the Histogram subsystem. Only one channel may be specified. For envelope mode one odd and one even channel may be specified.*

Example: OUTPUT 707,":DIGITIZE CHANNEL1,CHANNEL4"

Table 6-1. Where Digitized Data is Stored

| :ACQUIRE:TYPE           | :DIGITIZE < source > | WMEMORY         |
|-------------------------|----------------------|-----------------|
| NORMAL<br>or<br>AVERAGE | CHANnel 1            | 1               |
|                         | CHANnel 2            | 2               |
|                         | CHANnel 3            | 3               |
|                         | CHANnel 4            | 4               |
| ENVELOPE                | CHANnel 1            | min = 1,max = 3 |
|                         | CHANnel 2            | min = 2,max = 4 |
|                         | CHANnel 3            | min = 1,max = 3 |
|                         | CHANnel 4            | min = 2,max = 4 |
| HISTOGRAM               | ANY CHANnel          | 5               |

## EOI

---

| EOI | (End or Identify) | command/query |
|-----|-------------------|---------------|
|-----|-------------------|---------------|

The EOI command specifies whether or not the last byte of a reply from the HP 54121T 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).

Refer to chapter 2 for the proper use of EOI.

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;Eoi\$  
PRINT Eoi\$

---

**ERASe****command**

The ERASE command erases a specified pixel memory. Erasing pixel memory 0 is the same as pressing the CLEAR DISPLAY front-panel key. If the oscilloscope is running and being triggered and ERASE PMEMORY0 is executed, the HP 54121T will momentarily stop acquiring data, clear the CRT and continue with data acquisition.

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

**Example:** OUTPUT 707;":ERASE PMEMORY1"

**Note**

*The PMEMORY1 selection over HP-IB is the same as front panel PMEM5, and PMEMORY2 over HP-IB is the same as front panel PMEM6.*

## LER?

---

### LER?

### (Local Event Register)

### query

The LER query allows the LCL 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. A zero indicates a remote-to-local transition has not taken place.

**Query Syntax:** :LER?

**Returned Format:** [:LER] {1 | 0}

**Example:** Output 707;":LER?"  
Enter 707;ler\$  
Print ler\$

## MENU

## command/query

The MENU command selects one of the 14 menus on the front panel.

The MENU query returns the current menu.

**Command Syntax:** :MENU <menu> <NL>  
<menu> :: = {CHANnel | DISPlay | HISTogram | MEASure | MATH |  
NETWork | PLOT | PRINT | SAVE | TDELta | TIMEbase | TRIGger  
| UTILity | VDELta}

**Example:** OUTPUT 707;":MENU DISPlay"

**Query Syntax:** :MENU?

**Returned Format:** [:MENU] <menu> <NL>  
<menu> :: = {CHANnel | DISPlay | HISTogram | MEASure | MATH |  
NETWork | PLOT | PRINT | SAVE | TDELta | TIMEbase | TRIGger  
| UTILity | VDELta}

**Example:** DIM Menu\$(30)  
OUTPUT 707;":MENU?"  
ENTER 707;Menu\$  
PRINT Menu\$

## MERGE

---

### MERGE

### command

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

The active display includes all displayed channels and displayed functions, but not displayed waveform memories.

**Command Syntax:** :MERGe {PMEMory1 | PMEMory2}

**Example:** OUTPUT 707;":MERGE PMEMory2"



## PLOT?

query

The PLOT query outputs a copy of the display to an HPGL compatible plotter as soon as the oscilloscope is addressed to talk.

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

**Command Syntax:** :PLOT?

**Example:**

```

10 CLEAR 707                ! Clear interface buffers
20 OUTPUT 707;":HARDCOPY:PEN AUTOMATIC"
30                          ! Enables automatic pen selection while plotting
40 OUTPUT 707;":HARDCOPY:SOURCE PMEMORY0, FACTORS"
50                          ! Plots the active display and the factors
60 OUTPUT 707;":HARDCOPY:SPEED FAST"
70                          ! Sets the plot speed! for normal paper
80 OUTPUT 707;":PLOT?;"     ! Tells the instrument to plot
90                          ! when next addressed to talk;
100 SEND 7;UNL UNT          ! Unaddress bus, asserts ATN line
110 SEND 7;LISTEN 5         ! Address plotter @ 705 to listen
120 SEND 7;TALK 7           ! Address scope @ 707 to talk
130 SEND 7;DATA             ! Negates ATN line to allow data transfer
140 WAIT 30                 ! Waits for the plot to complete
150 END

```

**Note**

*Refer to the program examples, Appendix C, for using SRQ with plot.*

## PRINT?

---

## 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, graticule, time and voltage markers, horizontal and vertical setup, and measurement results.

**Command Syntax:** :PRINT?

**Example:**

```
10 CLEAR 707                ! Clear interface buffers
20 OUTPUT 707;":HARDCOPY:PRINTER DEFAULT"
30                          ! Selects print device as default printer
40 OUTPUT 707;":HARDCOPY:SOURCE PMEMORY0, FACTORS"
50                          ! Prints the active display and the factors
60 OUTPUT 707;":HARDCOPY:PAGE AUTOMATIC"
70                          ! Causes a printer formfeed when print is complete
80 OUTPUT 707;":PRINT?;"    ! Tells the instrument to print
90                          ! when next addressed to talk;
100 SEND 7;UNL UNT          ! Unaddress bus, asserts ATN line
110 SEND 7;LISTEN 1         ! Address printer @ 701 to listen
120 SEND 7;TALK 7           ! Address scope @ 707 to talk
130 SEND 7;DATA             ! Negates ATN line to allow data transfer
140 WAIT 30                 ! Waits for data transfer to complete
150 END
```

### Note

*When programming the HP 54121T you should use the SRQ (Service Request) capabilities to determine if the print data transfer is complete. Attempting to program the instrument while making a print data transfer will cause errors.*

---

**RUN****command**

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

## SER

---

### SER

(Serial)

command

The SER 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.

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

**Command Syntax:** :SER < string >  
< string > ::= 10 character serial number

**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.

**Note**

*An Autoscale operation, a Network Preset, and some measurements will also restart acquiring data.*

**Command Syntax:** :STOP

**Example:** OUTPUT 707;":STOP"

# STORe

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, which can be channel 1, 2, 3, or 4; function 1 or 2; or waveform memory 1 through 4. The second parameter is the destination of the waveform, which can be waveform memory 1 through 4.

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

**TER?**

**TER?**

**(Trigger Event Register)**

**query**

The TER query allows the TRG Event Register to be read. When the TRG Event Register is read, it is cleared. A one indicates a trigger has occurred. A zero indicates a trigger has not occurred. This bit will not be set if :TIMEBASE:MODE FREERUN is selected.

**Query Format:** :TER?

**Returned Format:** [:TER] {1 | 0}

**Example:** OUTPUT 707;":TER?"  
ENTER 707; Ter\$  
PRINT Ter\$

# VIEW

---

## VIEW

## command

The VIEW command causes the HP 54121T to turn on (start displaying) an active channel, function, pixel memory, waveform memory, or histogram. If you want to display a channel use the parameter CHANNEL{1 | 2 | 3 | 4}. If you want to display a pixel memory, use the parameter PMEMORY{1 | 2}. To display a function, send :VIEW FUNCTION{1 | 2}. To display a histogram send :VIEW WMEMORY5. Using the :VIEW WMEMORY{1 | 2 | 3 | 4 | 5} command in the DUAL screen format causes odd numbered memories to be displayed on the upper screen and even numbered memories to be displayed on the lower screen. When the HP 54121T is in QUAD screen mode, the memories are displayed on their own graticule. For example, WMEMORY1 is displayed in the top graticule.

**Command Syntax:** :VIEW {CHANnel{1 | 2 | 3 | 4} | FUNCtion{1 | 2} | PMEMory{1 | 2} | WMEMory{1 | 2 | 3 | 4 | 5}}

**Example:** OUTPUT 707;":VIEW CHANNEL1"

### Note

*Function 1 should be blanked before you try to view channel 1, and Function 2 should be blanked before you try to view channel 2.*



# System Subsystem

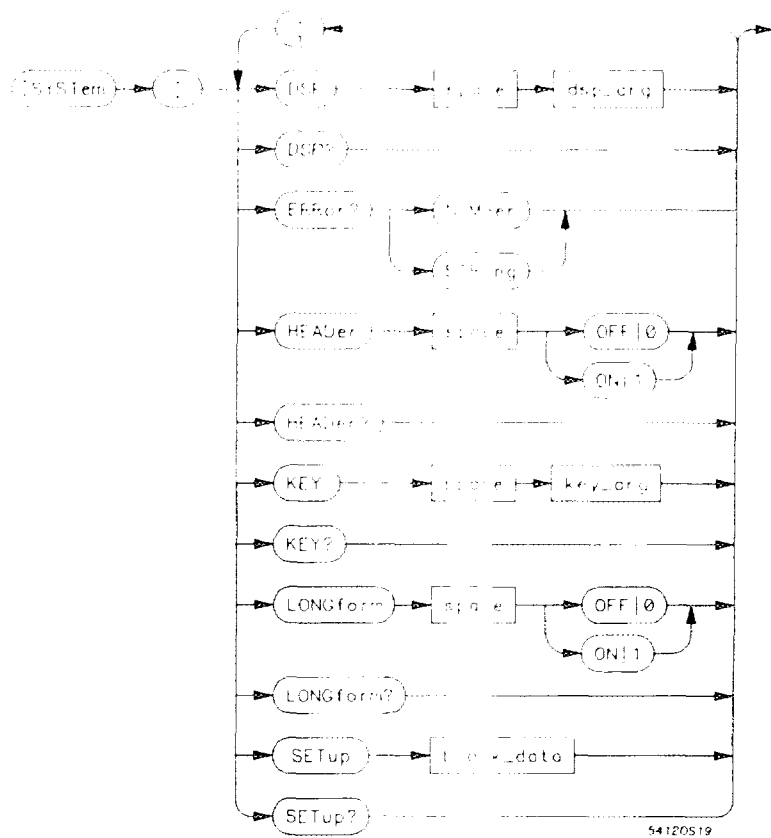
---

7

## Introduction

System 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 the system commands syntax diagram.



**dsp\_arg** = any quoted string

**key\_arg** = keycode 1 through 63

**block\_data** = data in IEEE 488.2 # format

*Figure 7-1. System Commands Syntax Diagram*

**DSP****command/query**

The DSP command writes a quoted string, excluding quotes, to the advisory line (line 15) of the 21 lines on the CRT.

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

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

**Example:** OUTPUT 707;"SYSTEM:DSP ""COLOR DISPLAY""

**Query Syntax:** SYSTem:DSP?

**Returned Format:** [SYSTem:DSP] <string> <NL>  
<string> :: = last information written on line 15

**Example:** DIM Dsp\$[40]  
OUTPUT 707;"SYSTEM:DSP?"  
ENTER 707;Dsp\$  
PRINT Dsp\$

## ERRor?

---

## ERRor?

## query

The 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, ERROR?, returns the error numbers in the order that they occurred until the queue is empty. Any further queries then return 0's until another error occurs. See table 6-1 for the ERROR numbers.

**Query Syntax:** :SYSTem:ERRor?

**Returned Format:** [:SYSTem:ERRor] <code> <NL>  
<code> :: = integer error code

**Example:**

```
Emsg = 1
WHILE(Emsg)
  OUTPUT 707;"ERROR?"
  ENTER 707;Emsg
  PRINT Emsg
END WHILE
```

This example reads and prints the contents of the error queue until it is empty. In this example, the WHILE statement will loop as long as "Emsg" is evaluated to be true. Emsg is true as long as it is anything other than 0.

*Table 7-1. Error Messages*

| ERROR NUMBER | DESCRIPTION   |
|--------------|---|
| — 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 too 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 is return-to-local

# HEADer

## HEADer

## command/query

The 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>

**Example:** DIM Header\$[10]  
OUTPUT 707;":SYSTEM:HEADER?"  
ENTER 707;Header\$  
PRINT Header\$

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

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

With headers ON and 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 **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. Only one key code should be sent in each program message.

The **KEY** query returns the key code for the last key pressed from the front panel or for the last simulated key press over the HP-IB. Key codes range from 1 to 63, zero represents no key and will be returned after power up.

Refer to table 7-2 for key codes.

**Command Syntax:** :SYSTem:KEY <keycode>  
<keycode> :: = 1 to 63

**Example:** OUTPUT 707;":SYSTEM:KEY 48"

**Query Syntax:** :SYSTem:KEY?

**Returned Format:** [:SYSTem:KEY] <keycode> <NL>  
<keycode> :: = 0 through 63 (integer - NR1 format)

**Example:** DIM Key\$[20]  
OUTPUT 707;":SYSTEM:KEY?"  
ENTER 707;Key\$  
PRINT Key\$

# KEY

*Table 7-2. HP 54121T Front-Panel Key Codes*

| KEY                | KEY CODE | KEY           | KEY CODE |
|--------------------|----------|---------------|----------|
| Menu Select 1      | 1        | "-" (minus)   | 23       |
| Menu Select 2      | 2        | 0             | 24       |
| Menu Select 3      | 3        | 1             | 25       |
| Menu Select 4      | 4        | 2             | 26       |
| Menu Select 5      | 5        | 3             | 27       |
| Menu Select 6      | 6        | 4             | 28       |
| Menu Select 7      | 8        | 5             | 29       |
| Menu Select 8      | 9        | 6             | 30       |
| Function Select 1  | 15       | 7             | 31       |
| Function Select 2  | 14       | 8             | 32       |
| Function Select 3  | 13       | 9             | 33       |
| Function Select 4  | 12       | CLEAR DISPLAY | 40       |
| Function Select 5  | 11       | RUN           | 41       |
| Function Select 6  | 10       | STOP          | 42       |
| sec/Volt           | 16       | SAVE SETUP    | 43       |
| msec/mV            | 17       | RECALL SETUP  | 44       |
| m sec              | 18       | LOCAL         | 45       |
| nsec               | 19       | AUTOSCALE     | 48       |
| psec               | 20       | ↕→            | 56       |
| CLEAR              | 21       | ←↓            | 63       |
| ". " (decimal pt.) | 22       | no key        | 0        |

The menu select keys are at the bottom of the screen with menu select 1 at the lower left of the screen. The function select keys are at the right of the screen with function select 1 at the upper right of the screen.



---

**LONGform****command/query**

The LONGFORM command sets the longform variable which tells the HP 54121T how to format query responses. If the LONGFORM command is set to OFF, command headers and alpha arguments are sent from the HP 54121T 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 54121T. Headers and arguments may be sent to the HP 54121T in either the longform or shortform regardless of how the LONGFORM command is set.

The LONGFORM query returns the state of the LONGFORM command.

**Note**

*Even though the command can be sent by 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>

**Example:** DIM Long\$(30)  
OUTPUT 707;":SYSTEM:LONGFORM?"  
ENTER 707;Long\$  
PRINT Long\$

# SETup

## SETup

## command/query

The SETUP command sets the HP 54121T as defined by the data in the learn string sent from the controller. The setup string contains 272 bytes of setup data. The 272 bytes does not include the header or "#3272".

The SETUP query outputs the current HP 54121T 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

**Example:** OUTPUT 707;":SYSTEM:SETUP "; <setup >  
<setup > :: = # 3272 <setup data >

**Query Syntax:** :SYSTEM:SETup?

**Returned Format:** [:SYST:SETup] <setup >  
<setup > :: = # 3272 <setup data >

**Example:**

|  |   |
|--|---|
| 10 DIM Set\$[400]                                | ! Setup the instrument as desired             |
| 20 OUTPUT 707;":SYST:HEAD OFF"                   |   |
| 30 OUTPUT 707;":SYSTEM:SETUP?"                   | ! Transfer the instrument setup to controller |
| 40 ENTER 707 USING " - K";Set\$                  | ! Store the setup PAUSE                       |
| 50   | ! Change the setup as desired                 |
| 60 OUTPUT 707 USING " # ,K";":SYST:SETUP ";Set\$ |   |
| 70   | ! Return the instrument to first setup        |

### 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 54121T.*

# 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 syntax diagram.

The ACQUIRE subsystem is also the only HP-IB control for three display parameters: Display Mode, Number of Averages, and Bandwidth. 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 follow, and that when the front-panel parameters are changed, the HP-IB parameters will follow.

---

## Persistence (Normal) Mode

The :ACQUIRE:TYPE NORMAL command will set the HP 54121T to the variable persistence mode. The front-panel user activates the same mode by selecting the Display menu, then setting the display mode to persistence. The persistence time is set in the HP-IB DISPLAY subsystem by :DISPLAY:PERSISTENCE <time> and is applied when the variable persistence mode is entered. The :ACQUIRE:COUNT can be set in the persistence mode, but has no impact on the current display mode or HP-IB acquisition. The :ACQUIRE:COUNT query will return a 1.

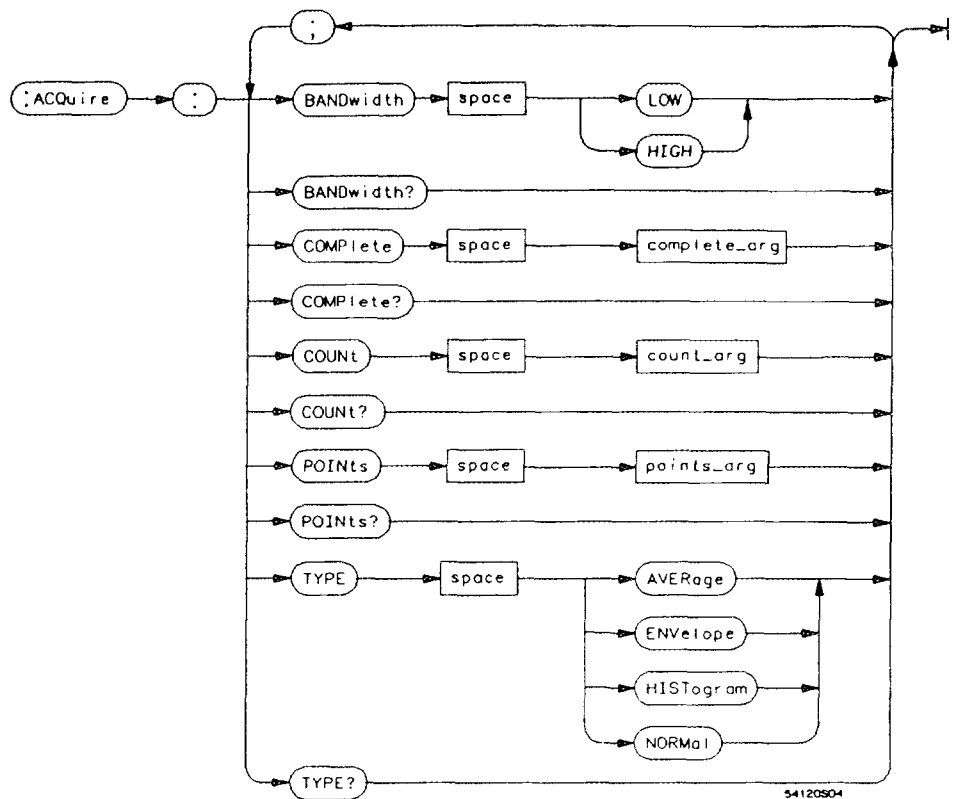
---

## Averaging Mode

The `:ACQUIRE:TYPE AVERAGE` command sets the HP 54121T to the Averaging ON mode.

COUNT can be set in Average mode by sending the `:ACQUIRE:COUNT` command followed by the number of averages. The command determines the number of averages that must be acquired before the digitize is complete.

To activate the averaging mode from the front panel, select the Display menu, then set the Disp Mode to Averaged. Changing the number of averages changes the COUNT value.



**complete\_arg** ::= An integer, 0 through 100.

**count\_arg** ::= An integer. It specifies the number of values to average for each time point when the HP 54121T is in averaged mode, the number of samples to take for a histogram, or the number of hits per each time point to form the envelope in ENVELOPE ACQUISITION mode. The count in ENVELOPE mode can be 1 through 2048. The count in HISTOGRAM mode can be 100 to 655000000. In AVERAGE mode the value must be a power of two between 2 and 2048. For NORMAL (persistence) mode the count is 1.

**points\_arg** ::= 100 to 1024.

Figure 8-1. Acquire Subsystem Syntax Diagram Bandwidth

## BANDwidth

---

### BANDwidth

### command/query

When bandwidth is set to **HIGH**, the hardware is changed, increasing the -3 dB bandwidth of all channels. This is at the expense of an increased noise floor.

**HIGH** is the same as the front-panel 20 GHz selection and **LOW** is the same as the front-panel 12.4 GHz selection.

The **BANDWIDTH** query returns the current selected mode.

**Command Syntax:** :ACQuire:BANDwidth {HIGH | LOW}

**Example:** OUTPUT 707;":ACQUIRE:BANDWIDTH LOW"

**Query Syntax:** :ACQuire:BANDwidth?

**Returned Format:** [:ACQuire:BANDwidth] {LOW | HIGH} <NL>

**Example:** DIM Type\$[30]  
OUTPUT 707;":ACQUIRE:BAND?"  
ENTER 707;Type\$  
PRINT Type\$

## **COMPlete**

### **COMPlete**

### **command/query**

The HP 54121T always uses 100% but will accept a range of 0 to 100 as input parameters.

The COMPLETE query always returns 100%.

# COUNT

## COUNT

### command/query

:ACQUIRE:COUNT specifies: the number of values to average for each time point when the HP 54121T is in the averaged mode; the number of samples to take for a histogram; the number of hits per time point to form the envelope in envelope mode.

The COUNT query returns the last specified count value. The COUNT parameter is an integer from 1 through 655000000 depending on acquisition type.

#### Note

*Any value can be sent, however the value will be rounded to the nearest legal value for the selected TYPE.*

**Command Syntax:** :ACQUIRE:COUNT <count>  
<count> :: = (Chart Below)

**Example:** OUTPUT 707;":ACQUIRE:TYPE AVERAGE;COUNT 1000"

#### Note

*The count will be rounded to 1024, the nearest power of 2.*

| Mode       | Accepted Values          |
|------------|--------------------------|
| Normal     | 1                        |
| Average    | 1 to 2048 in powers of 2 |
| Envelope   | 1 to 2048                |
| Histograms | 100 to 655000000         |



## COUNT

---

**Query Syntax:** :ACQuire:COUNT?

**Returned Format:** [:ACQuire:COUNT] <count> <NL>  
<count> :: = 1 through 655000000  
(integer - NR1 format)

**Example:** OUTPUT 707;":ACQ:COUNT?"  
ENTER 707;Cnt  
PRINT Cnt

## POINTs

## POINTs

## command/query

The :ACQUIRE:POINTS command determines the length of the acquisition record (how many time points).

This command has no effect on the acquisition if type is histogram.

For voltage histograms the number of points = 256.

For time histograms the number of points = 500.

| Time/division            |               |                                |
|--------------------------|---------------|--------------------------------|
| equal to or greater than | but less than | number of points               |
| 10 ps/div                | 20 ps/div     | 100 or 400                     |
| 20 ps/div                | 50 ps/div     | 100, 400, or 500               |
| 50 ps/div                | 200 ps/div    | 100, 500, or 1000              |
| 200 ps/div               | 1 s/div       | 128, 256, 500, 512,<br>or 1024 |

The :ACQUIRE:POINTS query returns the number of points to be acquired.

**Command Syntax:** :ACQuire:POINts <points>  
<points> :: = 100 - 1024 as defined above

**Query Syntax:** :ACQuire:POINts?

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

**Returned Format:** [:ACQuire:POINts] <points> <NL>  
<points> :: = 100 - 1024 as defined above

**Example:** OUTPUT 707;":ACQUIRE:POINTS?"  
ENTER 707;Points  
PRINT Points

**TYPE**

**command/query**

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

**Type Average or Normal**

Channels 1 through 4 may be digitized simultaneously. Functions may not be digitized. When the acquisition is complete, the waveform data for each channel will be transferred to the waveform memory, which has the same number as the channel. For example, a :DIGITIZE CHANNEL3 would have its data stored in waveform memory 3.

**Type Histogram**

Digitize causes a histogram to be created based on the previously set histogram parameters. Histogram data always goes to the histogram buffer (waveform memory 5). Histograms can only be computed for one channel at a time.

**Type Envelope**

Envelope mode computes both the maximum and minimum values that fall in each bucket. Only two channels can be digitized at once in this mode, one even-numbered channel and one odd-numbered channel. The data for the odd-numbered channel is stored in waveform memories 1 (min) and 3 (max). Data for the even-numbered channel is stored in waveform memories 2 (min) and 4 (max).

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

**Command Syntax:** :ACQUIRE:TYPE {NORMAL | AVERAGE | ENVELOPE | HISTOGRAM}

**Example:** OUTPUT 707,":ACQUIRE:TYPE ENVELOPE"

**Query Syntax:** :ACQUIRE:TYPE?

**Returned Format:** [:ACQUIRE:TYPE] <type> <NL>  
<type> :: = { NORMAL | AVERAGE | ENVELOPE | HISTOGRAM }

**Example:** DIM Type\$(30)  
OUTPUT 707,":ACQUIRE:TYPE?"  
ENTER 707;Type\$  
PRINT Type\$



# Calibrate Subsystem

9

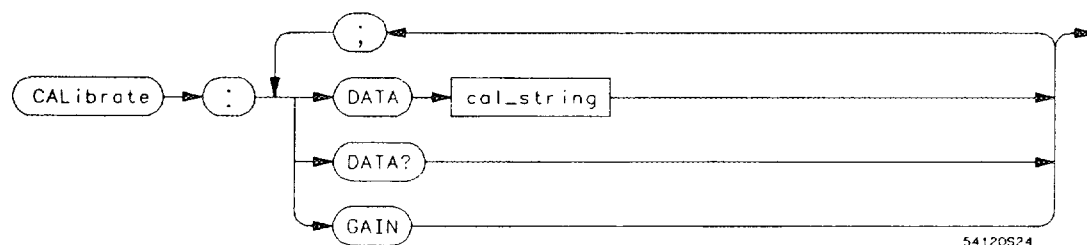
## Introduction

The CALIBRATE subsystem contains two commands: GAIN and DATA.

The GAIN command will initiate a channel vertical calibration. This is the same thing that happens when the Channel Vertical Cal key in the Utility Cal menu is pressed.

The DATA command and DATA? query allow you to store Skew Calibration factors for various setups or measurement conditions (cable lengths, etc.). This data can then be returned to the instrument when you wish to repeat a particular setup. Skew calibration factors are set up in the Channel Skew Cals menu.

Refer to figure 9-1 for the CALIBRATE subsystem syntax diagram.



**cal\_string** :: = Skew calibration data in the format # 216 < 16 dabs >

Figure 9-1. Calibrate Subsystem Syntax Diagram

# DATA

---

## DATA

### command/query

The :CALIBRATE:DATA command allows you to store the Skew Calibration factors that are set in the Cal menu. The cal string can be sent to the controller, stored, then returned to the instrument when you need the calibration restored.

The Cal String consists of 16 eight-bit bytes containing the Delay Calibration factors.

The :CALIBRATE:DATA query sends the Cal String to the controller with the same format required by the :CALIBRATE:DATA command. The string should not be modified between the time that it is received from the instrument with the query and the time that it is sent back to the instrument.

**Command Syntax:** :CALibrate:DATA <cal string>  
<cal string> :: = # 216 <16 dabs> <NL>

**Example:** OUTPUT 707;":CAL:DATA" <cal string>

**Query Syntax:** :CALibrate:DATA?

**Returned Format:** [:CALibrate:DATA] <cal string>  
<cal string> :: = # 216< 16 dabs> <NL>

**Example:**

```

10 CLEAR 707
20 DIM A$(100)
31 ALLOCATE INTEGER Skew(1:8)
40 OUTPUT 707;":SYST:HEAD OFF"
50 OUTPUT 707;":CAL:DATA?"
60 ENTER 707 USING " # ,4A";A$
70 ASSIGN @ Scope TO 707
80 ASSIGN @ Scope;FORMAT OFF
90 ENTER @ Scope;Skew(*)
100 ASSIGN @ Scope;FORMAT ON
110 ENTER 707 USING " # ,A";Line$
120 PAUSE
125 OUTPUT 707 USING " # ,K";":CAL:DATA "
127
130 OUTPUT 707 USING " # ,K";A$
132
133 ASSIGN @ Scope;FORMAT OFF
140 OUTPUT @ Scope;Skew(*)
141
143 ASSIGN @ Scope;FORMAT ON
150 OUTPUT 707 USING " # ,K";Line$
155
160 END

```

! Sets length of A\$  
! Define an integer array  
! Turn Headers off  
! Send data from instrument  
! Read # 216 into A\$  
! Assign path name to bus address 707  
! Turn controller formatter off  
! Read skew data to the array SKEW  
! Turn controller formatter on  
! Read the linefeed character into Line\$  
! Send command header to  
! the instrument  
! Send # 216 to instrument  
! from A\$  
! Turn controller formatter off  
! Send Cal Skew data  
! to instrument  
! Turn controller formatter on  
! Send linefeed to controller  
! from Line\$

In this program, lines 10 through 110 get the data from the instrument and store it in the controller. You must press the "Continue" key on the controller in order to send the calibrate data back to the instrument.

# GAIN

---

## GAIN

## command

The :CALIBRATE:GAIN command initiates a channel vertical gain calibration. This is the same thing that happens when the Channel Vertical Gain in the Calibration menu is pressed.

There is no query form of this command.

**Command Syntax:** :CALibrate:GAIN

**Example:** OUTPUT 707;":CAL:GAIN"

### Note

*All front-panel inputs must be disconnected from signal source before this command is sent.*



# Channel < N > Subsystems

---

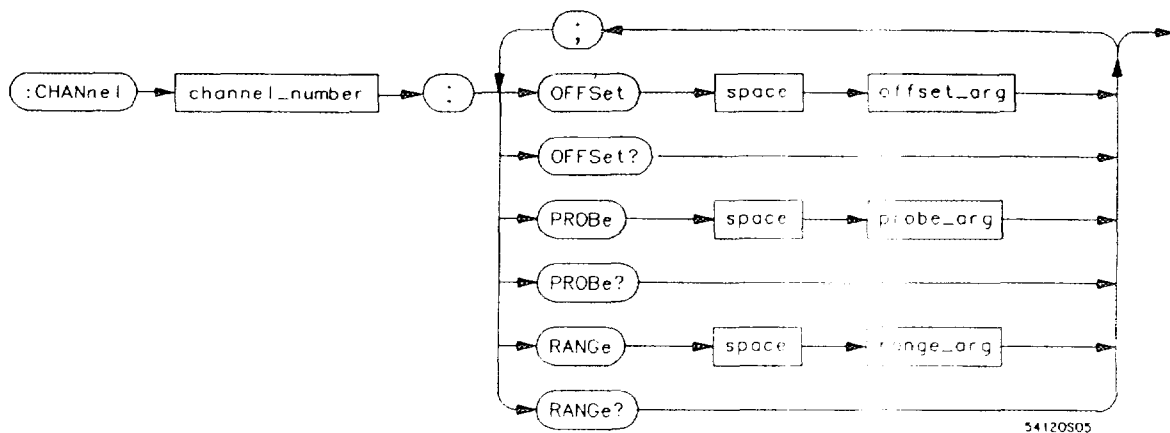
10

## Introduction

The CHANNEL subsystem commands control the channel display and vertical or Y axis of the HP 54121T. Channels 1, 2, 3, and 4 are independently programmable for all offset, probe, and range functions. See figure 10-1 for a syntax diagram of the channel subsystem commands.

There are four channel subsystems. The subsystem is entered with the :CHANNEL < N > command, where < N > is 1,2,3, or 4.

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. The value of offset is  $-500\text{ mV}$  to  $500\text{ mV}$  when 1:1 probe attenuation is being used. If the probe attenuation is changed, allowable offset values are multiplied by the probe attenuation factor.

**probe\_arg** = A real number from 1.0 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. The RANGE can be set to any value from  $8\text{ mV}$  to  $640\text{ mV}$  in  $1\text{ mV}$  increments when 1:1 probe attenuation is being used. If the probe attenuation is changed, allowable range values are multiplied by the probe attenuation factor.

*Figure 10-1. Channel <N> Subsystems Syntax Diagram*

## OFFSet

## command/query

The OFFSET command sets the voltage that is represented at center screen for the selected channel. The value of offset is  $-500\text{ mV}$  to  $500\text{ mV}$  when the probe attenuation is 1:1. If the probe attenuation is other than 1:1, the allowable offset value is multiplied by the probe attenuation factor.

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

### Command Syntax:

```
:CHANnel <N> :OFFSet <value>
<N> :: = 1, 2, 3, or 4
<value> :: = offset value (  $-500\text{ mV}$  to  $500\text{ mV}$ )
```

### Example:

```
OUTPUT 707;":CHANNEL2:OFFSET 32e3"
OUTPUT 707;":CHAN2:OFFS 32 mV"
OUTPUT 707;":CHAN1:OFFS 200M;:CHAN2:OFFSET .032"
```

### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

### Query Syntax:

```
:CHANnel <N> :OFFSet?
<N> :: = 1, 2, 3, or 4
```

### Returned Format:

```
[:CHANnel <N> :OFFSet] <value> <NL>
<N> :: = 1, 2, 3, or 4
<value> :: = offset value in volts
(exponential - NR3 format)
```

### Example:

```
OUTPUT 707;":CHANNEL3:OFFSET?"
ENTER 707;Offset
PRINT Offset
```

# PROBe

## PROBe

## command/query

The PROBE command specifies the probe attenuation factor for the selected channel. The range of the probe attenuation factor is from 0.000001 to 1000.0.

### Note

*This command does not change the actual input sensitivity of the HP 54121T. It changes the reference constants for scaling the display factors, automatic measurements, and trigger levels, etc.*

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

### Command Syntax:

:CHANnel < N > :PROBe < atten >  
< N > :: = 1, 2, 3, or 4  
< atten > :: = 0.000001 to 1000

### Example:

OUTPUT 707;":CHANNEL4:PROBE 10"

### Query Syntax:

:CHANnel < N > :PROBe?  
< N > :: = 1, 2, 3, or 4

### Returned Format:

[ :CHANnel < N > :PROBe ] < atten > < NL >  
< N > :: = 1, 2, 3, or 4  
< atten > :: = 0.000001 to 1000  
(exponential - NR3 format)

### Example:

OUTPUT 707;":CHANNEL1:PROBE?"  
ENTER 707:Probe  
PRINT Probe

## RANGe

## command/query

The RANGE command defines the full-scale vertical axis of the selected channel. The RANGE can be set to any value from 8 mV to 640 mV, when 1:1 probe attenuation is being used. 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>
<N> :: = 1, 2, 3, or 4
<range> :: = .008 V to .640 V when 1:1 probe attenuation is being used
<multiplier> :: = as defined in chapter 3
```

### Example:

```
OUTPUT 707;":CHANNEL1:RANGE .520"
OUTPUT 707;":CHAN1:RANG 520 mV"
OUTPUT 707;":CHAN1:RANG 52E-2"
```

### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

### Query Syntax:

```
:CHANnelN:RANGe?
```

### Returned Format:

```
[[:CHANnel<N>:RANGe] <range> <NL>
<N> :: = 1, 2, 3, or 4
<range> :: = .008 V to .640 V when 1:1 probe attenuation is being used
(exponential - NR3 format)
```

### Example:

```
OUTPUT 707;":CHAN3:RANGE?"
ENTER 707;Range
PRINT Range
```



## Introduction

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

### Note

*The command that changes the DISPLAY mode from average to persistence is :ACQUIRE:TYPE NORMAL. The command that changes the display mode from persistence to average is :ACQUIRE:TYPE AVERAGE. The command that controls the number of averages is ACQUIRE:COUNT.*

See figure 11-1 for a syntax diagram of the DISPLAY subsystem.

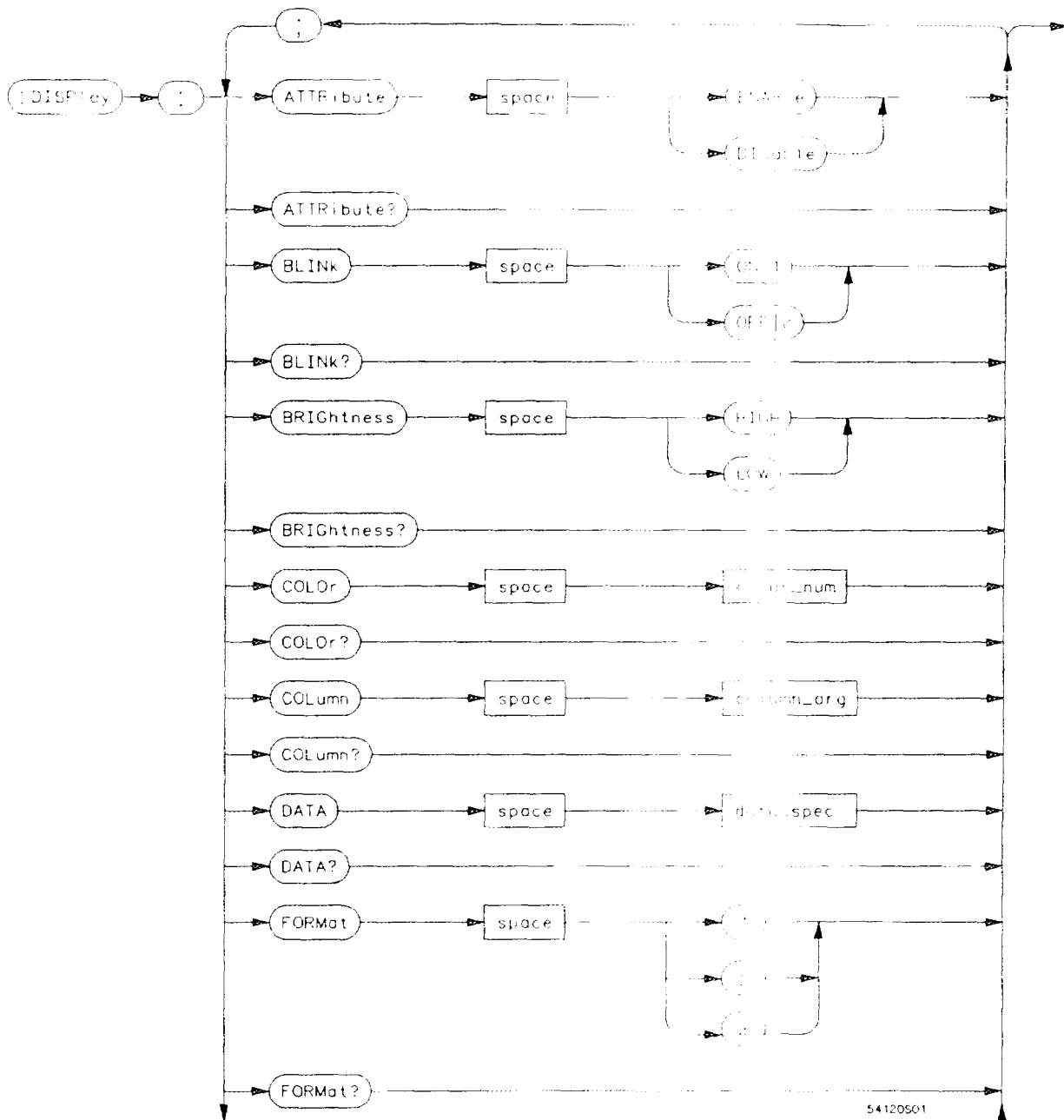


Figure 11-1. Display Subsystem Syntax Diagram



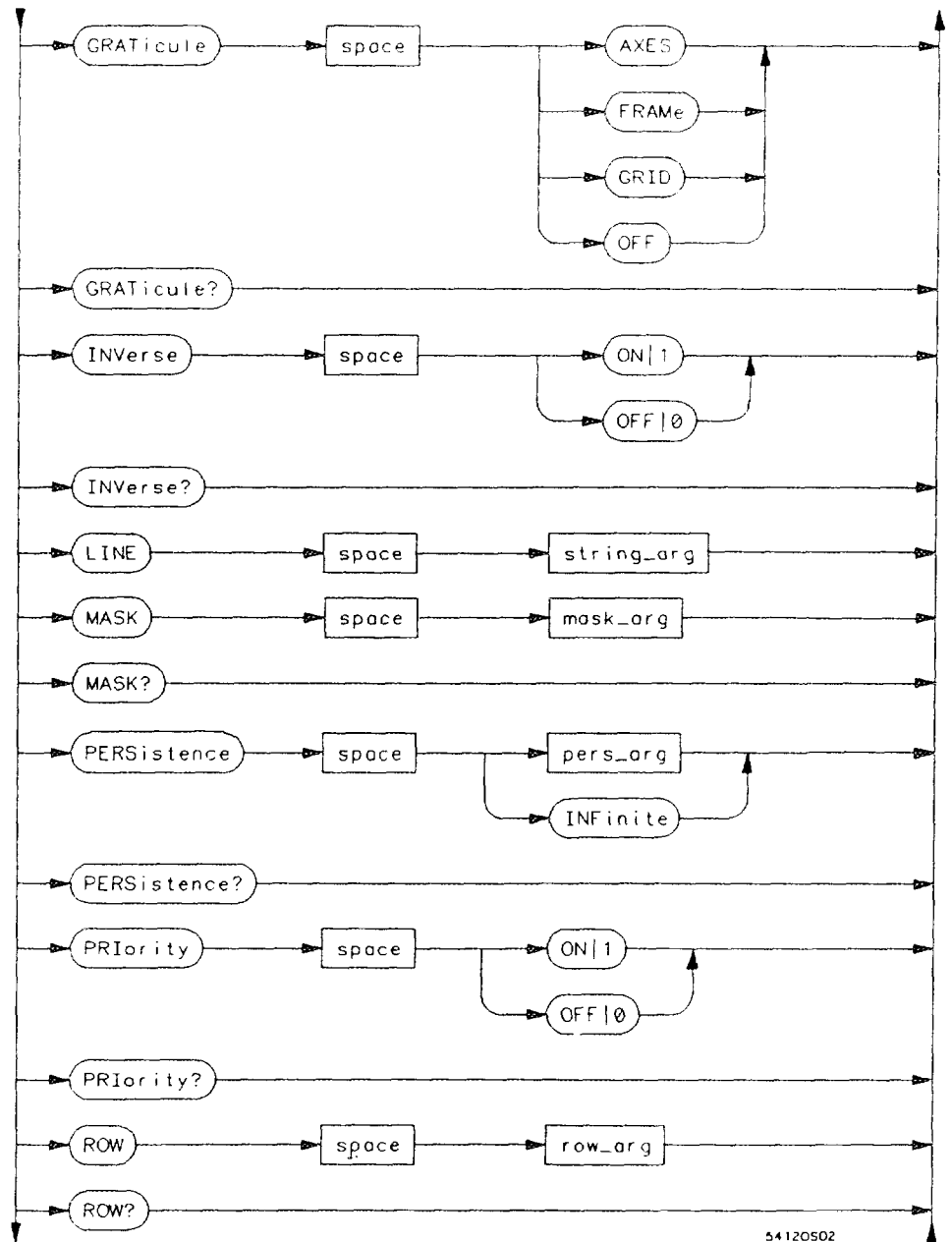
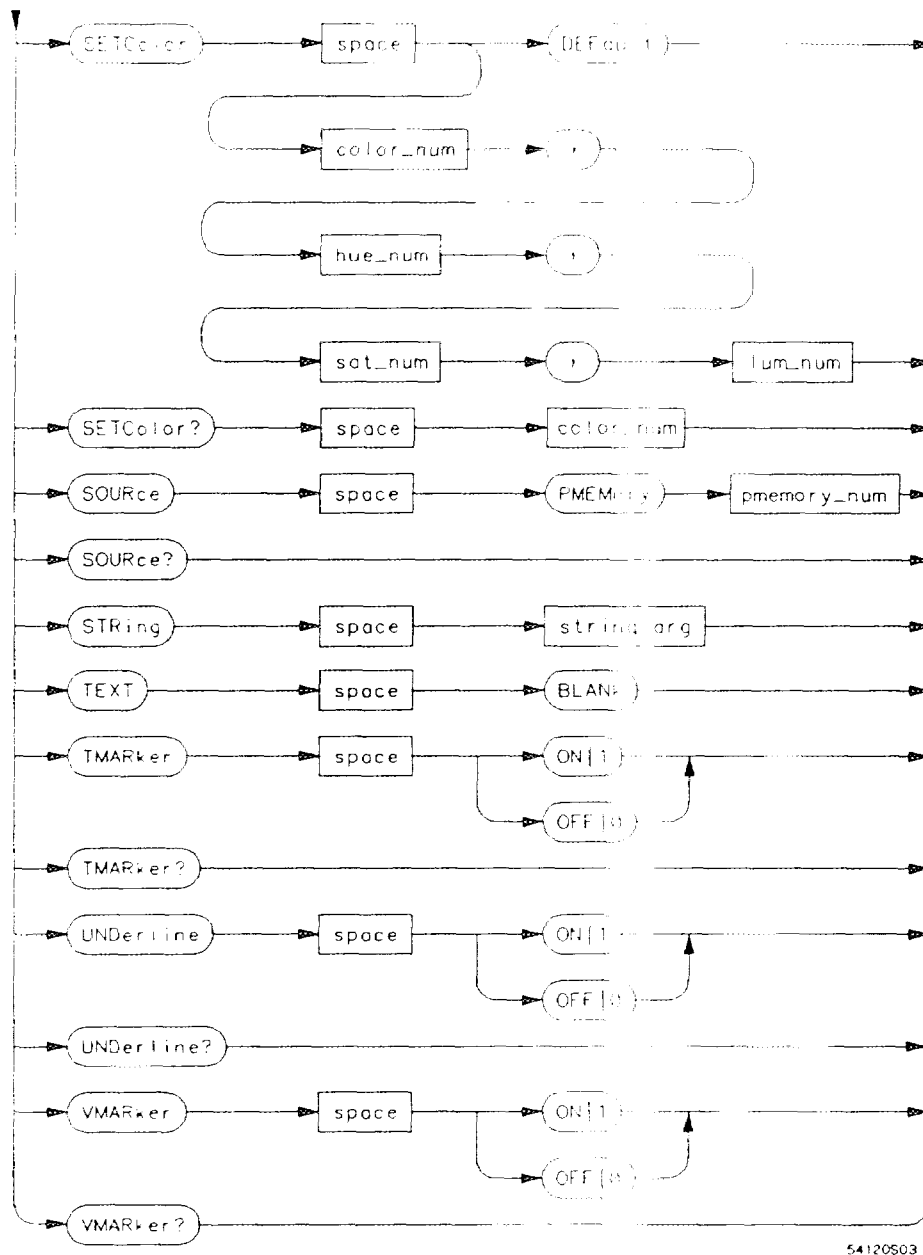


Figure 11-1. Display Subsystem Syntax Diagram (continued)



54120503

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

**column\_arg** = An integer from 0 through 71.

**color\_number** = An integer from 0 through 15.

**data\_spec** = A block of data in IEEE 488.2 definite binary block format.

**hue\_num** = An integer, 0 through 100.

**line\_arg** = Any quoted string.

**lum\_num** = An integer, 0 through 100.

**mask\_arg** = An integer, 0 through 255.

**pers\_arg** = A real number 0.3 through 10, or infinite.

**pmemory\_num** = An integer, 0 through 8.

**row\_arg** = An integer, 0 through 22.

**sat\_num** = An integer, 0 through 100.

**string\_arg** = Any quoted string.

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

## ATTRibute

---

### ATTRibute

### command/query

The :DISPLAY:ATTRIBUTE command controls the attributes used to display text in strings sent with the DSP system subsystem command or with the LINE or STRING commands in the DISPLAY subsystem. Attributes can be embedded in the text when any of these commands are used. The attributes control how the text will be displayed, the color of the text display, and whether or not it will blink, be in inverse video, or be underlined. If no attributes are used the text will be written in color # 1 (default gray) and will not blink, be in inverse video, or be underlined. Refer to figure 11-2 for embedded attribute byte information. The text attributes include INVerse, UNDERline, BLINK, COLOr.

When this command is enabled, the last attributes override previously set attributes. This means that the text display can be changed by sending new attributes to highlight key words, etc. When the ATTRIBUTE command is disabled, any new attributes are disabled. This command has no effect on text that was sent previous to the enable or disable command.

The ATTRIBUTE query returns the state of the command, which is either ENABLE or DISABLE.

**Command Syntax:** :DISPlay:ATTRibute {DISable | ENABLE}

**Example:** OUTPUT 707;":DISPLAY:ATTRIBUTE ENABLE"

**Query Syntax:** :DISPlay:ATTRibute?

**Returned format:** [:DISPlay:ATTRibute] <state> <NL>  
<state> :: = {ENABLE | DISABLE}

**Example:** DIM Attr\$[30]  
OUTPUT 707;":DISPLAY:ATTRIBUTE?"  
ENTER 707;Attr\$  
PRINT Attr\$

| COLOR BITS                                |  |    |    |   |   |                       |                                 |
|---|--|----|----|---|---|-----------------------|---------------------------------|
| A<br>T<br>T<br>R<br>I<br>B<br>U<br>T<br>E | 0 = BEIGE<br>8 = GRAY<br>16 = RED<br>24 = YELLOW<br>32 = GREEN<br>40 = ORANGE<br>48 = BLUE<br>56 = MAGENTA<br>64 = TANGERINE<br>72 = PINK<br>80 = PURPLE<br>88 = WHITE<br>96 = BLACK<br>104 = BLUE<br>112 = MAUVE<br>120 = BLACK |    |    |   | U<br>N<br>D<br>E<br>R<br>L<br>I<br>N<br>E | B<br>L<br>I<br>N<br>K | I<br>N<br>V<br>E<br>R<br>S<br>E |
| MSB                                       |  |    |    |   |   |                       | LSB                             |
| 128                                       | 64   | 32 | 16 | 8 | 4   | 2                     | 1                               |

Figure 11-2. Attribute Byte

When you want to send attributes to control a display, embedded within the text, add their binary values and send them over the HP-IB to the instrument. The embedded attributes can be inhibited with the :ATTRIBUTE:DISABLE command and enabled with the :ATTRIBUTE:ENABLE command. Attributes affect the text sent to the display of the instrument when you use the DISPLAY subsystem LINE or STRING commands, or the DSP SYSTEM subsystem command.

## ATTRibute

---

The four color bits (8,16,32 and 64 in the figure above) when taken by themselves can represent the values 0 through 15. These are the same as the 16 values available with the COLOR command and in the front panel Color Cal menu.

The following example causes "HELLO" to be written in red at the current ROW and COLUMN of the display. The word HELLO will be displayed in red inverse video and will blink.

```
OUTPUT 707;":DISPLAY:TEXT BLANK; ATTRIBUTE ENABLE"  
OUTPUT 707 USING "14A,B,6A";":DISP:STRING """,128 + 16 + 2 + 1,"HELLO"""
```

The :DISPLAY:TEXT BLANK command erases the text displayed in the user area of the display, and the :DISPLAY:ATTRIBUTE ENABLE command enables the attributes.

The second line in the example contains some formatting. (See Appendix C.) The 14A enables the 14 character ":DISP:STRING" to be sent, the B changes the attribute value (147) to an ASCII character. The 6A enables six characters of string data "Hello" to be sent to the instrument. The 128 indicates that this is an attribute byte, 16 indicates the color red, 2 the blink attribute, and 1 the invert attribute. This information could just as easily have been output to the instrument as "147."

If you want to send attributes that are not embedded in the text, the following method can be used. In this case a color number must be sent with the :DISP:COLO command, not a color attribute value.

```
OUTPUT 707;":DISP:TEXT BLAN; ATTR ENAB; COLO 2; BLIN ON; INV ON"  
OUTPUT 707;":DISP:STRING ""HELLO"""
```

---

**BLINK****command/query**

The BLINK command determines whether text sent with the DSP SYSTEM subsystem command, or the LINE or STRING command in the DISPLAY subsystem is to be written with the BLINK attribute. If the blink attribute is on, the text will flash on and off when displayed.

The BLINK query returns the state of the BLINK attribute.

**Command Syntax:** :DISPlay:BLINk {{ON | 1} | {OFF | 0}}

**Example:** OUTPUT 707;":DISPLAY:BLINK ON"

**Query Syntax:** :DISPlay:BLINK?

**Returned Format:** [:DISPlay:BLINK] <state> <NL>  
<state> :: = {1 | 0}

**Example:** OUTPUT 707;":DISPLAY:BLINK?"  
ENTER 707;Blink\$  
PRINT Blink\$

## BRIghtness

---

### BRIghtness

### command/query

The BRIGHTNESS command specifies whether text sent with the DSP SYSTEM subsystem command, or the LINE or STRING commands in the DISPLAY subsystem are to be displayed in beige or gray. LOW provides gray text (color 0) and HIGH provides beige text (color 1).

The BRIGHTNESS query returns the state of the BRIGHTNESS attribute. This command overrides previous COLOR commands.

#### Note

*For more choice of colors use the :DISPLAY:COLOR command.*

**Command Syntax:** :DISPlay:BRIghtness {LOW | HIGH}

**Example:** OUTPUT 707;";DISPLAY:BRIGHTNESS LOW"

**Query Syntax:** :DISPlay:BRIghtness?

**Returned Format:** [:DISPlay:BRIghtness] < state > < NL >  
< state > :: = {LOW | HIGH}

**Example:** DIM Bright\$[30]  
OUTPUT 707;";DISPLAY:BRIGHTNESS?"  
ENTER 707;Bright\$  
PRINT Bright\$



**COLOR****command/query**

The COLOR command specifies what color the text will be when sent with the DSP SYSTEM subsystem command or the LINE or STRING commands in the DISPLAY subsystem. This command overrides a previous BRIGHTNESS command.

The COLOR query returns the value of the currently specified COLOR attribute.

**Note**

*The short form for this command is COLO, convention would have it as COL. However, COL is the short form for COLumn.*

**Command Syntax:** :DISPlay:COLOr <number>  
<number> :: = 0 to 15, where

- 0 :: = beige, highlighted text
- 1 :: = gray, text fields and graticule
- 2 :: = red, advisories and errors
- 3 :: = yellow, channel 1 waveforms and function 1
- 4 :: = green, channel 2 waveforms and function 2
- 5 :: = orange, markers
- 6 :: = blue, stored waveforms
- 7 :: = magenta, 2 trace overlap
- 8 :: = tangerine, channel 3
- 9 :: = pink, channel 4
- 10 :: = purple, Chan 1 for HP 3630A
- 11 :: = white background for HP 3630A
- 12 :: = black, overlap for HP 3630A
- 13 :: = blue, not used by system, available for user
- 14 :: = mauve, not used by system, available for user
- 15 :: = black, background

The colors shown are the default colors. These colors can be set as the user desires either with the :DISPLAY:SETCOLOR command or the front-panel Color Cal menu.

## COLOR

---

Example: OUTPUT 707;":DISPLAY:COLOR 2"

**Query Syntax:** :DISPlay:COLOr?

**Returned Format:** [:DISPlay:COLOr] <number> <NL>  
<number> :: = 0 to 15  
(integer - NR1 format)

Example: DIM Color\${30}  
OUTPUT 707;":DISPLAY:COLOR?"  
ENTER 707;Color\$  
PRINT Color\$

---

**COLumn****command/query**

The 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>  
                          <number> :: = 0 through 71

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

**Query Syntax:**   :DISPlay:COLumn?

**Returned Format:**   [:DISPlay:COLumn] <number> <NL>  
                          <number> :: = 0 through 71  
                          (integer - NR1 format)

**Example:**   DIM Column\$[30]  
              OUTPUT 707;":DISPLAY:COLUMN?"  
              ENTER 707;Column\$  
              PRINT Column\$

## DATA

---

### DATA

### command/query

The DATA command is used to write waveform data to or from one of the nine pixel memories in the HP 54121T. The pixel memories available are memories 0 through 8. They are specified by the :DISPLAY:SOURCE command with PMEMORY0 through PMEMORY8.

The DATA query causes the HP 54121T to output pixel data from the specified memory. If plane 0 is specified, the HP 54121T will transfer the active display. In all other cases the specified memory will be transferred.

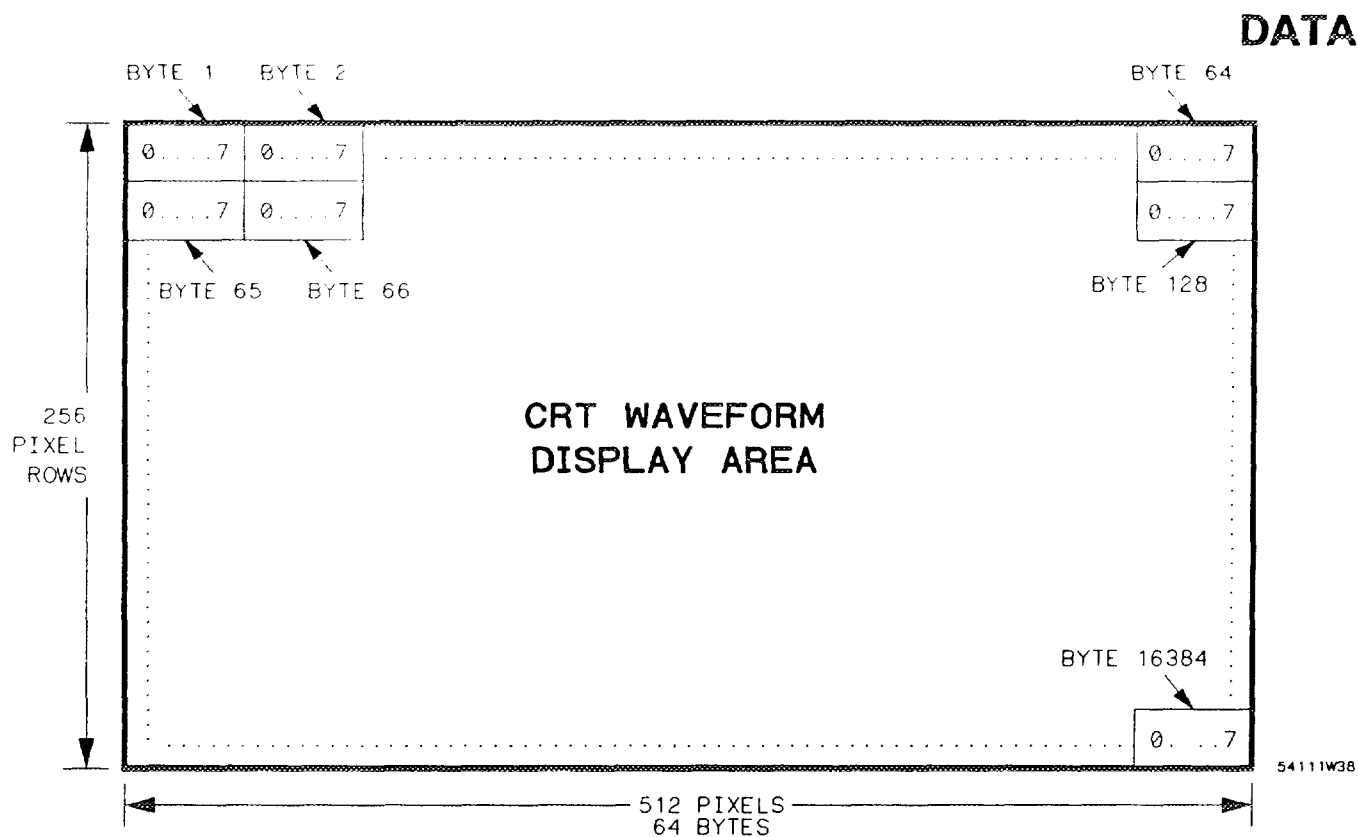
The DATA command is followed by a block of binary data that is transferred from the controller to a specific memory in the HP 54121T. If PMEMORY0 is specified, that data will be transferred to the channel 1 memory. In all other cases the data will be transferred to the specified memory.

The data is in the IEEE 488.2 definite block form with 16384 bytes of data preceded by seven block header bytes.

The data block contains: # 516384 < data bytes >

### Pixel Format

When you use the :DISPLAY:DATA command, the information is output as bytes of pixel data. The CRT is 512 pixel columns by 256 pixel rows. Each pixel row is divided into 64 bytes of eight bits each. Refer to figure 11-3, CRT Pixel Format.



*Figure 11-3. CRT Pixel Format*

### Data Output Format

The pixel memories are 16,384 bytes representing the waveform portion of the display. The waveform portion of the display is the portion of the display from, and including the top graticule line, to and including the bottom graticule line, as well as all information inside the graticule. The first 64 bytes of data that are output correspond to the top row of the waveform display (figure 11-3). The next 64 bytes that are output are the second row of the waveform display. The third 64 bytes that are output correspond to the third row, etc. The data is output in rows until the 256th pixel row of data is finally sent. The 256th pixel row corresponds to the bottom graticule line of the waveform display area.

## DATA

---

Each of the pixel rows contains 512 pixels that are sent in 8-bit bytes, and 256 rows of pixel data are sent. Some calculations at this point will show the relationship of the numbers. Each row is sent as 64 bytes, therefore  $64 \text{ bytes} \times 8 \text{ bits} = 512 \text{ pixels per row}$ . 64 bytes of data are sent for each row and 256 rows of data are sent therefore  $64 \text{ bytes} \times 256 \text{ rows} = 16,384 \text{ bytes sent for each screen of data}$ .

**Command Syntax:** :DISPlay:DATA < binary block >

**Query Syntax:** :DISPlay:DATA?

**Returned Format:** [:DISPlay:DATA] # 516384 < 16384 bytes of binary data >

**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 USING " # , - K";";DISPLAY:SOURCE PMEM1;DATA"
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 54121T.

---

**FORMat****command/query**

The **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 screen mode to **DUAL** 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] <mode> <NL>  
                          <mode> :: = {1 | 2 | 4}

**Example:**       DIM format\$[30]  
                  OUTPUT 707;":DISPLAY:FORMAT?"  
                  ENTER 707;Format\$  
                  PRINT Format\$

# GRATicule

---

## GRATicule

## command/query

The 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>  
<type> :: = {OFF | GRID | AXES | FRAME}

**Example:** DIM Grat\$[30]  
OUTPUT 707;":DISPLAY:GRATICULE?"  
ENTER 707;Grat\$  
PRINT Grat\$



---

**INVerse****command/query**

The INVERSE command determines whether text sent with the DSP SYSTEM subsystem command or 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 responds with the on or off 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] <state> <NL>  
<state> :: = {1 | 0}

**Example:** DIM Inv\$[30]  
OUTPUT 707;":DISP:INVERSE?"  
ENTER 707;Inv\$  
PRINT Inv\$

## LINE

---

## LINE

## command

The 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 the :DISPLAY:LINE command being sent. Text may be written up through column 61 with the LINE command. If the characters in the text string do not fill the line, the rest of the line is blanked. If the text string is longer than the available space on the current line, the excess characters will be discarded. 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 21, the last line in the display area, ROW is reset to 0.

**Command Syntax:** DISPlay:LINE <quoted string>  
<quoted string> :: = any series of ASCII characters enclosed in quotes.

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

---

**MASK****command/query**

The 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 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 one enables writing to the area.

The MASK query returns the current value of the MASK.

**Command Syntax:**   :DISPlay:MASK <value>  
                          <value> :: = 0 to 255  
                          (integer - NR1 format)

**Example:**   OUTPUT 707;":DISPLAY:MASK 254" ! Inhibits advisories only

**Query Syntax:**   :DISPlay:MASK?

**Return Format:**   [:DISPlay:MASK] <value> <NL>  
                          <value> :: = 0 to 255  
                          (integer - NR1 format)

**Example:**   DIM Mask\$(30)  
              OUTPUT 707;":DISP:MASK?"  
              ENTER 707;Mask\$  
              PRINT Mask\$

# MASK

---

*Table 11-1. Display Mask Byte*

| Bit | Mask Weight | Screen Area Affected  |
|-----|-------------|---|
| 7   | 128         | Function Softkey Underlines   |
| 6   | 64          | Function Softkeys - softkey labels on the right side of the display (rows 0-17, columns 62-71)      |
| 5   | 32          | Menu Selection Softkeys - text on the bottom line of the display (row 22, columns 0-71)             |
| 4   | 16          | Parameter Values - text below the graticule (rows 18-21, columns 0-71)                              |
| 3   | 8           | Graticule Labels - text inside the graticule (rows 2-17, columns 0-61)                              |
| 2   | 4           | Value Label - displays value of selected knob function (row 1 columns 19-61)                        |
| 1   | 2           | Status Line - status information on the first two lines (row 0 columns 0-71 and row 1 columns 0-18) |
| 0   | 1           | Advisory - Advisory and Error messages appear on row 15, columns 0-61.                              |

## PERSistence

## command/query

The PERSISTENCE command sets the display persistence. The PERSISTENCE command is only effective in the PERSISTENCE display mode, selected in the ACQUIRE subsystem. The parameters for this command are the keyword INFINITE or a real number from 0.3 through 11.0 representing the persistence in seconds. Any value greater than 10 seconds will set the PERSISTENCE to infinite.

The PERSISTENCE query returns the current persistence value.

**Command Syntax:** :DISPlay:PERSistence {INFINITE | 0.3 through 11}

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

**Query Syntax:** :DISPlay:PERSistence?

**Returned Format:** [:DISPlay:PERSistence] <value> <NL>  
<value> :: = 3.000E-01 to 1.100E + 1  
where 1.100E + 1 = infinite  
(exponential - NR3 format)

**Example:** OUTPUT 707;":DISPLAY:PERSISTENCE?"  
ENTER 707;Pers  
PRINT Pers

# PRiority

---

## PRiority

## command/query

The PRIORITY command sets the priority on or off for the subsequent DSP SYSTEM subsystem command and the LINE or STRING commands in the DISPLAY subsystem. The priority determines whether text or graphics will have priority over each other. When PRIORITY is ON, the text field overwrites the displayed signal(s) and graticule. When PRIORITY is OFF, the displayed signal(s) and graticule overwrite any text in the waveform display area.

The PRIORITY query returns the state of the PRIORITY command.

**Command Syntax:** :DISPlay:PRiority {{ON | 1} | {OFF | 0}}

**Example:** OUTPUT 707;":DISPLAY:PRIORITY OFF"

**Query Syntax:** :DISPlay:PRiority?

**Returned Format:** [:DISPlay:PRiority] <state> <NL>  
<state> :: = {1 | 0}

**Example:** DIM Pri\$[30]  
OUTPUT 707;":DISPLAY:PRIORITY?"  
ENTER 707;Pri\$  
PRINT Pri\$

---

**ROW****command/query**

The 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 21.

The ROW query returns the current value of ROW.

**Command Syntax:**   :DISPlay:ROW <row number>  
                          <row number> :: = 0 through 21

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

**Query Syntax:**   :DISPLAY:ROW?

**Returned Format:**   [:DISPlay:ROW] <row number> <NL>  
                          <row number> :: = 0 through 21  
                          (integer - NR1 format)

**Example:**    DIM Row\$[30]  
                OUTPUT 707;":DISPLAY:ROW?"  
                ENTER 707;Row\$  
                PRINT Row\$

# SETColor

---

## SETColor

## command/query

The SETCOLOR command allows you to change any of the color selections on the CRT. All of the color selections may be changed over the HP-IB one at a time. This command has four parameters: color number, hue, saturation, and luminosity.

The hue portion of this command allows you to determine the gradation of color. Hue can have a value of 0 to 100. As the hue number increases, the selected color cycles through the color spectrum. There is no difference between hue 0 and hue 100.

The saturation portion of this command allows you to choose the percentage of the pure color that is mixed with white. The acceptable values for saturation are 0 to 100, where 0 is white and 100 is maximum saturation of the chosen hue.

The luminosity portion of this command determines the brightness of the chosen hue. The acceptable values for luminosity are 0 to 100 where 0 is black and 100 is maximum brightness.

The :DISPLAY:SETCOLOR command followed by the keyword DEFAULT sets all colors to the default settings.

The SETCOLOR query returns the specified color number, hue saturation, and luminosity. Refer to the *Front-Panel Operation Reference* manual for more information on color.

Color values are not reset with a \*RST command.

**Command Syntax:** :DISPlay:SETColor { <color> , <hue> , <sat> , <lum> | DEFault }

Where:

<color> :: = integer from 0 to 15  
<hue> :: = integer from 0 to 100  
<sat> :: = integer from 0 to 100  
<lum> :: = integer from 0 to 100



**Example:**    OUTPUT 707;":DISPLAY:SETColor 0,0,100,50"

**Query Syntax:**    :DISPlay:SETColor? <color>  
                         <color> :: = color number 0 through 15

**Returned Format:**    [:DISPlay:SETColor] <color>,<hue>,<sat>,<lum> <NL>  
                         <color> :: = integer from 0 to 15  
                         <hue> :: = integer from 0 to 100  
                         <sat> :: = integer from 0 to 100  
                         <lum> :: = integer from 0 to 100

**Example:**    DIM Set\$(35)  
                 OUTPUT 707;":DISPLAY:SETCOLOR? 2"  
                 ENTER 707;Set\$  
                 PRINT Set\$

## SOURce

### SOURce

### command/query

The SOURCE command specifies the source or destination for the :DISPLAY:DATA query and command. The SOURCE command has one parameter, PMEMORY0 through PMEMORY8.

The SOURCE query returns the currently specified SOURCE.

**Command Syntax:** :DISPlay:SOURce PMEMORY {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8}

WHERE:

PMEMORY0 :: = active display  
PMEMORY1 :: = pixel memory 1  
PMEMORY2 :: = pixel memory 2  
PMEMORY3 :: = graticule and markers  
PMEMORY4 :: = displayed stored waveforms and markers  
PMEMORY5 :: = channel 1  
PMEMORY6 :: = channel 2  
PMEMORY7 :: = channel 3  
PMEMORY8 :: = channel 4

**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 USING "#,-K";"DISPLAY:SOURCE PMEM1;DATA" 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 54121T.

**Query Syntax:** :DISPlay:SOURce?

**Returned Format:** [:DISPlay:SOURce] PMEMORY {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8} <NL>

**Example:**

```
DIM Src$(30)
OUTPUT 707;";DISP:SOUR?"
ENTER 707;Src$
PRINT Src$
```

---

**STRing****command**

The **STRING** command writes a text string to the CRT of the HP 54121T. The text will be written starting at the current **ROW** and **COLUMN** values. If the column limit is reached (column 71) the excess text is discarded. 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

---

## TEXT

## command

The TEXT command allows you to erase the user text area on the CRT. The user text area is rows 2 through 17, columns 0 through 61 and rows 18 through 21, columns 0 through 71. 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 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] <state> <NL>  
<state> :: = {1 | 0}

**Example:** DIM Tmar\${30}  
OUTPUT 707;":DISP:TMARker?"  
ENTER 707;Tmar\$  
PRINT Tmar\$

## UNDERline

---

### UNDERline

### command/query

The UNDERLINE command determines whether text sent with the DSP SYSTEM subsystem command or the LINE or STRING command in the DISPLAY subsystem is to be written with the UNDERLINE attribute. If the UNDERLINE attribute is on, the text will be underlined on the display.

The UNDERLINE query returns the state of the UNDERLINE attribute.

**Command Syntax:** :DISPlay:UNDErline {{ON | 1} | {OFF | 0}}

**Example:** OUTPUT 707;":DISP:UNDERLINE ON"

**Query Syntax:** :DISPlay:UNDErline?

**Returned Format:** [:DISPlay:UNDErline] <state> <NL>  
<state> :: = {1 | 0}

**Example:** DIM Under\${30}  
OUTPUT 707;":DISP:UNDERLINE?"  
ENTER 707;Under\$  
PRINT Under\$

---

**VMARker****command/query**

The 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] <state> <NL>  
<state> :: = { 1 | 0 }

**Example:** DIM Vmark\${30}  
OUTPUT 707;":DISP:VMARKER?"  
ENTER 707;Vmark\$  
PRINT Vmark\$

**Note**

*It is recommended practice to turn the Vmarkers on before attempting to use them with any Measure Subsystem command.*





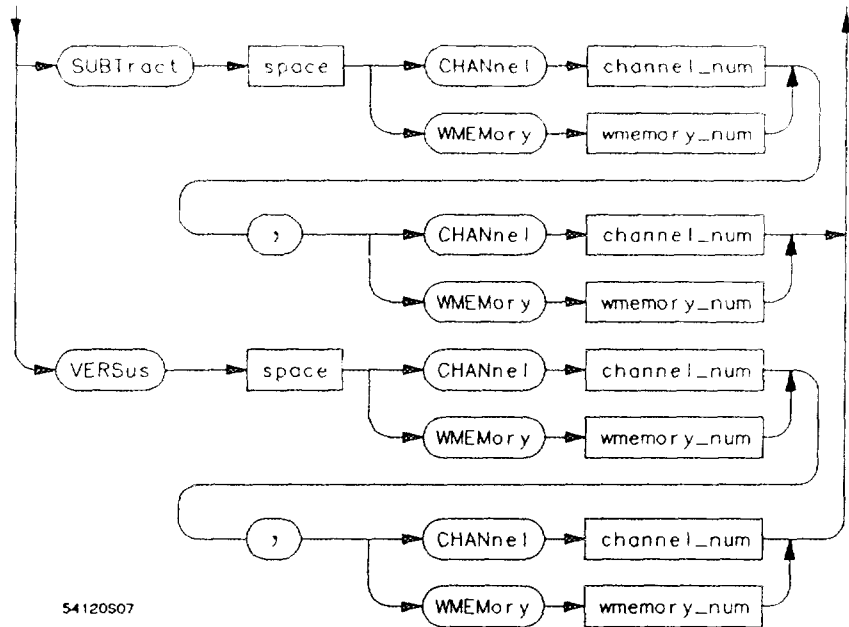
## Introduction

The FUNCTION subsystem defines seven functions by using the displayed channels and/or the waveform memories as operands. The operators are ADD, SUBTRACT, ONLY, VERSUS, MIN, MAX, and INVERT. See figure 12-1 for a syntax diagram of the FUNCTION subsystem commands.

There are two Function subsystems. The subsystem is entered with the :FUNCTION < N > command, where < N > is 1 or 2.

Memories 1 through 4 are available for functions. The MIN and MAX operators function only in the persistence display mode.





54120507

**channel\_num** = 1, 2, 3, or 4

**function\_num** = 1 or 2

**offset\_arg** = initial offset + or – maximum volts full scale for the function

**range\_arg** = 0.008 V to 0.640 V when a 1:1 probe attenuation and single operand functions are used

**wmemory\_num** = 1, 2, 3 or 4

*Figure 12-1. Function <N> Subsystems Syntax Diagram (continued)*

## ADD

---

### ADD

### command

The ADD command algebraically sums the two defined operands.

**Command Syntax:** :FUNCTION<N>:ADD <operand1>,<operand2>  
<N> :: = 1 or 2  
<operand 1,2> :: = {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 |  
WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

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

---

**INVert****command**

The INVERT command inverts the operand.

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

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

# MAX

---

## MAX

## command

The MAX command accumulates the maximum value of the operand. This function is only available in the persistence display mode.

For best results, make sure the persistence is not set to infinite.

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

**Example:** OUTPUT 707;":FUNCTION2:MAX WMEMORY3"

---

**MIN****command**

The MIN command accumulates the minimum values of the operand. This function is available only in the persistence display mode.

For best results, make sure the persistence is not set to infinite.

**Command Syntax:** :FUNCTION<N>:MIN <operand>

<N> :: = 1 or 2

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

**Example:** OUTPUT 707;":FUNCTION2:MIN CHANNEL3"

## OFFSet

## OFFSet

## command/query

The OFFSET command sets the voltage represented at center screen for the selected function. The maximum value of offset is initial offset  $\pm$  volts full scale, when volts full scale is at maximum for the function.

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

**Command Syntax:** :FUNCTION <N>:OFFSet <offset>  
<N> :: = 1 or 2  
<offset> :: =  $\pm$  volts full screen

**Example:** OUTPUT 707;":FUNCTION1:OFFSET 650E-4"  
OUTPUT 707;":FUNC1:OFFS .065" OUTPUT 707;":FUNC1:OFFS 65 mV"

### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :FUNCTION <N>:OFFSet?  
<N> :: = 1 or 2

**Returned Format:** [:FUNCTION <N>:OFFSet] <offset> <NL>  
<N> :: = 1 or 2  
<offset> :: =  $\pm$  volts full screen  
(exponential - NR3 format)

**Example:** OUTPUT 707;":FUNCTION2:OFFSET?"  
ENTER 707;Offset  
PRINT Offset



---

**ONLY****command**

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

**Example:** OUTPUT 707,":FUNCTION2:ONLY WMEMORY4"

# RANGe

## RANGe

command/query

The RANGE command defines the full-scale vertical axis of the selected function. The RANGE can be set to any value from 8 mV to 640 mV, when you are using 1:1 probe attenuation and single operand functions. When the functions operand is ADD or SUBTRACT, the legal range is the range of operand 1 plus the range of operand 2. See the VERSUS operand in this section for a description of how RANGE works for VERSUS. 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 function.

**Command Syntax:** :FUNCTION<N>:RANGe <range>  
<N> :: = 1 or 2  
<range> :: = .008 V to .640 V  
when a 1:1 probe attenuation and  
single channel operand functions are being used

Example: OUTPUT 707,":FUNCTION2:RANGE 40 MV"

### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :FUNCTION<N>:RANGe?  
<N> :: = 1 or 2

**Returned Format:** [:FUNCTION<N>:RANGe] <range> <NL>  
<N> :: = 1 or 2  
<range> :: = .008 V to .640 V or  
when a 1:1 probe attenuation and single channel operand functions are being used  
(exponential - NR3 format)

Example: OUTPUT 707,":FUNCTION2:RANGE?"  
ENTER 707;Range  
PRINT Range

---

**SUBTract****command**

The SUBTRACT command algebraically subtracts operand 2 from operand 1.

**Command Syntax:** :FUNCTION <N>:SUBTract <operand1>,<operand2>  
<N> :: = 1 or 2  
<operand 1,2> :: = {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

---

## VERSus

## command

This 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.

**Command Syntax:** **:FUNCTION<N>:VERSus <operand1>,<operand2>**  
**<N> :: = 1 or 2**  
**<operand> :: = {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}**

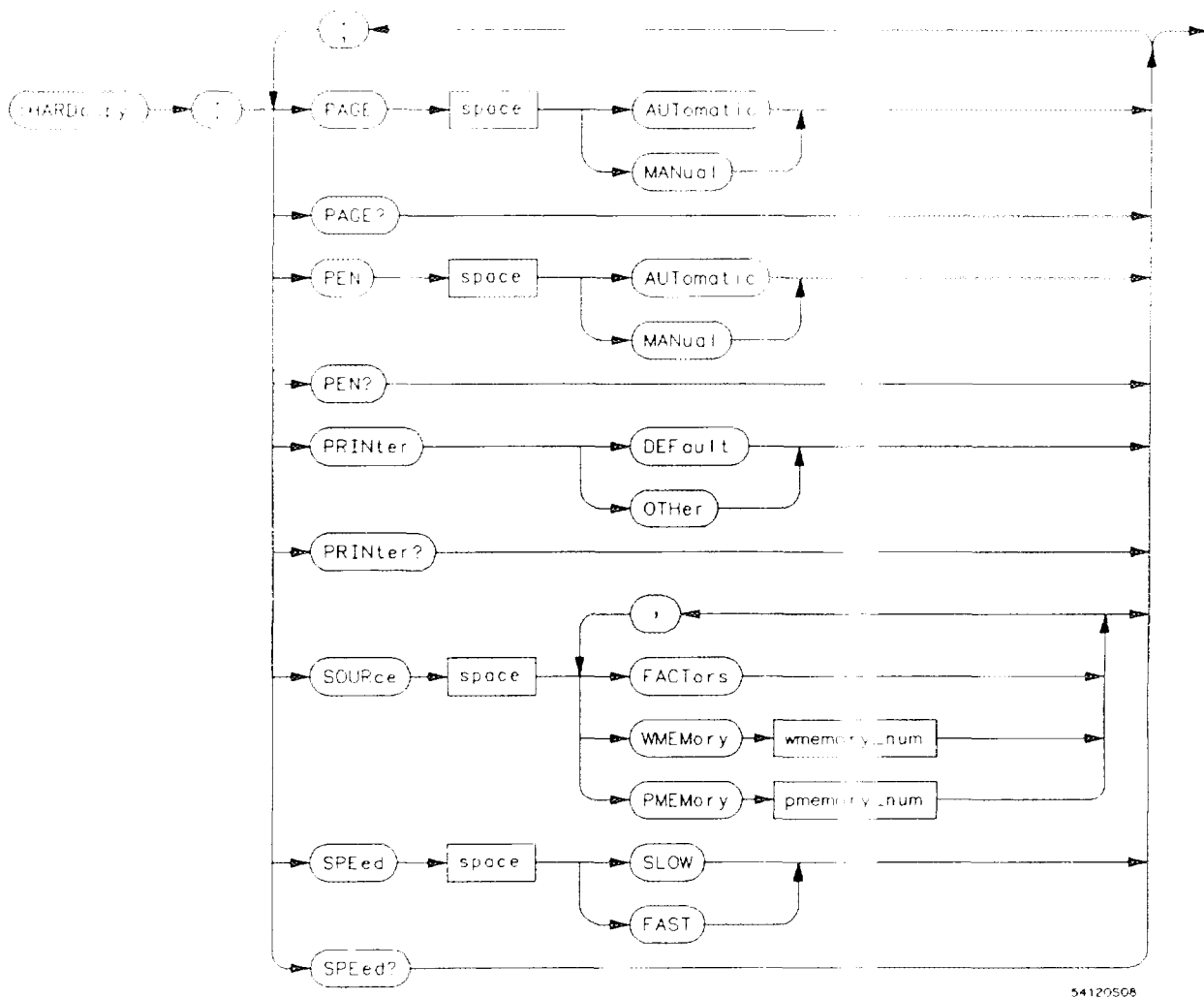
**Example:** **OUTPUT 707;":FUNCTION2:VERSUS CHAN1,CHAN4"**

## Introduction

The HARDCOPY subsystem commands set various parameters for plotting and printing waveforms from the HP 54121T. Displayed waveforms, graticules, voltage and time markers, and waveform factors can be copied to an output device. The portion of the waveform to be copied must be placed on the display.

To actually make the hardcopy print or plot, refer to the root level commands :PRINT? and :PLOT? for the sequence of bus commands that gets the data to the printer or plotter.

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



**wmemory\_num** = 1, 2, 3, or 4

**pmemory\_num** = 0, 1 or 2

*Figure 13-1. Hardcopy Subsystem Syntax Diagram*

## PAGE

---

## PAGE

command/query

The PAGE command sets up the HP 54121T to send a form feed after a hardcopy output to a printer.

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] <state> <NL>  
<state> :: = {MANual | AUTomatic}

**Example:** DIM Page\${30}  
OUTPUT 707;":HARDCOPY:PAGE?"  
ENTER 707;Page\$  
PRINT Page\$

## PEN

## command/query

The PEN command sets the pen control function of the HP 54121T. When this command is set to AUTOMATIC, the HP 54121T will instruct the plotter to get a different pen for different parts of the drawing. Up to six pens can be used for different parts of the drawing.

When the PEN command is set to AUTOMATIC, the HP 54121T assigns the following pen numbers to the following parts of the drawing:

| Pen | # Parts of the Drawing  |
|-----|---|
| 1   | Graticule, timebase factors, and channel 3 and its associated factors |
| 2   | Channel 1, function 1 and their associated factors                    |
| 3   | All waveform memories and their associated factors                    |
| 4   | Channel 2, function 2 and their associated factors                    |
| 5   | Markers   |
| 6   | Channel 4 and its associated factors                                  |

When the PEN command is set to the MANUAL mode and a plot is requested, the plotter will not be instructed to select a pen. On completion of the plot, an instruction will be sent to the plotter to put away the pen.

The PEN query returns the state of the pen control command.



## PEN

---

**Command Syntax:** :HARDcopy:PEN {MANual | AUTomatic}

**Example:** OUTPUT 707;":HARDCOPY:PEN AUTOMATIC"

**Query Syntax:** :HARDcopy:PEN?

**Returned Format:** [:HARDcopy:PEN] <state> <NL>  
<state> :: = {MANual | AUTomatic}

**Example:** DIM Pen\${30}  
OUTPUT 707;":HARDCOPY:PEN?"  
ENTER 707;Pen\$  
PRINT Pen\$

**PRINter****command/query**

The PRINter command specifies the type of printer that will be used as the output device.

When DEFAULT is selected the HP 2225A is the active printer, and when OTHER is selected the HP 3630A color printer is active.

The PRINter query returns the current printer type.

**Command Syntax:** :HARDcopy:PRINter {DEFault | OTHer}

**Example:** OUTPUT 707;":HARD:PRINT OTH"

**Query Syntax:** :HARDcopy:PRINter?

**Returned Format:** [:HARDcopy:PRINter] {DEFault | OTHer} <NL>

**Example:** DIM Print\${30}  
OUTPUT 707;":HARD:PRINTER?"  
ENTER 707;Print\$  
PRINT Print\$

## SOURce

## SOURce

command

The SOURCE command specifies the source(s) to be output during the printing or plotting of a hardcopy.

There is no query form of this command.

< source-id > :    PMEMory0 = active display  
                     PMEMory1 = pixel memory 1  
   (same as front panel PMEM5)  
                     PMEMory2 = pixel memory 2  
   (same as front panel PMEM6)  
                     FACTors = scale factors  
                     WMEMory1 = waveform memory 1  
                     WMEMory2 = waveform memory 2  
                     WMEMory3 = waveform memory 3  
                     WMEMory4 = waveform memory 4

**Command Syntax:**    HARDcopy:SOURce < source1 > , [ < source2 > , ... ]  
                         < source1,2,... > :: = { PMEMory { 0 | 1 | 2 } | FACTors | { WMEMory1 | 2 | 3 | 4 } }

**Example:**    OUTPUT 707;":HARD:SOUR FACT,WMEM1,WMEM3"

### Note

*In order to print histogram data, display WMEMORY5 then use PMEMORY0 as the hardcopy source.*

---

**SPEed****command/query**

The SPEED command specifies the pen speed for plotting. FAST is for normal paper and SLOW is for plotting transparencies.

The SPEED query returns the current pen speed.

**Command Syntax:** :HARDcopy:SPEed {SLOW | FAST}

**Example:** OUTPUT 707;":HARD:SPEED FAST"

**Query Syntax:** :HARDcopy:SPEed?

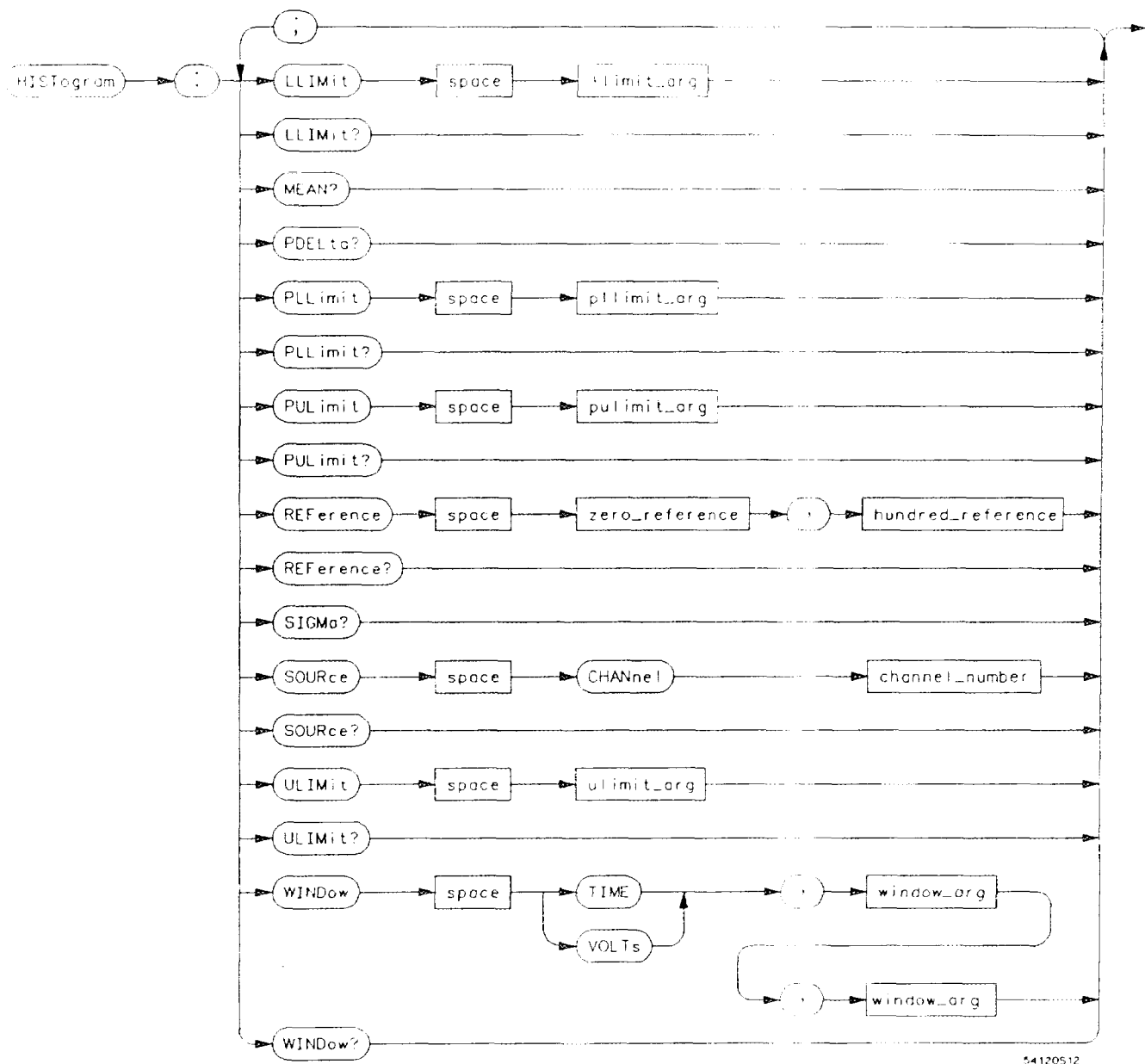
**Returned Format:** [:HARDcopy:SPEed] <state> <NL>  
<state> :: = {SLOW | FAST}

**Example:** DIM Speed\$[30]  
OUTPUT 707;":HARD:SPE?"  
ENTER 707;Speed\$  
PRINT Speed\$

## Introduction

The HISTOGRAM subsystem allows you to set voltage or time windows and calculate histograms within them. It also allows you to set markers on the resulting histogram. See the Histogram Subsystem of the *Front-Panel Operation Reference* manual for details on setting up the histogram. Histograms over the Bus Follow these steps to do a histogram over the bus:

1. Select the desired bandwidth with ACQUIRE:BANDWIDTH
2. Select the desired source with HISTOGRAM:SOURCE.
3. Determine the window type and size with HISTOGRAM:WINDOW
4. Select the type of acquisition with ACQUIRE:TYPE HISTOGRAM and the number of samples with ACQUIRE:COUNT.
5. Digitize the channel selected in step 1. This puts the histogram data into WMEMORY5, which is used only for histograms.
6. Turn on the histogram display with the :VIEW WMEM5 command.
7. Now you can set markers and compute mean and sigma (standard deviation).



54120512

Figure 14-1. Histogram Subsystem Syntax Diagram

**channel\_number** = 1, 2, 3, or 4.

**hundred\_reference** = The 100 percent time or voltage value. The hundred\_reference value must be on screen and there must be data between the zero\_reference and the hundred\_reference.

**llimit\_arg** = any on screen time or voltage value.

**pllimit\_arg** = any on screen percentage.

**pulimit\_arg** = any on screen percentage.

**ulimit\_arg** = any on screen time or voltage value.

**window\_arg** = any on screen time or voltage value.

**zero\_reference** = The zero percent time or voltage value. The zero\_reference value must be on screen and there must be data between the zero\_reference and the hundred\_reference.

*Figure 14-1. Histogram Subsystem Syntax Diagram (continued)*

## LLIMit

---

### LLIMit

### command/query

The LLIMIT command allows you to set the lower limit marker to the specified time or voltage on the histogram.

The LLIMIT query returns the current position of the lower limit marker in units of time or volts.

**Command Syntax:** :HISTogram:LLIMit <sec-or-volts>  
<sec-or-volts> :: = a time or voltage value within screen limits

**Example:** OUTPUT 707;":HISTOGRAM:LLIMIT 32.0 MV" (voltage histogram)  
OUTPUT 707;":HISTOGRAM:LLIM 17e-9" (time histogram)

**Query Syntax:** :HISTogram:LLIMit?

**Returned Format:** [:HISTogram:LLIMit] <sec-or-volts> <NL>

**Example:** DIM Llim\$(30)  
OUTPUT 707;":HISTOGRAM:LLIMIT?"  
ENTER 707;Llim\$  
PRINT Llim\$



---

**MEAN?****query**

The MEAN query computes the statistical mean on the histogram and returns the value.

**Query Syntax:** :HISTogram:MEAN?

**Returned Format:** [:HISTogram:MEAN] < mean > < NL >  
< mean > :: = statistical mean time or voltage

**Example:** DIM Mean\$(30)  
OUTPUT 707;":HISTOGRAM:MEAN?"  
ENTER 707;Mean\$  
PRINT Mean\$

## PDELta?

---

## PDELta?

query

The PDELTA query returns the difference in cumulative percentage between the upper limit and lower limit markers.

**Query Syntax:** :HISTogram:PDELta?

**Returned Format:** [:HISTogram:PDELta] <pdelta> <NL>  
<pdelta> :: = cumulative percentage difference between markers

**Example:** DIM Pdel\$[30]  
OUTPUT 707;":HISTOGRAM:PDELTA?"  
ENTER 707;Pdel\$  
PRINT Pdel\$

## PLLimit

## command/query

The PLLIMIT command allows you to set the lower limit marker to the specified cumulative percentage on the histogram. The range of valid percentages depends on the reference setting but will not be set off screen.

The PLLIMIT query returns the current position of the lower limit marker for the cumulative percentage.

**Command Syntax:** :HISTogram:PLLimit <value>  
<value> :: = cumulative percentage

**Example:** OUTPUT 707;":HIST:PLL 80"

**Query Syntax:** :HISTogram:PLLimit?

**Returned Format:** [:HISTogram:PLLimit] <value> <NL>  
<value> :: = cumulative percentage

**Example:** OUTPUT 707;":HISTOGRAM:PLLIMIT?"  
ENTER 707;PII  
PRINT PII

## PULimit

---

### PULimit

### command/query

The PULIMIT command allows you to set the upper limit marker to the specified cumulative percentage on the histogram. The range of valid percentages depends on the reference setting but will not be set off screen.

The PULIMIT query returns the current position of the upper limit marker for the cumulative percentage.

**Command Syntax:** :HISTogram:PULimit <value>  
<value> :: = cumulative percentage

**Example:** OUTPUT 707;":HIST:PUL 80"

**Query Syntax:** :HISTogram:PULimit?

**Returned Format:** [:HISTogram:PULimit] <value> <NL>  
<value> :: = cumulative percentage  
(exponential - NR3 format)

**Example:** OUTPUT 707;":HISTOGRAM:PULIMIT?"  
ENTER 707;PII  
PRINT PII

## REFERENCE

command/query

The REFERENCE command specifies the pair of values inside the histogram window that define the 0% and 100% cumulative percentage. The arguments are in units of time or volts. The valid range is dependent on the histogram window, but will not be off screen.

**Note**

*If there are no histogram data points between the zero and 100 percent reference value, the reference will not be changed and an error will be generated.*

The REFERENCE query returns the position of the reference settings in time or volts.

**Command Syntax:** :HISTogram:REFerence <0% reference> , <100% reference>  
<0% reference> :: = any on screen time or voltage  
<100% reference> :: = any on screen time or voltage

**Example:** OUTPUT 707,":HISTOGRAM:REFERENCE - 10.0 MV, 15.5 MV"

OUTPUT 707,":HISTOGRAM:REFERENCE 20.0 ns, 25.5 ns"

**Query Syntax:** :HISTogram:REFerence?

**Returned Format:** [:HISTogram:REFerence] <0% reference> , <100% reference> <NL>

**Example:**

OUTPUT 707,":HISTOGRAM:REF?"  
ENTER 707;Zero,Hundred  
PRINT Zero,Hundred

## SIGMa?

---

## SIGMa?

query

The SIGMA query computes the standard deviation and mean on the histogram in either time or volts and returns the standard deviation.

**Query Syntax:** :HISTogram:SIGMa?

**Returned Format:** [:HISTogram:SIGMa] <sigma> <NL>

**Example:** DIM Sigma\$[30]  
OUTPUT 707,":HISTOGRAM:SIGM?"  
ENTER 707;Sigma\$  
PRINT Sigma\$

---

**SOURce****command/query**

The **SOURCE** command determines which channel will be used for the histogram acquisition.

The **SOURCE** query returns the current histogram acquisition channel.

The digitize channel argument and the histogram source argument must be the same.

**Command Syntax:** :HISTogram:SOURce CHANnel {1 | 2 | 3 | 4}

Example: OUTPUT 707;":HISTOGRAM:SOURCE CHANNEL4"

**Query Syntax:** :HISTogram:SOURce?

**Returned Format:** [:HISTogram:SOURce] CHANnel {1 | 2 | 3 | 4} <NL>

Example: DIM Sour\${50}  
OUTPUT 707;":HIST:SOUR?"  
ENTER 707;Sour\$  
PRINT Sour\$

## ULIMit

---

### ULIMit

### command/query

The ULIMIT command allows you to set the upper limit marker to the specified time or volts on the histogram.

The ULIMIT query returns the current position of the upper limit marker in units of time or vol's.

**Command Syntax:** :HISTogram:ULIMit <sec-or-volts>  
<sec-or-volts> :: = from LLIMIT

**Example:** OUTPUT 707;":HISTOGRAM:ULIMIT 32.0 MV"  
OUTPUT 707;":HISTOGRAM:ULIM 17e-9"

**Query Syntax:** :HISTogram:ULIMit?

**Returned Format:** [:HISTogram:ULIMit] <sec-or-volts> <NL>

**Example:** DIM Llim\$[30]  
OUTPUT 707;":HISTOGRAM:ULIMIT?"  
ENTER 707;Llim\$  
PRINT Llim\$



## WINDow

## command/query

The WINDOW command determines whether the histogram is accumulated from a time or voltage window, and it sets the lower and upper window limits. The valid range for the arguments is any on-screen time or voltage value.

The WINDOW query returns the type of window and its size.

**Command Syntax:** :HISTogram:WINDow {TIME | VOLTs}, <sec-or-volts> , <sec-or-volts>  
<sec-or-volts> :: = any on screen value

**Example:** OUTPUT 707;":HISTOGRAM:WINDOW TIME,1.6e-8,1.8e-8" (time window/voltage histogram)

OUTPUT 707;":HISTOGRAM:WINDOW VOLTS, -.25,.25" (voltage window/time histogram)

**Query Syntax:** :HISTogram:WINDow?

**Returned Format:** [:HISTogram:WINDow] {TIME | VOLTs}, <sec-or-volt> , <sec-or-volt> <NL>

**Example:** DIM Wind\$[10]  
OUTPUT 707;":HIST:WIND?"  
ENTER 707;Wind\$,Lower,Upper  
PRINT Wind\$,Lower,Upper

—

—)

—)

—

## 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 defined events. An example of a defined event is the time when a waveform crosses a specified voltage level (TVOLT? query). In this case, the time at the voltage crossing is determined and reported to the controller.

---

## Measurement Setup

For the most accuracy, measurements should be made in the :ACQUIRE:TYPE AVERAGE and in fine precision mode. Measurements must be made with the portion of the waveform required for that measurement 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.

It is recommended that you turn the voltage and time markers on with the :DISPLAY:VMARKER and :DISPLAY:TMARKER commands prior to making any measurements.

---

## Invalid Measurement

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

---

## 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. The 50% voltage value is used for measuring frequency, period, pulse width, and duty cycle. Period is used to measure rms voltage.

Voltage measurements are made with 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 `VMarker1` and `VMarker2` 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 `VMarker1` and the start marker are assigned to the first specified source and `VMarker2` and the stop marker are assigned to the second specified source. The measurements available using two sources are: `ESTART`, `ESTOP`, `TSTART`, `TSTOP`, `TDELTA`, `VSTART`, `VSTOP`, `VDELTA`, `VFIFTY`, `EDELTA`, and `CURSOR`.

Some 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 and the source specifier is changed to a single source.

Refer to figure 15-1 for the syntax diagram of the `MEASURE` subsystem.

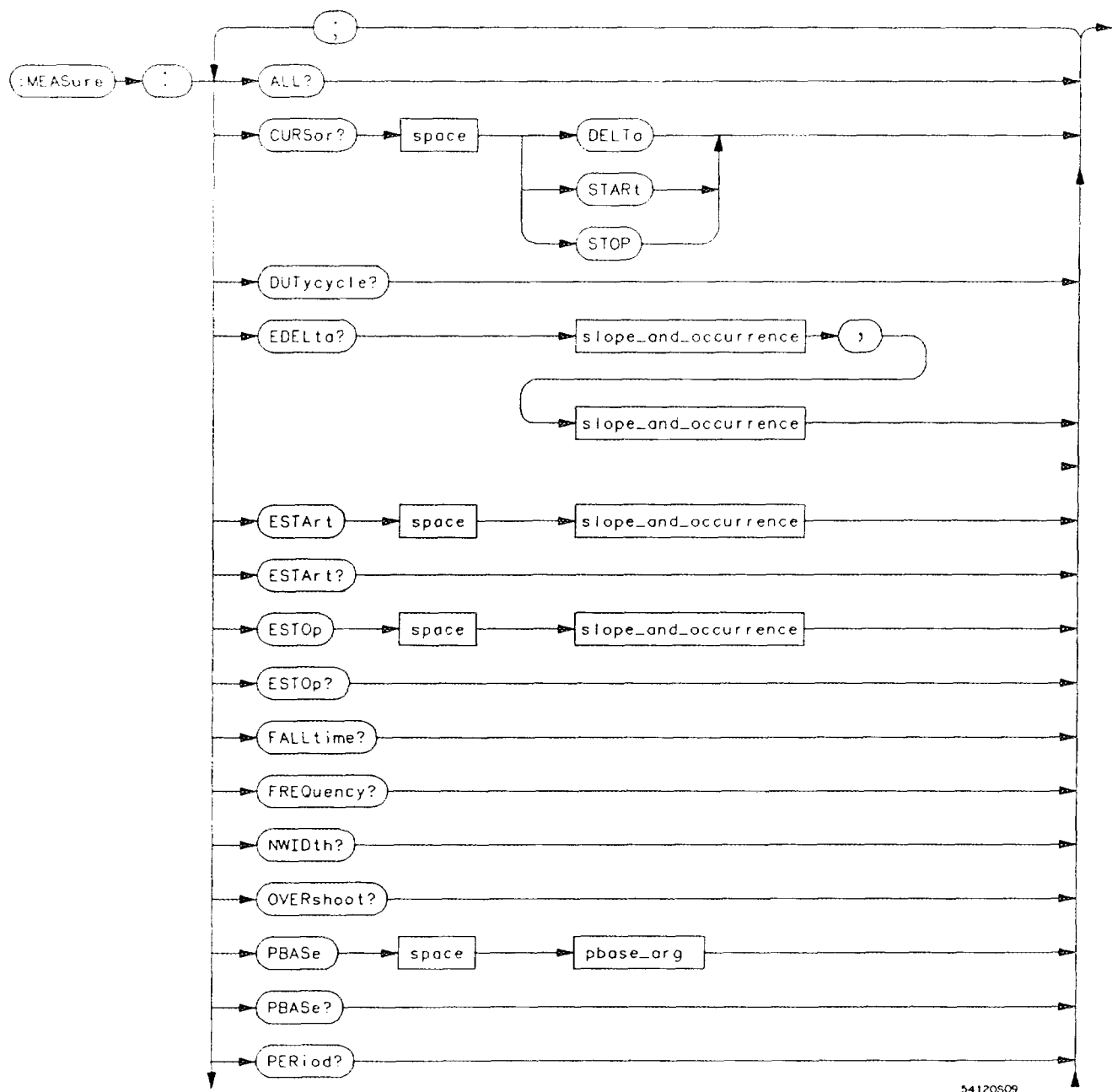
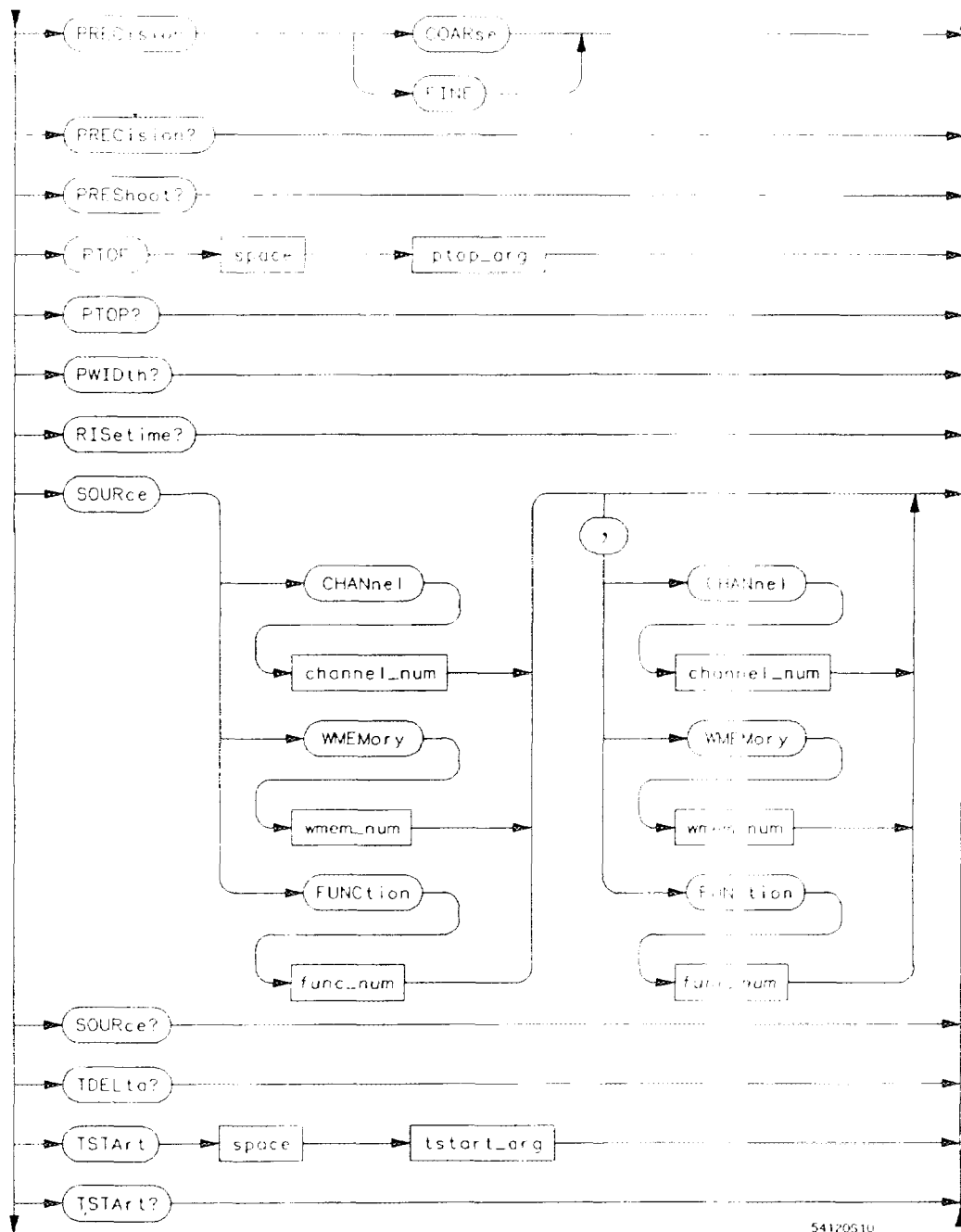
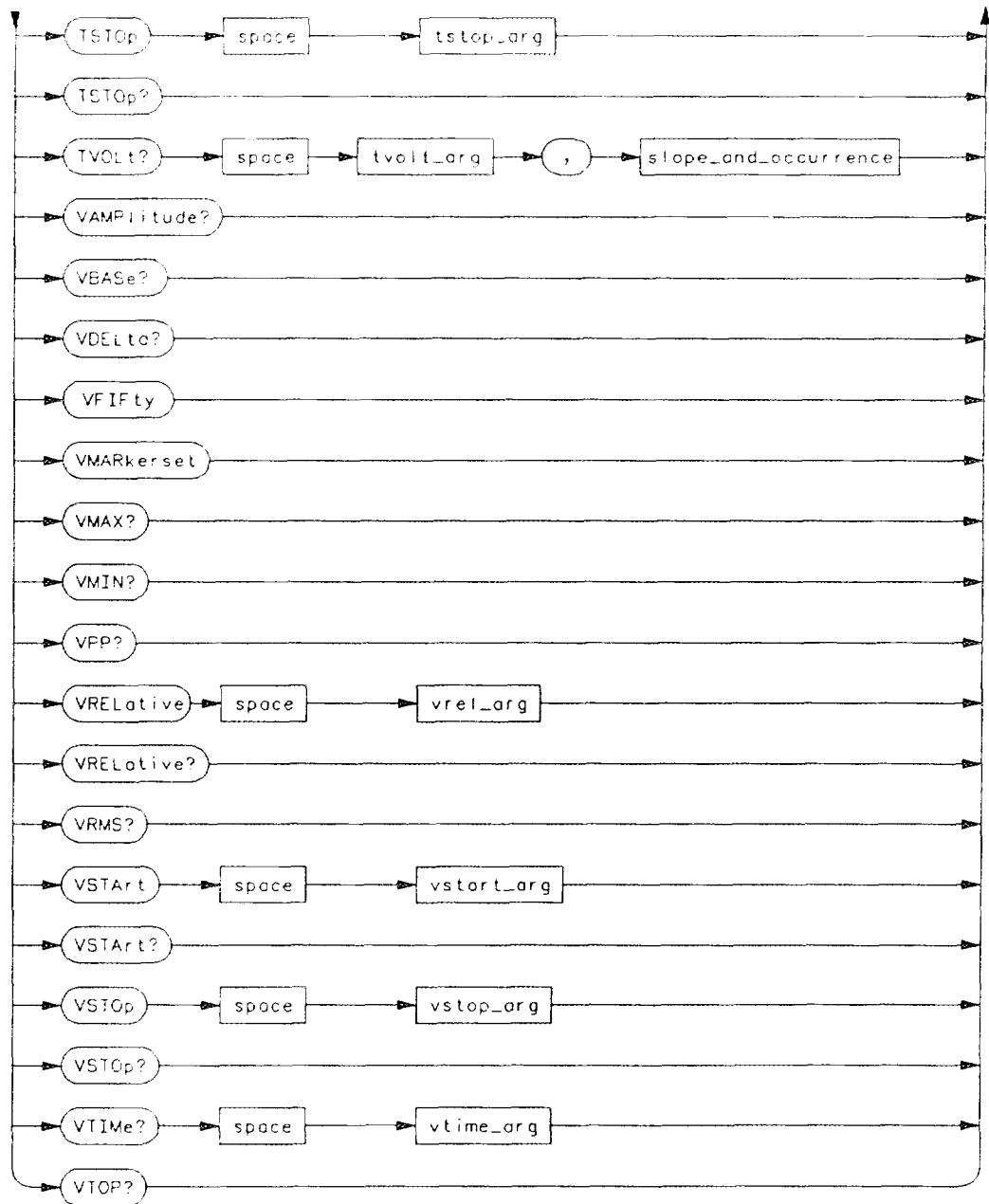


Figure 15-1. Measure Subsystem Syntax Diagram



54120510

Figure 15-1. Measure Subsystem Syntax Diagram (continued)



54120S11

Figure 15-1. Measure Subsystem Syntax Diagram (continued)

**channel\_num** = An integer, 1, 2, 3 or 4.

**func\_num** = An integer, 1 or 2.

**wmem\_num** = An integer, 1 through 4.

**pbase\_arg** = An integer, –25 through 125, specifying percentage.

**ptop\_arg** = An integer, –25 through 125, specifying percentage.

**slope\_and\_occurrence** = An integer, –100 to 100 (excluding 0) specifying an edge.

**tstart\_arg** = A real number. The range of this argument depends on the sweep speed.

**tstop\_arg** = A real number. The range of this argument depends on the sweep speed.

**tvolt\_arg** = A real number specifying voltage.

**vrel\_arg** = An integer, 0, 10, 20, or 50.

**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.

*Figure 15-1. Measure Subsystem Syntax Diagram (continued)*



## ALL?

## query

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

Measurements must be made with the portion of the waveform required for the measurement displayed. Measurements are made on the first (left-most) displayed edges of the waveforms. For the most accurate measurement possible, use fine precision mode.

If a parameter cannot be measured, the instrument responds with +9.99999E+37 for that measurement result. All parameters that can be measured will have the correct value as the measurement result.

Refer to the individual commands for information on how the measurements are made and the returned format of the measurement results.

This query returns multiple <response message unit> s.

**Query Syntax:** :MEASure:ALL?

**Returned Format:** [:MEASure:FREQuency] <NR3> ; [PERiod] <NR3> ; [PWIDTH] <NR3> ; [NWIDTH] <NR3> ; [RISetime] <NR3> ; [FALLtime] <NR3> ; [VAMplitude] <NR3> ; [VPP] <NR3> ; [PREShoot] <NR3> ; [OVERshoot] <NR3> ; [DUTYcycle] <NR3> ; [VRMS] <NR3> ; [VMAX] <NR3> ; [VMIN] <NR3> ; [VTOP] <NR3> ; [VBASe] <NR3> <NL>

**Eample:** DIM AII\$[500]  
OUTPUT 707;":MEASURE:ALL?"  
ENTER 707;"AII\$"  
PRINT AII\$

**Note**

*These values can be returned to numeric variables instead of the string variable as shown. If numeric variables are used, the headers should be turned off. Refer to Appendix B, Program Examples, for using numeric variables.*

## CURSor?

---

## CURSor?

## query

The CURSOR query returns the time and voltage values of the specified marker as an ordered pair of time and 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>

<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

---

**DUTYcycle?****query**

The DUTYCYCLE query measures and outputs the duty cycle of the displayed signal. The value returned for duty cycle is the ratio of positive pulse width to the period of the displayed signal. The positive pulse width and the period of the displayed signal are measured, then the duty cycle is calculated. The duty cycle is calculated with the following formula:

$$\text{duty cycle} = + \text{ pulse width/period}$$

**Query Syntax:** :MEASure:DUTYcycle?

**Returned Format:** [MEASure:DUTYcycle] <value> <NL>  
<value> ::= ratio of + pulse width to period  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:DUTYCYCLE?"  
ENTER 707;Dut  
PRINT Dut

## EDELta?

---

## EDELta?

## query

The EDELTA query causes the instrument to position the start marker and the stop marker on the specified edges and slope. The first parameter sets the start marker and the second parameter sets the stop marker. If a positive value is sent for the parameter, that marker will be placed on a positive edge. If a negative value is sent for the parameter, the marker is placed on a negative waveform edge. The start marker is placed where VMarker1 intersects the waveform and the stop marker is placed where VMarker2 intersects the waveform.

If the time markers do not intersect the waveform(s) as specified, the error message "Edges required for measurement not found" is displayed on the CRT.

**Query Syntax:** :MEASure:EDELta? <slope & occurrence> , <slope & occurrence>

<slope & occurrence> :: = sign and edge number  
(if a positive value is sent, the + sign may be omitted or a space may be used)

**Returned Format:** :MEASURE:EDELta] <value> <NL>

<value> :: = time difference between start and stop markers  
(real - NR3 format)

**Example:** OUTPUT 707;":MEASURE:EDELTA? 2, -2"  
ENTER 707;Edelta  
PRINT Edelta

This example places the start marker at the second positive-going intersection of the waveform and VMarker 1, and the stop marker at the second negative-going intersection of the waveform and VMarker 2.

## ESTArt

## command/query

The ESTART command causes the instrument to position the start marker on the specified edge and slope of the displayed waveform. 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, it will place the start marker on a negative-going waveform edge.

If VMarker 1 does not intersect the waveform as specified, the error message, "Edges required for measurement not found," is displayed.

The ESTART query responds with the currently specified edge.

**Command Syntax:** :MEASure:ESTArt <edge>  
 <edge> :: = sign and number  
 (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 positive-going intersection of the waveform and VMarker 1.

**Query Syntax:** :MEASure:ESTArt?

**Returned Format:** [:MEASure:ESTART] <edge> <NL>  
 <edge> :: = edge number  
 (integer - NR1 format)

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

# ESTOp

## ESTOp

## command/query

The ESTOP command causes the instrument to position the stop marker on the specified edge and slope of the displayed waveform. 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, it places the stop marker on a negative-going waveform edge.

If VMarker 2 does not intersect the waveform as specified, the error message "Edges required for measurement not found" is displayed.

The ESTOP query responds with the currently specified edge.

**Command Syntax:** :MEASure:ESTOp <edge>  
<edge> :: = sign and number  
(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 negative-going intersection of the waveform and VMarker 2.

**Query Syntax:** :MEASure:ESTOp?

**Returned Format:** [:MEASure:ESTOP] <edge> <NL>  
<edge> :: = edge number  
(integer - NR1 format)

**Example:** OUTPUT 707;":MEASURE:ESTOP?"  
ENTER 707;Estop  
PRINT Estop

---

**FALLtime?****query**

The FALLTIME query causes the instrument to measure and output the fall time of the first displayed falling edge. Falltime is measured between the 10% and 90% points of the falling (negative-going) edge.

For the most accurate measurement, set :MEASURE:PRECISION to FINE.

The fall time is calculated with the following formula:

$$\text{fall time} = \text{time at 10\% point} - \text{time at 90\% point.}$$

**Query Syntax:** :MEASure:FALLtime?

**Returned Format:** [:MEASure:FALLtime] <value> <NL>  
<value> :: = time in seconds between 10% and 90% voltage points  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:FALLTIME?"  
ENTER 707;Fall  
PRINT Fall

## FREQuency?

---

### FREQuency?

query

The FREQUENCY query measures and outputs the frequency of the first complete cycle on screen using the 50% levels.

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)
```

**Query Syntax:** :MEASURE:FREQuency?

**Returned Format:** [:MEASure:FREQuency] <value> <NL>  
<value> ::= frequency in Hertz  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:FREQUENCY?"  
ENTER 707;Freq  
PRINT Freq



---

**NWIDth?****query**

The NWIDTH query measures and outputs the width of the first negative pulse on screen using the 50% levels.

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)

**Query Syntax:** :MEASure:NWIDth?

**Returned Format:** [:MEASure:NWIDth] <value> <NL>  
<value> :: = negative pulse width in seconds  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:NWIDTH?"  
ENTER 707;Nwid  
PRINT Nwid

## OVERshoot?

---

### OVERshoot?

query

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
```

**Query Syntax:** :MEASure:OVERshoot?

**Returned Format:** [:MEASure:OVERshoot] <value> <NL>  
<value> :: = ratio of overshoot to Vamplitude (exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:OVERSHOOT?"  
ENTER 707;Over  
PRINT Over

---

**PBASe****command/query**

The PBASE command sets a parameter to a percentage value. This parameter is used to set VMarker1 to a percentage of the value of the displayed signal. The marker is actually set to the proper percent value with the VMARKERSET command. The range of the argument for this command is an integer from -25 to 125.

**Note**

*When the PBASE command is sent, the marker will be set to a percentage of the last values.*

The PBASE query returns the current value of the PBASE setting.

**Command Syntax:** :MEASure:PBASe <value>  
<value> :: = -25 to 125 (in percent)

**Example:** OUTPUT 707;":MEAS:PBAS 10"

**Query Syntax:** :MEASure:PBASe?

**Returned Format:** [:MEASure:PBAS] <value> <NL>  
<value> :: = VMarker 1 setting in percent of top-base (integer - NR1 format)

**Example:** OUTPUT 707;":MEASURE:PBAS?"  
ENTER 707;Pbase  
PRINT Pbase

## PERiod?

---

## PERiod?

query

The PERIOD command measures and outputs the period of the first complete cycle on screen. The period is measured at the 50% voltage level of the waveform. 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)
```

**Query Syntax:** :MEASure:PERiod?

**Returned Format:** [:MEASure:PERiod] <value> <NL>  
<value> :: = waveform period in seconds  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:PERIOD?"  
ENTER 707;Period  
PRINT Period

---

**PRECision****command/query**

The PRECISION command allows you to specify the precision for subsequent measurements. When PRECISION is set to FINE, the edges for making a measurement are evaluated by increasing the sweep speed until the edge has a slope of approximately 45 degrees or the limit of the horizontal system has been reached. This increases the time resolution of the measurement. When PRECISION is set to COARSE, no horizontal expansion is accomplished.

**Note**

*Waveform Memories and Functions that use waveform memory operands cannot be expanded in time.*

The PRECISION query returns the present selection.

**Command Syntax:** :MEASure:PRECision {COARse | FINE}

**Example:** OUTPUT 707;":MEAS:PREC FINE"

**Query Syntax:** MEASure:PRECision?

**Returned format:** [:MEASure:PRECision] {COARse | FINE} <NL>

**Example:** DIM Pre\$[50]  
OUTPUT 707;":MEASURE:PRECISION?"  
ENTER 707;Pre\$  
PRINT Pre\$

## PREShoot?

---

### PREShoot?

query

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
```

**Query Syntax:** :MEASure:PREShoot?

**Returned Format:** [:MEASure:PREShoot] <value> <NL>  
<value> :: = ratio of preshoot to Vamplitude  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:PRESHOOT?"  
ENTER 707;Pres  
PRINT Pres

---

**PTOP****command/query**

The PTOP command sets a parameter to a percentage value. This parameter is used to set the VMarker2 to a percentage of the value of the displayed signal. The marker is actually set to the proper percent value with the VMARKERSET command. The range for this command is an integer from - 25 to 125 percent.

**Note**

*When the PTOP command is sent, the marker will be set to a percentage of the last values.*

The PTOP query returns the current value of the PTOP setting.

**Command Syntax:** :MEASure:PTOP <value>

<value> :: = - 25 to 125 (in percent)

**Example:** OUTPUT 707;":MEAS:PTOP 95"

**Query Syntax:** :MEASure:PTOP?

**Returned Format:** [:MEASure:PTOP] <value> <NL>

<value> :: = VMarker 2 setting in percent of top-base  
(integer - NR1 format)

**Example:** OUTPUT 707;":MEAS:PTOP?"  
ENTER 707;Ptop  
PRINT Ptop

## PWIDth?

## PWIDth?

query

The PWIDTH query measures and outputs the width of the first displayed positive pulse. Pulse width is measured at the 50% voltage level. 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)**

**Query Syntax:** :MEASure:PWIDth?

**Returned Format:** [:MEASure:PWIDth] <value> <NL>  
<value> :: = width of positive pulse in seconds  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:PWIDTh?"  
ENTER 707;Pwid  
PRINT Pwid



---

**RISetime?****query**

The RISETIME query measures and outputs the risetime of the first displayed rising (positive-going) edge. For the most accurate measurement possible, set PRECISION to FINE or set the sweep speed as fast as possible while leaving the leading edge of the waveform on the display. The risetime is determined by measuring the time at the 10% and 90% voltage points on the rising edge, then the risetime is calculated with the formula:

$$\text{rise time} = (\text{time at 90\% point} - \text{time at 10\% point})$$

**Query Syntax:** :MEASure:RISetime?

**Returned Format:** [:MEASure:RISetime] <value> <NL>  
<value> :: = rise time in seconds  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEAS:RIS?"  
ENTER 707;Rise  
PRINT Rise

## SOURce

---

### SOURce

### command/query

The SOURCE command selects the source(s) for the measurements. The source specified will become the source for the MEASURE subsystem commands.

For two source measurements, two parameters are specified after the source command. When two sources are specified, VMarker 1 and the start marker are assigned to the first specified source, and VMarker 2 and the stop marker are assigned to the second specified source.

The MEASURE commands that can be used when two sources are specified are: CURSOR, EDELTA, ESTART, ESTOP, TSTART, TSTOP, TDELTA, VSTART, VSTOP, VDELTA and VFIFTY.

From the front panel the measure, VMarker1, and VMarker2 sources can all be different. Under HP-IB control, the measure and VMarker1 sources will always agree. If the front panel is used to set the sources, the :MEASURE:SOURCE? query will return the measure and VMarker2 sources. When a measurement source is set that source will be turned on for display.

#### Note

*If two sources have been specified and a measurement is made with a command that can use only one source, then the measurement is made on the first specified source and the second source selection is deleted.*

The SOURCE query returns the current source selection.

**Command Syntax:** :MEASure:SOURce <source1> [, <source2> ]

<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>

<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\$

## TDELta?

---

### TDELta?

query

The TDELTA query returns the time difference between the start and stop time markers, that is:

$\text{delta} = T_{\text{stop}} - T_{\text{start}}$   
where  $T_{\text{start}}$  is the time at the start marker and  $T_{\text{stop}}$  is the time at the stop marker

**Query Syntax:** :MEASure:TDELta?

**Returned Format:** [:MEASure:TDELta] <value> <NL>  
<value> :: = difference between start and stop markers  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:TDELTA?"  
ENTER 707;Tdel  
PRINT Tdel

## TSTArt

## command/query

The TSTART command moves the start marker to the specified time with respect to the trigger time. The specified time must be a positive number. The minimum value is 16 ns. The maximum value is 1000 screen diameters or 10 seconds, whichever is less.

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>  
<start marker time> :: = time at start marker in seconds

**Example:** OUTPUT 707;":MEASURE:TSTA .00000003"  
OUTPUT 707;":MEASURE:TSTART 30 ns"  
OUTPUT 707;":MEAS:TSTA 3E-8"

**Note**

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :MEASure:TSTArt?

**Returned Format:** [:MEASure:TSTArt] <value> <NL>  
<value> :: = time at start marker in seconds  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:TSTART?"  
ENTER 707;Tsta  
PRINT Tsta

# TSTOp

## TSTOp

## command/query

The TSTOP command moves the stop marker to the specified time with respect to the trigger time. The specified time must be a positive number. The minimum value is 16 ns. The maximum value is 1000 screen diameters or 10 seconds, whichever is less.

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 >  
<stop marker time > :: = time at stop marker in seconds

**Example:** OUTPUT 707;":MEASURE:TSTOP .00000004"  
OUTPUT 707;":MEAS:TSTOP 40 ns"  
OUTPUT 707;":MEAS:TSTO 4E-8"

### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :MEASure:TSTOp?

**Returned Format:** [:MEASure:TSTOp] <value> <NL>  
<value> :: = time at stop marker in seconds  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:TSTOP?"  
ENTER 707;Tsto  
PRINT Tsto

## TVOLT?

query

When the 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 would happen if the waveform does not cross the specified voltage, or if the waveform does not cross the specified voltage 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 a 1, first crossing is reported; if a 2, second crossing is reported, etc.)

**Returned Format:** [:MEASure:TVOLT] <time> <NL>  
< time > :: = time in seconds of specified voltage crossing  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:TVOLT? -.250, +3"  
ENTER 707;Tvolt  
PRINT Tvolt

## VAMPlitude?

---

### VAMPlitude?

query

The VAMPLITUDE query returns the top-base amplitude 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 top-base value is calculated with the formula:

$$\text{Vamplitude} = \text{Vtop} - \text{Vbase}$$

#### Note

*For more information on how measurements are made refer to Appendix A.*

**Query Syntax:** :MEASure:VAMPlitude?

**Returned Format:** [:MEASure:VAMPlitude] <value> <NL>  
<value> :: = difference between top and base voltages - Delta V value  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:V AMPLITUDE?"  
ENTER 707;Vamp  
PRINT Vamp



---

**VBASe?****query**

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.

Refer to Appendix A for information on how voltage measurements are made.

**Query Syntax:** :MEASure:VBASe?

**Returned Format:** [:MEASure:VBASe] <value> <NL>  
<value> :: = voltage at base of selected waveform  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:VBASe?"  
ENTER 707;Base  
PRINT Base

## VDELta?

---

### VDELta?

query

The 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>  
<value> :: = Delta V value in volts  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEAS:VDELTA?"  
ENTER 707;Vdel  
PRINT Vdel

---

**VFIFty****command**

The 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 (VMarker1 and VMarker2) to the 50% voltage level on that source.

If two sources are specified with the source command, VMarker1 is set to the 50% level of the first specified source and VMarker2 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"

## VMARkerset

---

### VMARkerset

command

The VMARKERSET command causes the instrument to measure the top and base voltages of the displayed waveform, then moves both of the voltage markers (VMarker 1 and 2) to the values determined by the VRELATIVE command.

This function is similiar to the Auto Level Set key on the front panel.

**Command Syntax:** :MEASure:VMARkerset

**Example:** OUTPUT 707;":MEASURE:VMARKERSET"

---

**VMAX?****query**

The VMAX query measures and outputs the absolute maximum voltage present on the selected waveform.

Refer to Appendix A for information on how voltage measurements are made.

**Query Syntax:** :MEASure:VMAX?

**Returned Format:** [:MEASure:VMAX] <value> <NL>  
<value> :: = maximum voltage of selected waveform  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:VMAX?"  
ENTER 707;Vmax  
PRINT Vmax

## VMIN?

## VMIN?

query

The VMIN query measures and outputs the absolute minimum voltage present on the selected waveform. Refer to Appendix A for information on how voltage measurements are made.

**Query Syntax:** :MEASure:VMIN?

**Returned Format:** [:MEASure:VMIN] <value> <NL>

<value> :: = minimum voltage value of the selected waveform  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:VMIN?"  
ENTER 707;Vmin  
PRINT Vmin

## VPP?

query

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:

$$V_{pp} = V_{max} - V_{min}$$

where

Vmax and Vmin are the maximum and minimum voltages present on the selected source.

**Query Syntax:** :MEASure:VPP?

**Returned Format:** [:MEASure:VPP] <value> <NL>  
<value> :: = voltage peak to peak  
(exponential - NR3)

**Example:** OUTPUT 707;":MEAS:VPP?"  
ENTER 707;Vpeak  
PRINT Vpeak

## VRElative

---

### VRElative

### command/query

The 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.

It is recommended that the VMARKERSET command always be sent after the VRELATIVE command. This is a must if dual sources are specified.

For example, after a :MEAS:VMARKERSET command has been sent and :MEAS:VREL 0, VMarker 1 is located at the base (0%) of the signal and VMarker 2 at the top (100%) of the signal. If the VRELATIVE 10 command is executed, VMarker 1 is moved to the 10% level of the signal and VMarker 2 to the 90% level. VREL 0 moves the markers back to their original locations. VREL 50 moves both markers to the 50% point of their original positions. The values that can be used with this command are:

- 0 moves VMarker 1 to 0% and VMarker 2 to 100%
- 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%

The VRELATIVE query returns the current relative position of the markers.

#### Note

*The PTOP and PBASE commands are similar to the variable levels function on the front panel.*



**Command Syntax:** :MEASure:VRELative <percent>  
<percent> ::= {0 | 10 | 20 | 50}

**Example:** OUTPUT 707;":MEASURE:VRELATIVE 20"

**Query Syntax:** :MEASure:VRELative?

**Returned Format:** [:MEASure:VRELative] <value> <NL>  
<value> ::= marker position in percent {0 | 10 | 20 | 50}  
(integer - NR1 format)

**Example:** OUTPUT 707;":MEAS:VREL?"  
ENTER 707;Rel  
PRINT Rel

## VRMS?

---

## VRMS?

## query

The VRMS query measures and outputs the rms voltage of the selected waveform. The rms voltage is computed using all displayed data, but must be at least one complete cycle.

If a complete cycle is not present on the display, the error "Edges required for measurement not found" is displayed on the CRT and +9.99999E+37 is returned.

### Note

*Refer to Appendix A for information about how measurements are made.*

**Query Syntax:** :MEASure:VRMS?

**Returned Format:** [:MEASure:VRMS] <value> <NL>  
<value> :: = rms voltage of displayed waveform  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:VRMS?"  
ENTER 707;Vrms  
PRINT Vrms

## VSTArt

## command/query

The VSTART command moves VMarker 1 to the specified voltage. The values are limited to the currently defined channel, function, or memory range.

The maximum value for this command is  $-900\text{ mV}$  to  $+900\text{ mV}$  with 1:1 attenuation.

The VSTART query returns the current voltage level of VMarker 1.

**Command Syntax:** :MEASure:VSTArt <voltage> <NL>  
 <voltage> ::= between  $-900\text{ mV}$  and  $+900\text{ mV}$  with 1:1 attenuation

**Example:** OUTPUT 707;":MEAS:VSTART -.01"  
 OUTPUT 707;":MEASURE:VSTA -1E-2"  
 OUTPUT 707;":MEAS:VSTA -10MV"

**Note**

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :MEASure:VSTArt?

**Returned Format:** [:MEASure:VSTArt] <value> <NL>  
 <value> ::= voltage at VMarker 1  
 (exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:VSTART?"  
 ENTER 707;Vsta  
 PRINT Vsta

## VSTOp

## VSTOp

## command/query

The VSTOP command moves VMarker 2 to the specified voltage. The maximum value for the marker setting is  $-900$  mV to  $+900$  mV with 1:1 attenuation.

The VSTOP query returns the current voltage level of VMarker 2.

**Command Syntax:** :MEASure:VSTOp <voltage>  
<voltage> :: = between  $+900$  mV and  $-900$  mV with 1:1 attenuation

**Example:** OUTPUT 707;":MEAS:VSTOP -.1"  
OUTPUT 707;":MEASURE:VSTO -1E-1"  
OUTPUT 707;":MEAS:VSTO -100MV"

### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :MEASure:VSTOp?

**Returned Format:** [:MEASure:VSTOp] <value> <NL>  
<value> :: = voltage at VMarker 2  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:VSTOP?"  
ENTER 707;Vstop  
PRINT Vstop

## VTIMe?

query

The VTIMe query returns the voltage at the time specified by the query. The time is referenced to the trigger event, and the specified time must be on the screen. The time specified with the query must be a positive value greater than 16 ns. If the time specified with the query is off-screen, the error value  $+9.99999E+37$  will be returned.

If the specified source for the VTIMe query is a Waveform Memory or a Waveform Math Function with operands only of Waveform Memories, and the specified time point does not contain a voltage value, then the voltage value at the specified time will be linearly interpolated. The interpolation is accomplished using the last voltage value before the specified time and the first voltage value after the specified time.

If a Waveform Memory which contains no data is selected as the source or is an operand of a Waveform Math Function selected as the source, no voltage measurement can be made and the error value  $+9.99999E+37$  will be returned.

If a Waveform Math Function that contains no data is selected as the source, but has at least one channel as an operand, new waveform(s) will be acquired and the function will be recomputed.

If the source is a CHANNEL, this query can be used to make very fast voltage measurements. When the measurement precision selected is "coarse", the current voltage in the time point of interest will be returned. If the specified time point is empty, new data will be acquired at the specified time ONLY. No unnecessary data is acquired. If the measurement precision is "fine", then data is automatically reacquired at the specified time only. For coarse measurements which use existing data, the data is only guaranteed to have been taken at the time point on screen which contains the specified time. The width of a time point on screen varies with sweep speed. When new data is acquired, it is taken exactly at the specified time.

## VTIMe?

---

**Query Syntax:** :MEASure:VTIMe? <time>  
<time> :: = displayed time from trigger in seconds

**Returned Format:** [:MEASure:VTIMe] <value> <NL>  
<value> :: = voltage at specified time  
(exponential - NR3 format)

**Example:** OUTPUT 707,":MEASURE:VTIME? .001"  
ENTER 707;Vtim  
PRINT Vtim

---

**VTOP?****query**

The VTOP query returns the voltage at the top of a waveform.

**Note**

*Refer to Appendix A for information on how voltage measurements are made.*

**Query Syntax:** :MEASure:VTOP?

**Returned Format:** [:MEASure:VTOP] <value> <NL>  
<value> :: = voltage at top of waveform  
(exponential - NR3 format)

**Example:** OUTPUT 707;":MEASURE:VTOP?"  
ENTER 707;Vtop  
PRINT Vtop





# Network Subsystem

---

16

## Introduction

The NETWORK subsystem commands control the time domain reflectometry (TDR) and time domain transmission (TDT) functions of the HP 54121T.

The reflection and transmission paths must be calibrated before any time domain reflectometry or time domain transmission numeric answers can be requested or output.

See figure 16-1 for the syntax diagram of the NETWORK subsystem.

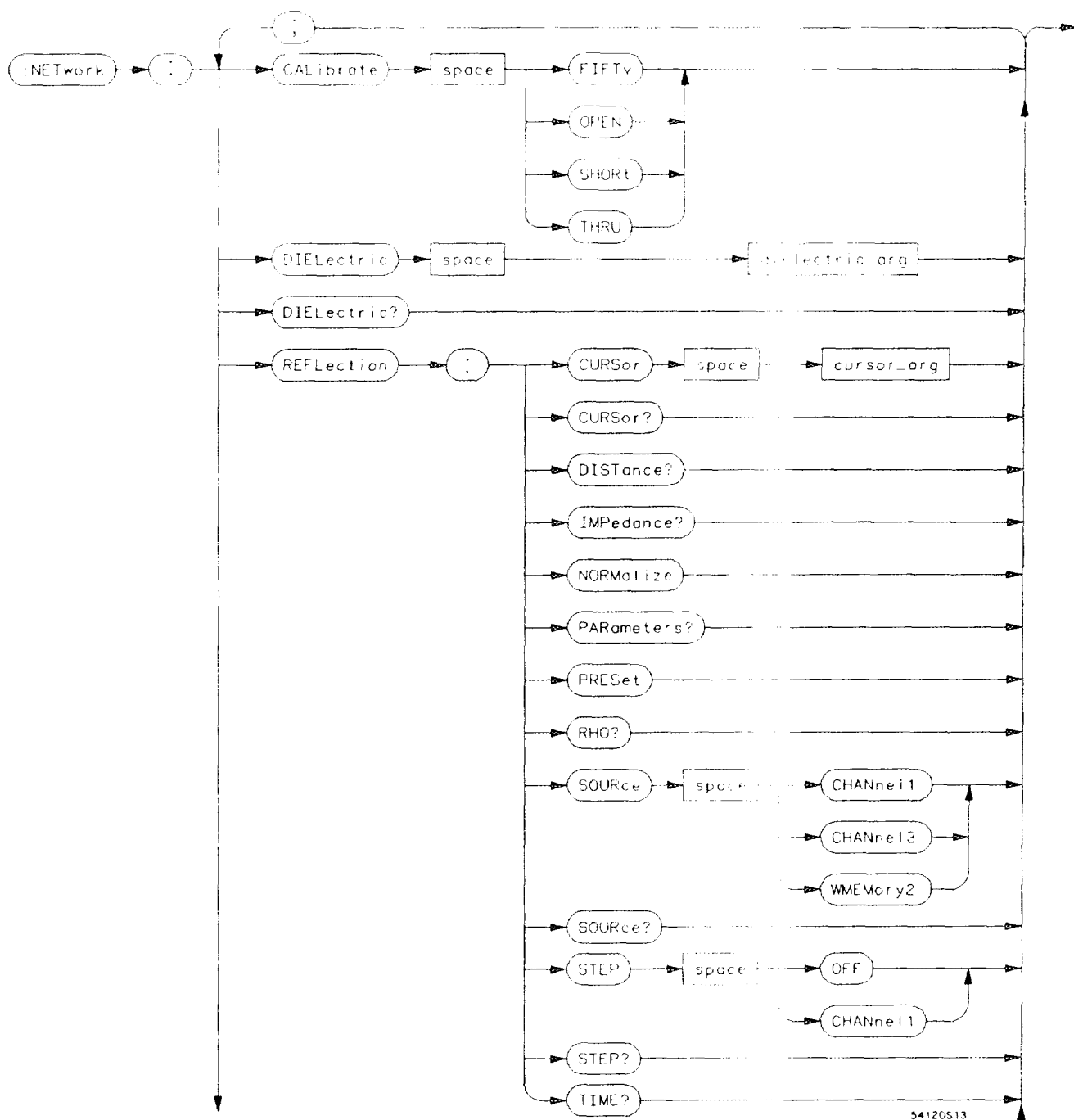
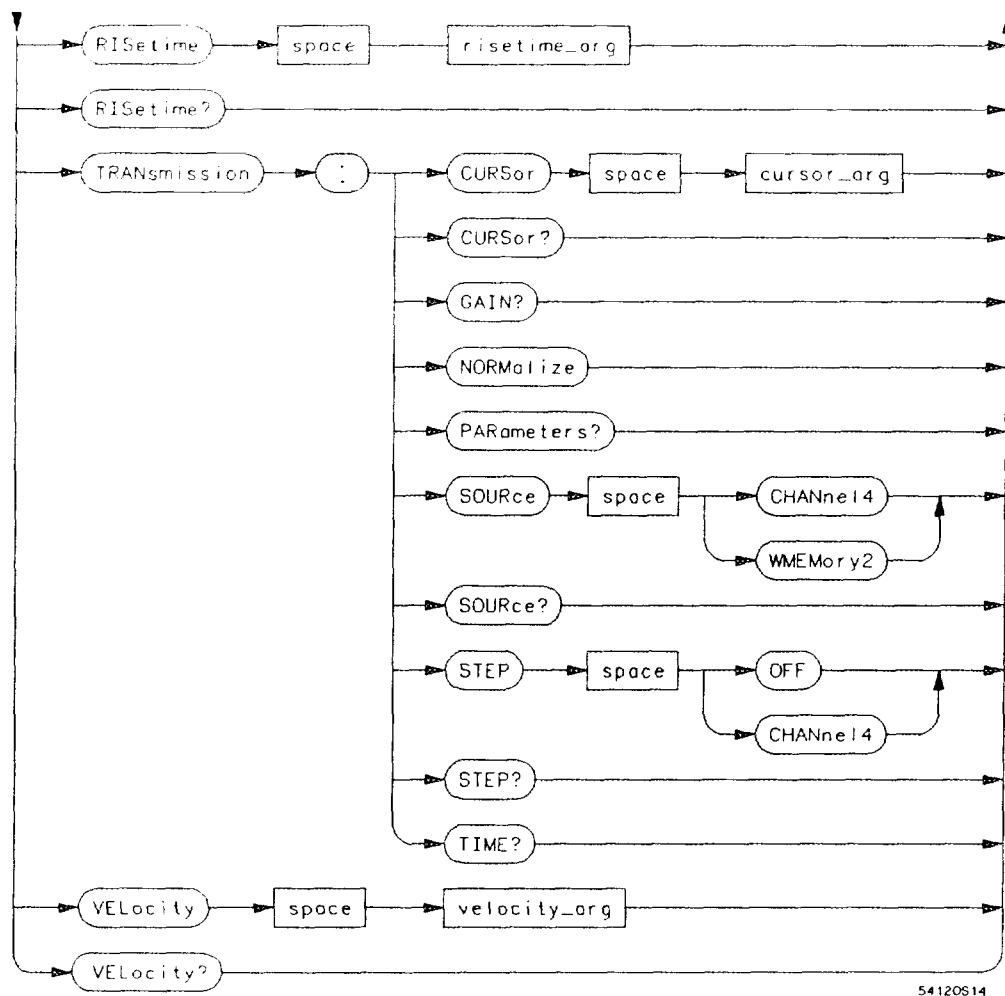


Figure 16-1. Network Subsystem Syntax Diagram



**cursor\_arg** = A real number representing the cursor's position from the reference plane. Values depend on delay and sweep speed.

**dielectric\_arg** = A real number, 1.0 through 100.0.

**risetime\_arg** = A real number defining the risetime of the filter used in normalization. Values depend on sweep speed.

**velocity\_arg** = A real number, 3.3356 through 33.356.

*Figure 16-1. Network Subsystem Syntax Diagram (continued)*

# CALibrate

---

## CALibrate

command

When SHORT is specified, the CALIBRATE command does a reflection path acquisition for a shorted termination. When FIFTY is specified, it does a reflection path acquisition for a 50  $\Omega$  termination. When OPEN is specified, it does a transmission path acquisition to remove baseline nonflatness. The THRU calibration establishes the reference time, 0 and 100 percent gain references.

### Note

*Calibration commands must be done in the following order. To do a reflection calibration, first specify SHORT. When the short calibration is complete, specify FIFTY. For the transmission cal specify OPEN, then specify THRU. Signals need to be connected and disconnected between the execution of each calibration command. See the Front-Panel Operation Reference about the network calibration procedures.*

There is no query form of this command.

**Command Syntax:** :NETWork:CALibrate {SHORT | FIFTy | OPEN | THRU}

**Example:** OUTPUT 707;":NETWORK:CALIBRATE OPEN"

---

**DIElectric****command/query**

The DIELECTRIC command sets the dielectric constant for the transmission line and forms a basis for distance measurements.

**Note**

*Changing the dielectric constant will also change the velocity constant.*

The DIELECTRIC query returns the current setting.

**Command Syntax:** :NETWORK:DIElectric < dielectric constant >  
< dielectric constant > :: = 1.0 through 100.0

**Example:** OUTPUT 707;":NETWORK:DIELECTRIC 1.5"

**Query Syntax:** :NETWork:DIElectric?

**Returned Format:** [:NETWork:DIElectric] < dielectric constant > < NL >  
< dielectric constant > :: = 1.0 through 100.0  
(exponential - NR3 format)

**Example:** DIM Diel\${30}  
OUTPUT 707;":NETWORK:DIELECTRIC?"  
ENTER 707;Diel\$  
PRINT Diel\$

## REFlection:CURSor

---

### REFlection:CURSor

command/query

The REFLECTION:CURSOR command moves the reflection cursor to the specified Delta t position relative to the reference plane established with the SHORT calibration. Once the cursor is positioned, the DISTANCE, IMPEDANCE, TIME and RHO commands can be used to determine characteristics at this point in time.

The REFLECTION:CURSOR query returns the reflection cursor position as the time from the reference plane.

**Command Syntax:** :NETWork:REFlection:CURSor <Delta t>  
<Delta t> :: = any on screen time from the reference plane to cursor

**Example:** OUTPUT 707;":NETW:REFL:CURS 25.5 NS"

**Query Syntax:** :NETWork:REFlection:CURSor?

**Returned Format:** [:NETWork:REFlection:CURSor] <Delta t> <NL>  
<Delta t> :: = time from the reference plane to cursor  
(exponential - NR3 format)

**Example:** OUTPUT 707;":NETWORK:REFLECTION:CURSOR?"  
ENTER 707;Cursor  
PRINT Cursor

#### Note

*Reflection path must be calibrated before the cursor can be turned on. The :NETWORK:REFLECTION SOURCE command determines which channel or memory the cursor will be on.*

---

### REFlection:DISTance?

query

The REFLECTION:DISTANCE query returns the distance from the cursor to the reference plane in metres.

**Query Syntax:** :NETWork:REFlection:DISTance?

**Returned Format:** [:NETWork:REFlection:DISTance] < distance > < NL >  
< distance > :: = distance in metres from cursor to reference plane  
(exponential- NR3 format)

**Example:** OUTPUT 707;":NETWORK:REFLECTION:DISTANCE?"  
ENTER 707;Dist  
PRINT Dist

The reflection cursor must be set before this command is sent.

## REFlection:IMPedance?

---

### REFlection:IMPedance?

query

The REFLECTION:IMPEDANCE query returns the impedance value in ohms at the reflection cursor's position.

**Query Syntax:** :NETWork:REFlection:IMPedance?

**Returned Format:** [:NETWork:REFlection:IMPedance] <impedance> <NL>  
<impedance> :: = impedance in ohms at cursor position  
(exponential - NR3 format)

**Example:** OUTPUT 707;":NETWORK:REFLECTION:CURSOR 20 NS;IMPEDANCE?"  
ENTER 707;Imp  
PRINT Imp

The reflection cursor **must** be set before this command is sent.



### REFlection:NORMalize

command

The REFLECTION:NORMALIZE command causes the current waveform on screen to be passed through a digital filter. The filter is defined by the :NETWORK:RISETIME command and the stored calibration waveform(s). The normalized waveform is the response of the device under test to an ideal step with the previously defined risetime. The normalized waveform is stored in waveform memory 1.

#### Note

*Reflection path must be calibrated before this command is sent.*

**Command Syntax:** :NETWork:REFlection:NORMalize

**Example:** OUTPUT 707;":NETWORK:REFLECTION:NORMALIZE"

## REFlection:PARameters?

---

### REFlection:PARameters?

query

The REFLECTION:PARAMETERS query returns the maximum and minimum percentage of reflection for the selected source (channel 1, channel 3, or waveform memory 1).

#### Note

*Reflection path must be calibrated before this command is sent.*

**Query Syntax:** :NETWork:REFlection:PARameters?

**Returned Format:** [:NETWork:REFlection:PARameters] <rho\_max>,<rho\_min> <NL>  
<rho\_max>,<rho\_min> :: = real value  
(exponential - NR3 format)

**Example:** OUTPUT 707;":NETWORK:REFL:PAR?"  
ENTER 707;Rho\_max,Rho\_min  
PRINT Rho\_max,Rho\_min

---

**REFlection:PRESet****command**

The REFLECTION:PRESET command sets the oscilloscope for viewing the step on channel 1. When this command is received, the step generator is turned on, the vertical and timebase are set, the display is set to the single screen mode, and all channels except channel 1 are turned off.

Sending this command is the same as pressing the Preset Reflect Channel key in the Network menu.

**Command Syntax:** :NETWork:REFlection:PRESet

**Example:** OUTPUT 707;":NETWORK:REFLECTION:PRESET"

## REFlection:RHO?

---

### REFlection:RHO?

query

The REFLECTION:RHO query returns the reflection ratio at the cursor position. The top and base values are established with the SHORT calibration.

The REFLECTION:RHO ratio is determined as follows:

$$\frac{V_{\text{cursor}} - V_{50}(\Omega)}{V_{\text{top}} - V_{\text{base}}}$$

**Query Syntax:** :NETWork:REFlection:RHO?

**Returned Format:** [:NETWork:REFlection:RHO] <rho> <NL>  
<rho> :: = (exponential - NR3 format)

**Example:** OUTPUT 707;"NETWORK:REFLECTION:RHO?"  
ENTER 707;Rho  
PRINT Rho

---

**REFlection:SOURce****command/query**

The REFLECTION:SOURCE command selects the source for subsequent cursor and parameter measurements. The cursor can be slaved to channel 1, if channel 1 is the currently selected reflection channel. The cursor can be slaved to channel 3, if channel 3 is the reflection channel, or to waveform memory 1, if waveform memory 1 contains a normalized waveform. The reflection channel is selected with NETWORK:REFLECTION:STEP.

**Command Syntax:** :NETWork:REFlection:SOURce {CHANnel1 | CHANnel3 | WMEMory1}

**Example:** OUTPUT 707;":NETW:REFL:SOUR WMEM1"

**Query Syntax:** :NETWork:REFlection:SOURce?

**Returned Format:** [:NETWork:REFlection:SOURce] {CHANnel1 | CHANnel3 | WMEMory1} <NL>

**Example:** DIM Sour\$[50]  
OUTPUT 707;":NETWORK:REFLECTION:SOURCE?"  
ENTER 707;Sour\$  
PRINT Sour\$

## REFlection:STEP

---

### REFlection:STEP

### command/query

The REFLECTION:STEP command turns on the step generator for reflection and determines which channel will be used for subsequent reflection calibrations and normalizations.

The REFLECTION:STEP query returns the currently selected reflection channel.

**Command Syntax:** :NETWork:REFlection:STEP {CHANnel1 | CHANnel3 | OFF}

**Example:** OUTPUT 707;":NETW:REFL:STEP CHAN1"

**Query Syntax:** :NETWork:REFlection:STEP?

**Returned Format:** [:NETWork:REFlection:STEP] {CHANnel1 | CHANnel3 | OFF} <NL>

**Example:** DIM Step\$[50]  
OUTPUT 707;":NETWORK:REFLECTION:STEP?"  
ENTER 707;Step\$  
PRINT Step\$

---

**REFLection:TIME?****query**

The REFLECTION:TIME query returns the time position of the reflection cursor, referenced to the trigger point, in seconds.

**Query Syntax:** :NETWork:REFLection:TIME?

**Returned Format:** [:NETWork:REFLection:TIME] <time> <NL>  
<time> :: = time in seconds  
(exponential - NR3 format)

**Example:** OUTPUT 707;":NETWORK:REFLECTION:TIME?"  
ENTER 707;Time  
PRINT Time

## RISetime

---

### RISetime

### command/query

The RISETIME command specifies the risetime of the normalization filter. After a reflection normalization is executed, the measured data is passed through the digital filter and the resulting waveform is placed in a waveform memory. Valid RISETIME values depend on the :TIMEBASE:RANGE setting. The minimum value is (timebase range  $\div$  125) or 10 ps, whichever is greater. The maximum value is (timebase range  $\div$  2).

The RISETIME query returns the current RISETIME value.

**Command Syntax:** :NETWork:RISetime <risetime>  
<risetime> :: = real value

**Example:** OUTPUT 707;":NETWORK:RISETIME 1.5 ns"

**Query Syntax:** :NETWork:RISetime?

**Returned Format:** [:NETWork:RISetime] <risetime> <NL>  
<risetime> :: = filter risetime  
(exponential - NR3 format)

**Example:** DIM Rise\$[30]  
OUTPUT 707;":NETWORK:RISETIME?"  
ENTER 707;Rise\$



## TRANmission:CURSor

command/query

The TRANSMISSION:CURSOR command moves the transmission cursor to the specified time from the reference time determined with the transmission calibration.

The TRANSMISSION:CURSOR query returns the transmission cursor position as the time from the reference time determined with the transmission calibration.

### Note

*Transmission path must be calibrated before this command is sent.*

**Command Syntax:** :NETWork:TRANsmission:CURSor <Delta t>  
<Delta t> :: = any on screen time from the reference time

**Example:** OUTPUT 707,":NETW:TRAN:CURS 15.5 ns"

**Query Syntax:** :NETWork:TRANsmission:CURSor?

**Returned Format:** [:NETWork:TRANsmission:CURSor] <Delta t> <NL>  
<Delta t> :: = time from the reference time to cursor (exponential - NR3 format)

**Example:** OUTPUT 707,":NETWORK:TRANSMISSION:CURSOR?"  
ENTER 707;Cursor  
PRINT Cursor

## TRANsmission:GAIN?

---

## TRANsmission:GAIN?

query

The TRANSMISSION:GAIN query returns the gain at the position of the transmission cursor in percentage.

### Note

*Transmission path must be calibrated before this command is sent.*

**Query Syntax:** :NETWork:TRANsmission:GAIN?

**Returned Format:** [:NETWork:TRANsmission:GAIN] <gain> <NL>  
<gain> :: = (exponential - NR3 format)

**Example:** OUTPUT 707;":NETWORK:TRANSMISSION:GAIN?"  
ENTER 707;Gain  
PRINT Gain

## TRANsmission:NORMAlize

### TRANsmission:NORMAlize

command

The TRANSMISSION:NORMALIZE command causes the displayed channel 4 waveform to be passed through a filter that is defined by the RISETIME command and the stored calibration waveform(s). The resulting waveform is stored in waveform memory 2.

#### Note

*Transmission path must be calibrated before this command is sent.*

**Command Syntax:** :NETWork:TRANsmission:NORMAlize

**Example:** OUTPUT 707;":NETWORK:TRANSMISSION:NORMALIZE"

## TRANsmission:PARAmeters?

---

### TRANsmission:PARAmeters?

query

The TRANSMISSION:PARAMETERS query returns the propagation delay in seconds, the distance in metres, and the gain in percentage for the device under test. The accuracy of the distance result depends on the dielectric constant.

#### Note

*Transmission path must be calibrated before this command is sent.*

**Query Syntax:** :NETWork:TRANsmission:PARAmeters?

**Returned Format:** [:NETWork:TRANsmission:PARAmeters] <prop\_delay> , <dist> , <gain> <NL>

<prop\_delay> :: = (exponential- NR3 format)

<dist> :: = (exponential- NR3 format)

<gain> :: = real value (exponential- NR3 format)

**Example:** OUTPUT 707;":NETWORK:TRAN:PAR?"  
ENTER 707;Delay,Dist,Gain  
PRINT Delay,Dist,Gain

## TRANsmission:SOURce

### TRANsmission:SOURce

command/query

The TRANSMISSION:SOURCE command selects the source for subsequent cursor and parameter measurements. The cursor can be slaved to channel 4 or waveform memory 2, if waveform memory 2 contains a normalized waveform.

The TRANSMISSION:SOURCE query returns the currently specified source.

**Command Syntax:** :NETWork:TRANsmission:SOURce { CHANnel4 | WMEMory2 }

**Example:** OUTPUT 707;":NETW:TRAN:SOUR WMEM2"

**Query Syntax:** :NETWork:TRANsmission:SOURce?

**Returned Format:** [:NETWork:TRANsmission:SOURce] { CHANnel4 | WMEMory2 } < NL >

**Example:** DIM Sour\$[50]  
OUTPUT 707;":NETWORK:TRANSMISSION:SOURCE?"  
ENTER 707;Sour\$  
PRINT Sour\$

## TRANmission:STEP

---

### TRANmission:STEP

command/query

The TRANSMISSION:STEP command turns the step generator for channel 4 on or off.

The TRANSMISSION:STEP query returns channel 4 or OFF.

**Command Syntax:** :NETWork:TRANmission:STEP {CHANnel4 | OFF}

**Example:** OUTPUT 707;":NETW:TRAN:STEP OFF"

**Query Syntax:** :NETWork:TRANmission:STEP?

**Returned Format:** [:NETWork:TRANmission:STEP] {CHANnel4 | OFF} <NL>

**Example:** DIM Step\$[50] OUTPUT 707;":NETWORK:TRANSMISSION:STEP?"  
ENTER 707;Step\$  
PRINT Step\$

---

### TRANsmission:TIME?

query

The TRANSMISSION:TIME query returns the time position of the transmission cursor, referenced to the trigger point, in seconds.

**Query Syntax:** :NETWork:TRANsmission:TIME?

**Returned Format:** [:NETWork:TRANsmission:TIME] <time> <NL>  
<time> :: = (exponential - NR3 format)

**Example:** OUTPUT 707;":NETWORK:TRANSMISSION:TIME?"  
ENTER 707;Time  
PRINT Time

## VELOCITY

---

### VELOCITY

### command/query

The VELOCITY command sets the velocity constant for the transmission line in ns/metre. The velocity constant is used for distance measurements.

#### Note

*Changing the velocity constant will also change the dielectric constant.*

The VELOCITY query returns the current setting.

**Command Syntax:** :NETWork:VELOCITY <velocity constant >  
<velocity constant > :: = 3.3356 ns/metre to 33.356 ns/metre

**Example:** OUTPUT 707;":NETWORK:VELOCITY 3.8e-9"

**Query Syntax:** :NETWork:VELOCITY?

**Returned Format:** [:NETWork:VELOCITY] <velocity constant > <NL>  
(exponential - NR3 value)

**Example:** DIM Vel\$[30]  
OUTPUT 707;":NETWORK:VELOCITY?"  
ENTER 707;Vel\$  
PRINT Vel\$



# Timebase Subsystem

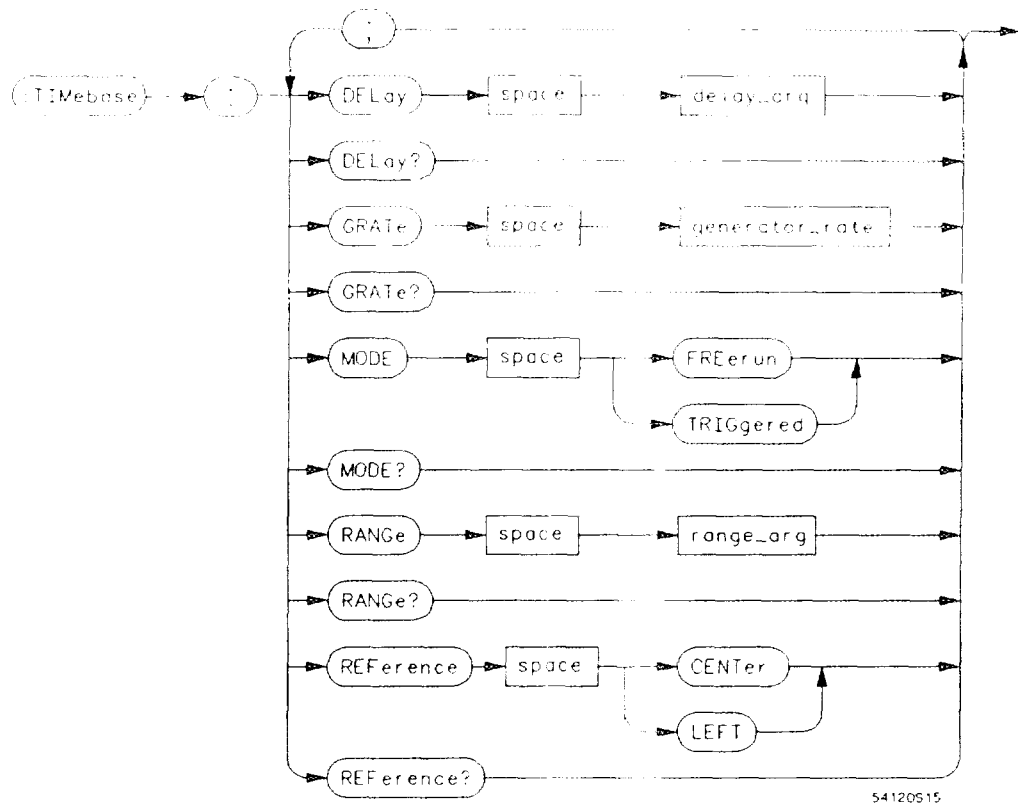
---

17

## Introduction

The TIMEBASE subsystem commands control the horizontal axis "X axis," oscilloscope functions.

See figure 17-1 for a syntax diagram of the TIMEBASE subsystem.



**delay\_arg** = A real number, maximum depends on sweep range.

**generator\_rate** = A real number, 15.3 through 500000.

**range\_arg** = A real number, 100 ps through 10 s.

*Figure 17-1. Timebase Subsystem Syntax Diagram*

**DELaY****command/query**

The DELAY command sets the timebase delay. Delay is the time interval between the trigger event and the displayed delay reference. The delay reference is the left edge or center of the display. The delay reference is set with the :TIMEBASE:REFERENCE command.

The DELAY query returns the current delay value.

**Command Syntax:** :TIMebase:DELaY <delay>

<delay> :: = time in seconds from trigger to display reference. Maximum value depends on time/division setting. Minimum value at left reference is 16 ns.

**Example:** OUTPUT 707;":TIMEBASE:DELAY 2E-3"  
OUTPUT 707;":TIMEBASE:DEL .002"  
OUTPUT 707;":TIM:DEL 2MS"

**Note**

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :TIMebase:DELaY?

**Returned Format:** [:TIMebase:DELaY] <delay> <NL>  
<delay> :: = time from trigger to display reference in seconds. Display reference is left or center  
(exponential - NR3 format)

**Example:** OUTPUT 707;":TIMEBASE:DELAY?"  
ENTER 707;Del  
PRINT Del

## GRATe

---

|              |                         |                      |
|--------------|-------------------------|----------------------|
| <b>GRATe</b> | <b>(Generator Rate)</b> | <b>command/query</b> |
|--------------|-------------------------|----------------------|

The GRATe command sets the rate of the freerun rate generator. Valid values for the generator rate are 15.3 Hz to 500 kHz.

The freerun rate generator is active only in the Freerun mode. The legal values for this command are defined by the following formula.

$\text{GRATE} = 1 \text{ MHz } N + 1$  ; where  $N :: = \text{An integer, 1 through 65535}$

The GRATe query returns the current frequency of the freerun rate generator.

**Command Syntax:**    :TIMebase:GRATe <rate>  
  
                          <rate> :: = 15.3 through 500000

**Example:**    OUTPUT 707;":TIMEBASE:GRATE 2000"  
                  OUTPUT 707;":TIMEBASE:GRAT 2E + 3"  
                  OUTPUT 707;":TIM:GRAT 2K"

**Query Syntax:**    :TIMebase:GRATe?

**Returned Format:**    [:TIMbase:GRATe] <value> <NL>  
  
                          <value> :: = freerun rate  
                          (exponential - NR3 format)

**Example:**    OUTPUT 707;":TIMEBASE:GRATE?"  
                  ENTER 707;Grate  
                  PRINT Grate

## MODE

## command/query

The MODE command selects the timebase mode. If the FREERUN mode is selected, the oscilloscope 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 oscilloscope will not sweep, and the data acquired on the previous trigger will remain on screen.

The MODE query returns the current mode.

**Command Syntax:** :TIMebase:MODE {FREerun | TRIGgered}

**Example:** OUTPUT 707;":TIM:MODE FREERUN"

**Query Syntax:** :TIMebase:MODE?

**Returned Format:** [:TIMebase:MODE] <mode> <NL>

<mode> :: = {FREerun | TRIGgered}

**Example:** DIM Mode\$[30]  
OUTPUT 707;"TIMEBASE:MODE?"  
ENTER 707;Mode\$  
PRINT Mode\$

# RANGe

---

## RANGe

command/query

The 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> :: = 100.0 ps to 10 s

**Example:** OUTPUT 707;:TIMEBASE:RANGE 0.1"  
OUTPUT 707;:TIMEBASE:RANG 1E-1"  
OUTPUT 707;:TIM:RANG 100 MS"

### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :TIMebase:RANGe?

**Returned Format:** [:TIMebase:RANGe] <range> <NL>  
<range> :: = 100.0 ps to 10 s  
(exponential - NR3 format)

**Example:** OUTPUT 707;:TIMEBASE:RANGE?"  
ENTER 707;Range  
PRINT Range

## REFerence

## command/query

The REFERENCE command sets the delay reference to the left side of the screen or to the center of the screen.

The REFERENCE query returns the current delay reference.

**Command Syntax:** :TIMebase:REFerence {LEFT | CENTer}

**Example:** OUTPUT 707;":TIMEBASE:REFERENCE LEFT"

**Query Syntax:** :TIMebase:REFerence?

**Returned Format:** [:TIMebase:REFerence] <reference> <NL>

<reference> :: = {LEFT | CENTer}

**Example:** DIM Ref\$[30]  
OUTPUT 707;":TIMEBASE:REFERENCE?"  
ENTER 707;Ref\$  
PRINT Ref\$

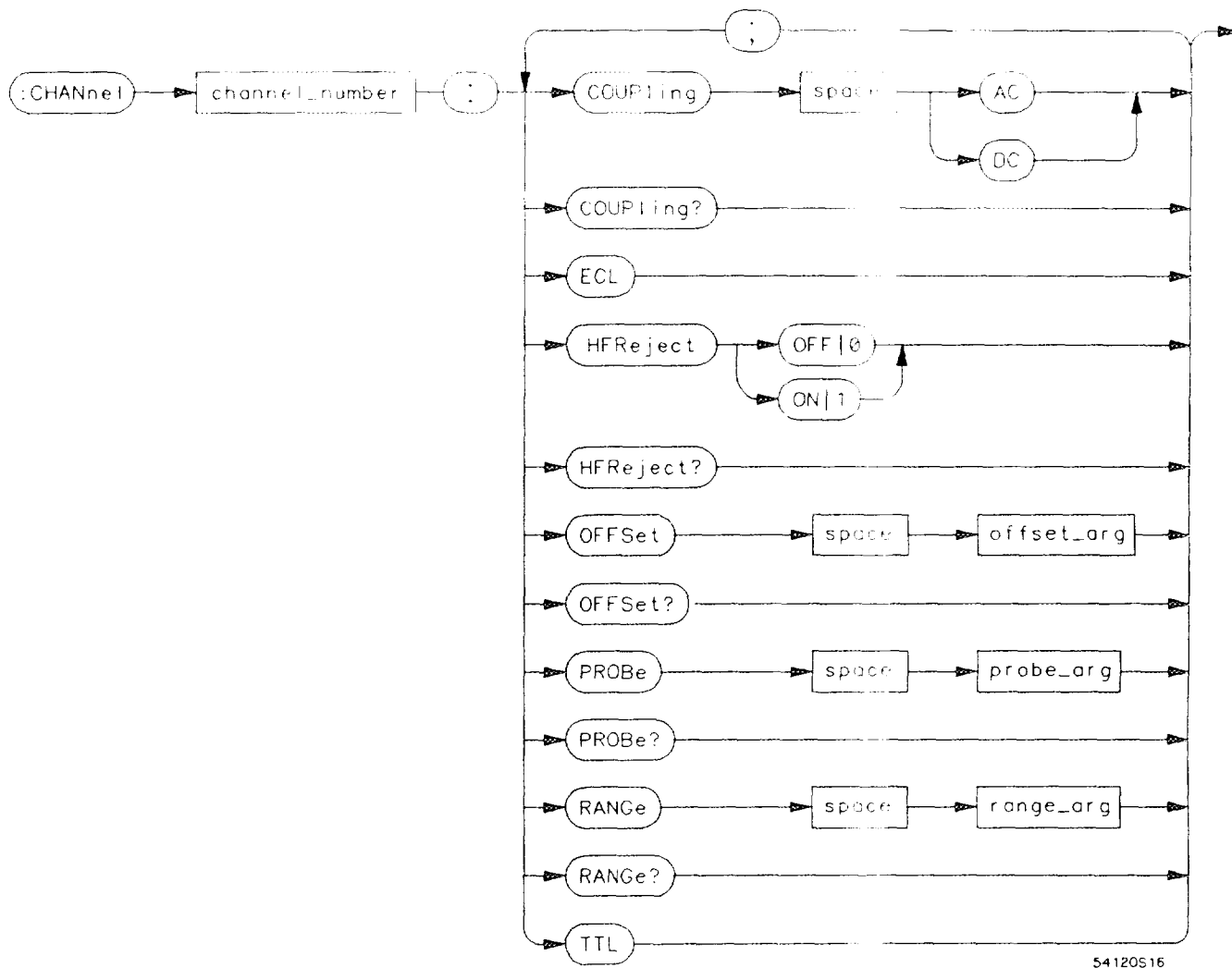




## Introduction

The commands in the TRIGGER subsystem are used to define the conditions for a trigger.

See figure 18-1 for the TRIGGER subsystem syntax diagram.



**level\_arg** = A real number, specifying the trigger level setting in volts.

**probe\_arg** = A real number, specifying the probe attenuation for the trigger source. Range is 1 through 1000.

*Figure 18-1. Trigger Subsystem Syntax Diagram*

## HFReject

## command/query

The HF REJECT command turns the HF Reject function on and off. With HF Reject set to On, the trigger bandwidth is reduced to less than 100 MHz. With this function set to Off, the full trigger bandwidth is used.

The HFREJECT query returns the current setting of this function.

**Command Syntax:** :TRIGger:HFReject {{ON | 1} | {OFF | 0}}

**Example:** Output 707;":TRIGGER:HFREJECT ON"

**Query Syntax:** :TRIGger:HFReject?

**Returned Format:** [:TRIGger:HFReject]{1 | 0}

**Example:** DIM RJT\${30}  
OUTPUT 707;":TRIG:HFR?"  
ENTER 707;RJT\$  
PRINT RJT\$

## LEVel

---

### LEVel

### command/query

The LEVEL command sets the trigger level voltage of the external trigger.

The LEVEL query returns the trigger level of the external trigger.

This command has no effect when the Freerun mode is selected, and is inactive when the step is displayed.

**Command Syntax:** :TRIGger:LEVel <level>

<level> :: = trigger level setting in volts

**Example:** OUTPUT 707;":TRIGGER:LEVEL .30"  
OUTPUT 707;":TRIGGER:LEV 300MV"  
OUTPUT 707;":TRIG:LEV 3E - 1"

#### Note

*Refer to chapter 3 for the syntax of using values with multipliers.*

**Query Syntax:** :TRIGger:LEVel?

**Returned Format:** [:TRIGger:LEVel] <level> <NL>

<level> :: = trigger level in volts  
(exponential - NR3 format)

**Example:** DIM Level\${30}  
OUTPUT 707;":TRIGGER:LEVEL?"  
ENTER 707;Level\$  
PRINT Level\$

---

**MODE****command/query**

The MODE query returns EDGE.

## PROBe

---

### PROBe

### command/query

The PROBE command specifies the attenuation factor for the external trigger.

The PROBE query returns the probe attenuation factor of the external trigger.

Changing the probe attenuation does not change the hardware, nor does it attenuate the input. The probe attenuation only changes the internal scale factors for display.

**Command Syntax:** :TRIGger:PROBe <probe atten> :: = 1 to 1000

<probe atten> :: = 1 to 1000

**Example:** OUTPUT 707;":TRIGGER:PROBE 10"

**Query Syntax:** :TRIGger:PROBe?

**Returned Format:** [:TRIGger:PROBe] <atten> <NL>

<atten> :: = 1 to 1000  
(exponential - NR3 format)

**Example:** OUTPUT 707;":TRIG:PROBE?"  
ENTER 707;Probe  
PRINT Probe

---

**SENSitivity****command/query**

The SENSITIVITY command sets the trigger sensitivity function to either NORMAL or HIGH.

When the trigger sensitivity is set to HIGH a smaller voltage will cause the oscilloscope to trigger. This setting is useful to allow the oscilloscope to trigger on a smaller signal at high frequencies. The high sensitivity setting should only be used at high frequencies. If the HIGH setting is used at low frequencies, double triggering will most likely occur due to noise.

The SENSITIVITY query returns the current setting.

**Command Syntax:** :TRIGger:SENSitivity{NORMal | }

**Example:** OUTPUT 707;":TRIG:SENS NORM"

**Query Syntax:** :TRIGGER:SENSITIVITY?

**Returned Format:** [:TRIGger:SENSitivity]{NORMAL | HIGH}

**Example** DIM SNS\$(30)  
OUTPUT 707;":TRIGGER:SENSITIVITY?"  
Enter 707;SNS\$  
PRINT SNS\$

**Note**

*The oscilloscope does not respond to the query with a short form of NORMAL, but only used the longform.*

# SLOPe

---

## SLOPe

## command/query

The SLOPE command specifies the slope for the external trigger.

The SLOPE query returns the current slope for the external trigger.

This command is inactive when the TDR step is displayed.

**Command Syntax:** :TRIGger:SLOPe {NEGative | POSitive}

**Example:** OUTPUT 707;":TRIGGER:SLOPE POSITIVE"

**Query Syntax:** :TRIGger:SLOPE?

**Returned Format:** [:TRIGger:SLOPe] <slope> <NL>

<slope> :: = {POSitive | NEGative}

**Example:** DIM Slope\$(30)  
OUTPUT 707;":TRIG:SLOP?"  
ENTER 707;Slope\$  
PRINT Slope\$



---

**SOURce**

**command/query**

The SOURCE query returns EXTERNAL1.



## Introduction

The WAVEFORM subsystem is used to transfer waveform data between a controller and the HP 54121T'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 with the root level command :DIGITIZE. The preamble contains the information for interpreting the waveform data. This would include 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 CHANNEL:RANGE command has no effect on waveform data.

The values set in the preamble are determined when the :DIGITIZE command is executed, and are based on the settings of variables in the ACQUIRE subsystem. Although the preamble values can be changed with a controller, the way the data was acquired can not be changed. Changing the preamble values can not 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 54121T) with two separate commands, DATA and PREAMBLE.

Refer to figure 19-1 for the syntax diagram of the WAVEFORM subsystem.

---

## Data Acquisition Types

There are four types of waveform acquisition that can be selected with the :ACQUIRE:TYPE command. The four types are NORMAL (persistence), AVERAGE, ENVELOPE, and HISTOGRAM. The type of data acquisition selected and the source(s) specified for the DIGITIZE command will determine where the acquired (digitized) data is placed. Data for TYPES NORMAL and AVERAGE is placed in the same number Waveform Memory as the channel number that was digitized. Envelope data is placed in Waveform Memories 1 and 3 for channels 1 and 3, and in Waveform Memories 2 and 4 for channels 2 and 4. Histogram data is placed in Waveform Memory 5.

**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 WAVEFORM:COUNT query. Time buckets that have fewer than  $n$  hits return the average of what data they do have. 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.

**Histogram** Histogram data consists of a list of the number of "hits" (samples) acquired in each voltage or time bucket (depending on histogram type) within the histogram window. Empty buckets will return 0. Histogram data is always stored in Waveform Memory 5.

The data is transferred over the HP-IB in a linear fashion starting with bucket 0 and proceeding through bucket  $n - 1$  where  $n$  is the value returned by the WAVEFORM:POINTS query. Voltage histograms will return 256 points with point 0 corresponding to the bottom of the display and point  $n - 1$  (255) corresponding to the top of the display. Time histograms return 500 points with the first point corresponding to the left side of the display and the last point corresponding to the next to last point on the right side of the display. Histogram data values may be transferred over the HP-IB in either ASCII or long format. Word format is not allowed because legal histogram data values can be too large to represent in that format.

## 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 side of the CRT. The last value is one data point away from the right side of the CRT.

Envelope data is stored in waveform memories as follows:

| Channel number | Minimum data:<br>stored in: | Maximum data<br>stored in: |
|----------------|-----------------------------|----------------------------|
| Odd            | Wmem1                       | Wmem3                      |
| Even           | Wmem2                       | Wmem4                      |

The data in the waveform memories is transferred to a controller using the WAVEFORM:DATA query. The memory source for the transfer is specified using the WAVEFORM:SOURCE command. If waveform memory 1 or 3 is specified as the source, the minimum and maximum data from waveform memories 1 and 3 is transferred over HP-IB.

Similarly, if waveform memory 2 or 4 is specified as the source, the minimum and maximum data from waveform memories 2 and 4 is transferred over HP-IB.

If it is desirable to transfer only the data from one of the envelope memories (ie. only Wmem1 and not Wmem3), the **WAVEFORM:TYPE** command should be used to change that waveform memory's type from envelope to normal.

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 or Waveform Memories 2 and 4 by specifying the types for the memories as envelope. The data is then transferred as two arrays. If Waveform Memory 1 or 3 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 or 4 is specified as the source, the first array is transferred into Waveform Memory 2 and the second array is transferred into Waveform Memory 4.

---

## **Data Conversion**

Data sent from the HP 54121T 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, which 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:

$$\text{voltage} = [(\text{data value} - \text{yreference}) * \text{yincrement}] + \text{yorigin}$$

### **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 = 1, and XINCREMENT = 2 ns position is calculated using the formula:

$$\text{time} = [(\text{data point number} - \text{xreference}) * \text{xincrement}] + \text{xorigin}$$

would result in the following calculation:

$$\text{time} = [(3 - 1) * 2 \text{ ns}] + 16 \text{ ns} = 20 \text{ ns}.$$

---

## Data Format for HP-IB Transfer

There are three formats for transferring waveform data over the HP-IB. These formats are WORD, ASCII, and LONG.

### WORD Format

WORD formatted waveform records are transmitted using the binary block format. When you use WORD format, the ASCII character string "`# <N> <DD...D>`" is sent before the actual data. The `<N>` 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 ASCII string "`# 41024`" would be sent. The 4 indicates that four length digits follow, 1024 indicates that 1,024 data bytes (binary) follow.

The number of data bytes is twice the number of words (data points). The number of words is also 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. WORD format is useful in applications where the information is read directly into an integer array in a controller.

### 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 would be 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.

### LONG Format

LONG data is used only for transferring the histogram data. The histogram data is sent in block format, like the word data. Each data point is sent as a 32-bit integer value, most significant byte first. The integer value represents the number of samples that fell in that time bucket. The x-axis voltage or time value is computed the same as for word or ASCII data.



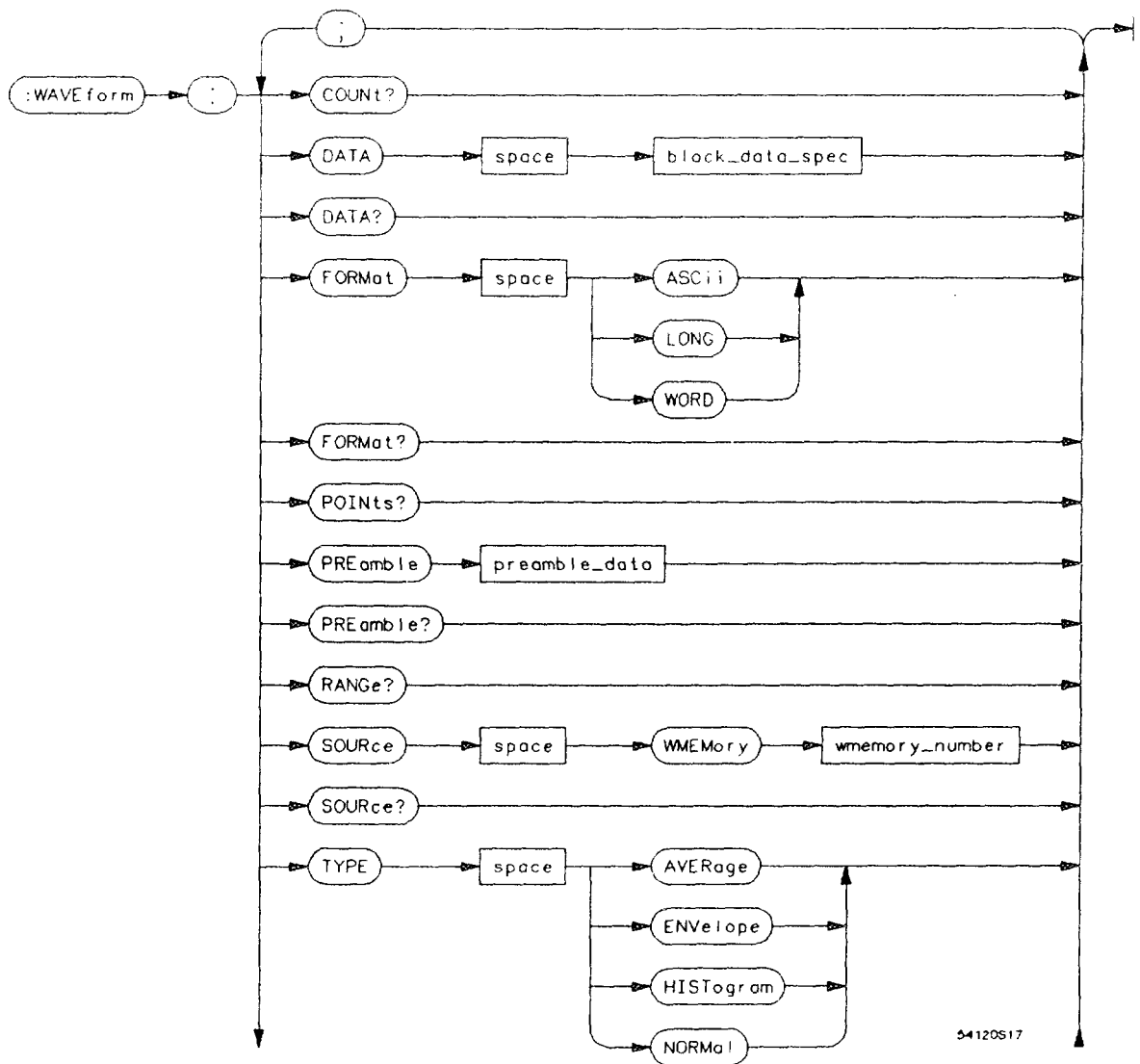


Figure 19-1. Waveform Subsystem Syntax Diagram



**block\_data\_spec** = A block of data in # format.

**preamble\_data** = Refer to PREAMBLE command.

**wmemory\_number** = An integer 1 through 5.

*Figure 19-1. Waveform Subsystem Syntax Diagram (continued)*

---

**COUNT?**

query

The COUNT query returns the count field of the waveform preamble. The count field contains the number of averages if the acquisition was in average mode, the number of hits in each time bucket if the acquisition was in envelope mode, or the number one if type was normal mode.

If ACQUIRE:TYPE is HISTOGRAM the COUNT query returns the actual number of samples taken.

**Query Syntax:** :WAVEform:COUNT?

**Returned Format:** [:WAVEform:COUNT] <value> <NL>

<value> :: = count as described above  
(integer - NR1 format)

**Example:** OUTPUT 707;":WAVEFORM:COUNT?"  
ENTER 707;Count  
PRINT Count

## DATA

---

### DATA

### command/query

The DATA command causes the instrument to accept a waveform data record over the HP-IB from the controller and store it in the previously specified waveform memory. The waveform memory is specified with a :WAVEFORM:SOURCE command.

If the output is ENVELOPE data the minimum and maximum values for each time point are output together. The first value is the minimum for that time point, the second is the maximum for that time point. The format of the data being sent must match the format previously specified by its preamble for the destination memory.

The DATA query tells the instrument to output the waveform record stored in the waveform memory, 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 54121T to the controller and then back to the HP 54121T.

## Example:

```

10  Clear 707
20                                     !Setup Acquire Subsystem
30  OUTPUT 707; " :ACQUIRE:TYPE NORMAL: COUNT 256;POINTS 512"
40  OUTPUT 707; " :DIGITIZE CHANNEL 1"           !Store Chan 1 Display to WMEM 1
50  OUTPUT 707; " :SYSTEM:HEADER OFF; :EOI ON"
60  OUTPUT 707;":WAVEFORM:SOURCE WMEMORY1;FORMAT WORD" !Select waveform
70                                     !Data, Source and Output Format
80  OUTPUT 707; " :WAVEFORM: DATA?"           !Output Source to Controller
90  ENTER 707 USING "#,A,D" ;Header$,Bytes      !Read Length Byte
100 IF Bytes = 3 THEN                    !Read length of Data Bytes to Follow
110   ENTER 707 USING "#,3D" ;Length
120 END IF
130 IF BYTES = 4 THEN
140   ENTER 707 using "#,4D" ;Length
150 END IF
160 Length = Length/2
170 ALLOCATE INTEGER Waveform (1:Length)
180 ENTER 707 USING "#" ;Waveform(*)          !Enter Waveform Data to Integer Arra
190 ENTER 707 USING "-K,B" ;End$              !Enter Terminator
200 DIM Preamble$[200]
210 OUTPUT 707; " :WAV:PREAMBLE?"           !Output Wave Source Preamble to Controller
220 ENTER 707 USING "-K" ;Preamble$           !Enter Preamble into Controller
230 OUTPUT 707; " :WAV:SOURCE WMEMORY4"       !Change Source to WMEMORY 4
240 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE !Send Preamble From
250                                           !Controller to WMEMORY 4
260 OUTPUT 707: " :WAVEFORM:DATA";Waveform(*) !Send Waveform Data to WMEMORY
270 OUTPUT 707;":BLANK CHANNEL1;;VIEW WMEMORY4 !Turn Chan 1 off - WMEM 4 on
280 END

```

# FORMat

---

## FORMat

## command/query

The FORMAT command sets the data transmission mode for waveform data output. This command controls how the data is sent out, while the preamble determines how the data is input. The preamble contains the format requested at the time the waveform data is output. When you send data to the HP 54121T, we recommend that the preamble be sent first.

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.

LONG format is used to transfer histogram data. LONG formatted data is transferred as 32-bit binary integers, in four bytes with the most significant byte sent first.

The FORMAT query returns the current output format for transfer of waveform data.

**Command Format:** :WAVeform:FORMat {ASCii | WORD | LONG}

**Example:** OUTPUT 707;":WAV:FORMAT WORD"

**Query Syntax:** :WAVeform:FORMat?

**Returned Format:** [:WAVeform:FORMat] <mode> <NL>

<mode> :: = {ASCii | WORD | LONG}

**Example:** DIM Format\${30}  
OUTPUT 707;":WAV:FORMAT?"  
ENTER 707;Format\$  
PRINT Format\$

### POINTS?

**query** The POINTS query returns the points value in the currently selected waveform preamble. The points value is the number of points acquired in the last :DIGITIZE command to the waveform memory previously selected with the :WAVEFORM:SOURCE command.

**Query Syntax:** :WAVeform:POINTs?

**Returned Format:** [:WAVeform:POINTs] <value> <NL>

<value> :: = number of acquired data points  
(integer - NR1 format)

**Example:** OUTPUT 707;":WAV:POINTS?"  
ENTER 707;Points  
PRINT Points

## PREamble

---

### PREamble

### command/query

The 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 specified memory.

**Command Syntax:** :WAVEform:PREamble <preamble block>

```
<preamble block> :: = <format> ,  
<type> ,  
<points> ,  
<count> ,  
<xincrement> ,  
<xorigin> ,  
<xreference> ,  
<yincrement> ,  
<yorigin> ,  
<yreference> ,  
<yrange>
```

**Query Syntax:** :WAVEform:PREamble?

**Returned Format:** [:WAVEform:PREamble] <format NR1> ,  
<type NR1> ,  
<points NR1> ,  
<count NR1> ,  
<xincrement NR3> ,  
<xorigin NR3> ,  
<xreference NR1> ,  
<yincrement NR3> ,  
<yorigin NR3> ,  
<yreference NR1> ,  
<yrange NR3> <NL>



## PREAmble

**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 it is printed by the controller. After being printed the preamble is returned to the previously selected waveform memory.

### Note

*This example only reads the waveform preamble. In normal operation the preamble would be read, then the waveform data would be read. To send the information back to the instrument, send the waveform preamble then the waveform data (refer to Appendix B, Program Examples).*

```
10 DIM Pre$[120]
20 OUTPUT 707;"SYSTEM:HEADER OFF"
30 OUTPUT 707;"WAVEFORM:PREAMBLE?"
50 ENTER 707 USING "—K";Pre$
60 PRINT USING "K";Pre$
70 OUTPUT 707 USING " # ,K";"WAV:PREAMBLE ";Pre$
80 END
```

In line 70 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 (PREAMBLE2 or PREAMBLE1) that is not a proper command word.

RANGe SOURce RANGe? query. The RANGE query returns the y-axis voltage range currently in the preamble.

**Query Syntax:** :WAVeform:RANGe?

**Returned Format:** [:WAVeform:RANGe] <range> <NL>  
<range> :: = y-axis voltage range in current preamble

**Example:** DIM Rng\$[30]  
OUTPUT 707;"WAVEFORM:RANGE?"  
ENTER 707;Rng\$  
PRINT Rng\$

## SOURce

---

### SOURce

### command query

The **SOURCE** command selects the 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 WMemory {1 | 2 | 3 | 4 | 5}

**Example:** OUTPUT 707;":WAV:SOURCE WMEMORY3"

**Query Syntax:** :WAVeform:SOURce?

**Returned Format:** [:WAVeform:SOURce] < argument > < NL >

< argument > :: = WMemory {1 | 2 | 3 | 4 | 5}

**Example:** DIM Src\$[30]  
OUTPUT 707;":WAVEFORM:SOURCE?"  
ENTER 707;Src\$  
PRINT Src\$

**TYPE**

**command/query**

The TYPE command sets the data type for the currently selected memory. This command is useful for changing a memory from envelope to normal in order to extract only the minimum or maximum data values from the envelope.

The TYPE query returns the data type for the previously specified memory.

**Command Syntax:** :WAVeform:TYPE {NORMal | AVERage | ENVELOpe | HISTogram}

**Example:** OUTPUT 707;":WAV:TYPE AVERAGE"

**Query Syntax:** :WAVeform:TYPE?

**Returned Format:** [:WAVeform:TYPE] <mode> <NL>

<mode> :: = {AVERage | ENVELOpe | NORMal | HISTogram}

**Example:** DIM Type\$[30]  
OUTPUT 707;":WAVEFORM:TYPE?"  
ENTER 707;Type\$  
PRINT Type\$

## VALid?

---

### VALid?

query

The VALID query returns 0 if there is no data in the memory. If there is valid data in the selected memory, the response will be 1.

**Query Syntax:** :WAVeform:VALid?

**Returned Format:** [:WAVeform:VALid] {1 | 0} <NL>

**Example:** OUTPUT 707;":WAV:VALID?"  
ENTER 707;Val  
PRINT Val

## XINCrement?

query

The XINCREMENT query returns the x-increment value currently in the preamble. This value is the time difference between consecutive data points for NORMAL, AVERAGE, or ENVELOPE data. For HISTOGRAM data it is either the time difference for a time histogram or the voltage difference for a voltage histogram between consecutive data points.

**Query Syntax:** :WAVeform:XINCrement?

**Returned Format:** [:WAVeform:XINCrement] <value> <NL>

<value> :: = x-increment in the current preamble  
(exponential - NR3 format)

**Example:** OUTPUT 707;":WAV:XINCREMENT?"  
ENTER 707;Xin  
PRINT Xin

## XORigin?

---

## XORigin?

**query**

The XORIGIN query returns the x-origin value currently in the preamble. This value is the time of the first data point in the memory with respect to the trigger point. If ACQUIRE:TYPE was Histogram the XORIGIN value can be a voltage value.

**Query Syntax:** :WAVeform:XORigin?

**Returned Format:** [:WAVeform:XORigin] <value> <NL>

<value> :: = x-origin value currently in preamble  
(exponential - NR3 format)

**Example:** OUTPUT 707;":WAV:XORIGIN?"  
ENTER 707;Xor  
PRINT Xor

## XREFerence?

query

The XREFERENCE query returns the current x-reference value in the preamble. This value specifies the data point associated with the x-origin data value. For the HP 54121T this value is always zero except in HISTOGRAM, when it will be 16384.

**Query Syntax:** :WAVeform:XREFerence?

**Returned Format:** [:WAVeform:XREFerence] <value> <NL>

<value> :: = x-reference value in the current preamble  
(integer - NR1 format)

**Example:** OUTPUT 707;":WAV:XREFERENCE?"  
ENTER 707;Xref  
PRINT Xref

## YINCrement?

---

## YINCrement?

query

The YINCREMENT query returns the y-increment value currently in the preamble. This value is the voltage difference between consecutive data values.

**Query Syntax:** :WAVeform:YINCrement?

**Returned Format:** [:WAVeform:YINCrement] <value> <NL>

<value> :: = y-increment value in the current preamble  
(exponential - NR3 format)

**Example:** OUTPUT 707;":WAV:YINCREMENT?"  
ENTER 707;Yin  
PRINT Yin



---

**YORigin?****query**

The YORIGIN query returns the y-origin currently in the preamble.  
This value is the voltage at center screen.

**Query Syntax:** :WAVeform:YORigin?

**Returned Format:** [:WAVeform:YORigin] <value> <NL>

<value> :: = y-origin in the current preamble  
(exponential -NR3 format)

**Example:** OUTPUT 707;":WAV:YORIGIN?"  
ENTER 707;Yor  
PRINT Yor

## YREference?

---

## YREference?

query

The YREFERENCE query returns the current y-reference value in the preamble. This specifies the data value where the y-origin occurs.

**Query Syntax:** :WAVeform:YREference?

**Returned Format:** [:WAVeform:YREference] <value> <NL>

<value> ::= y-reference value in the current preamble  
(integer - NR1 format)

**Example:** OUTPUT 707;"WAV:YREFERENCE?"  
ENTER 707;Yref  
PRINT Yref

# How Measurements Are Made

---

## Introduction

Hewlett-Packard's digitizing oscilloscopes provide the user with automatic measurement capability. By the push of a front-panel button or via HP-IB command the instruments will measure many of the pulse parametrics. These measurements provide a fast repeatable and accurate answer to simplify bench measurements and make automated time domain measurements possible.

Like any tool, it is important for the user to understand the tool's uses, its limitations, and the methods which may be used to overcome some of these limitations. Automatic parametric measurements are extremely useful and are an integral part of automated test systems. By understanding how these measurements work, it is possible (in the vast majority of applications) to overcome most of their limitations.

---

## Automatic Top - Base

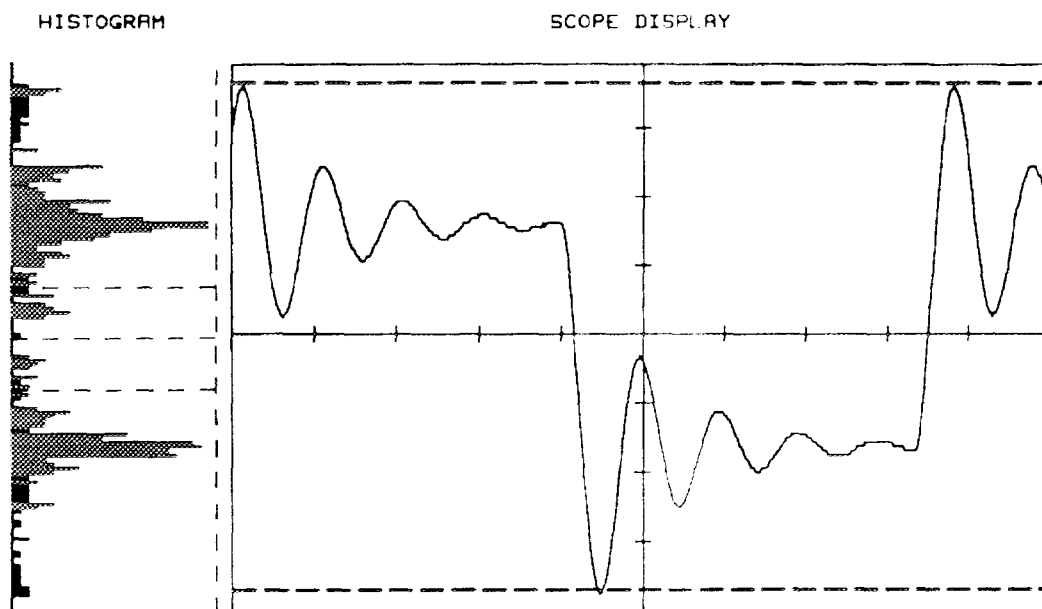
Top — Base is the heart of most automatic parametric measurements. It is used to find the top and bottom of the waveform. From this information the instrument can determine the 10, 50, and 90 percent points which are used in most automated measurements. The Top or Base of the waveform is not necessarily the maximum or minimum voltage present on the waveform. Consider a pulse that has a slight amount of overshoot. It would be incorrect to select the highest peak as the Top of the waveform since the waveform normally rests below the perturbation. Top — Base histograms the waveform and finds the most prevalent point above and below the waveform midpoint. The most prevalent point that represents greater than approximately 5% of the total display points (501) is considered to be either the Top or Base. If no point accounts for more than 5% of the total then the absolute maximum is chosen as the Top and the absolute minimum is chosen as the Base.

## Upper and Lower Limits

The algorithm does not look at all the points from the waveform midpoint to either maximum. It determines the midpoint and searches (histograms) from the absolute extreme to a level within 20% of the midpoint. This is intended to prevent the baseline from being chosen as the Top or Base in non-symmetrical waveforms.

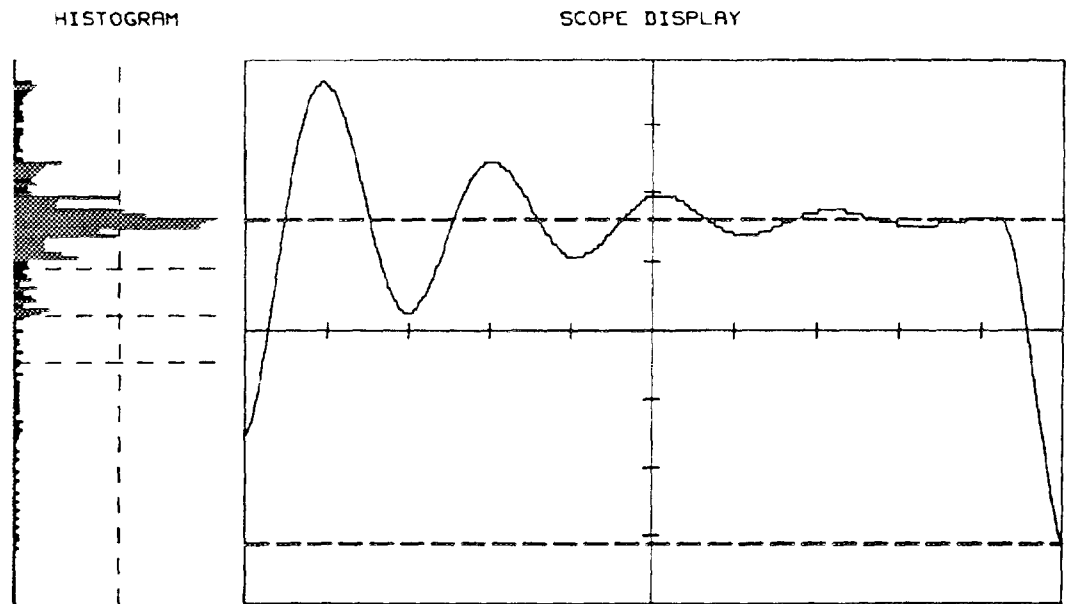
Figure A-1 shows a waveform and the histogram of the vertical pixels (Q- levels) that the instrument uses to find the Top and Base. The three horizontal dashed lines in the histogram show the waveform's MIDPOINT, UPPER LIMIT and LOWER LIMIT. The vertical dashed line shows the 5% point (represents approximately 25 points). The greatest magnitude beyond this level will be chosen as the Top if above the UPPER LIMIT and as the Base if below the LOWER LIMIT. If no point has a 5% magnitude then the Top or Base will revert to the absolute maximum or minimum (as in figure A-1).

TOP BASE



*Figure A-1. Top of Waveform Incorrectly Found*

## TOP BASE



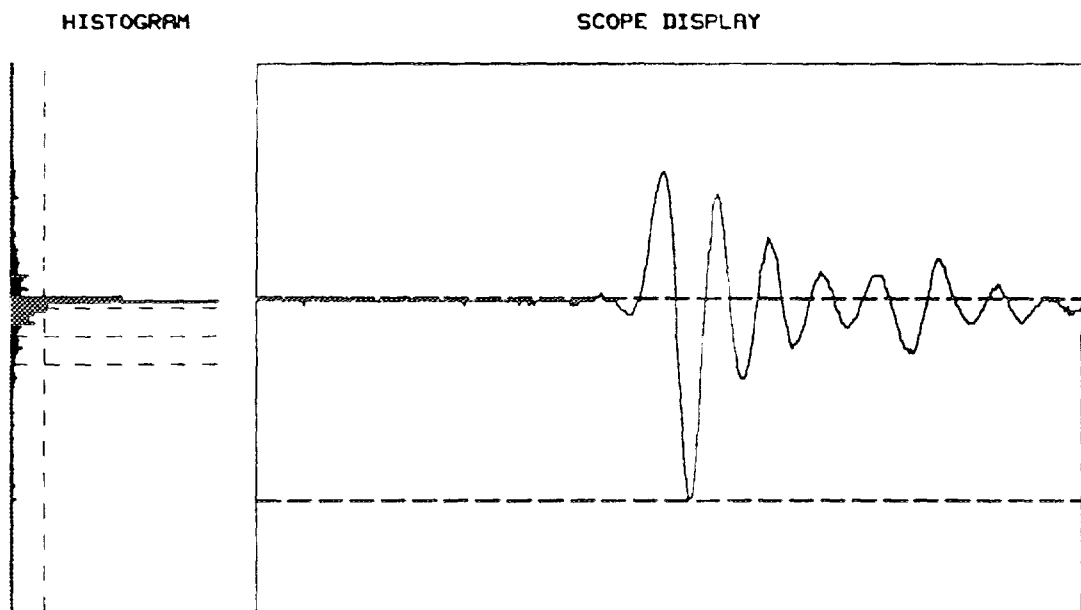
*Figure A-2. Horizontal Expansion of Waveform Allows Top to be Easily Found*

In order for the instrument to correctly determine the Top of the waveform shown in figure A-1, the distribution of the waveform must be changed so that some point meets the 5% criteria. This is accomplished by expanding the horizontal scale as shown in figure A-2. Note that the Top of the waveform is correctly found. The same technique can be used to find the Base.

## Non-Symmetrical Waveforms

The middle 40% of the waveform is not checked when looking for the Top and Base. This maximizes the amount of non-symmetry that a waveform may have and still be measured correctly. Figure A-3 shows the histogram and points where the Top and Base were found. Note that the baseline was chosen for the Top of the signal. To eliminate this error the baseline preceding the pulse should be moved off-screen. This is accomplished by either using negative timebase delay or referencing the trigger point to the left side of the screen.

## NON-SYMETRICAL WAVEFORMS



*Figure A-3. Baseline Preceding Waveform Must Be Moved Off-screen to Find Correct Top of Signal*

---

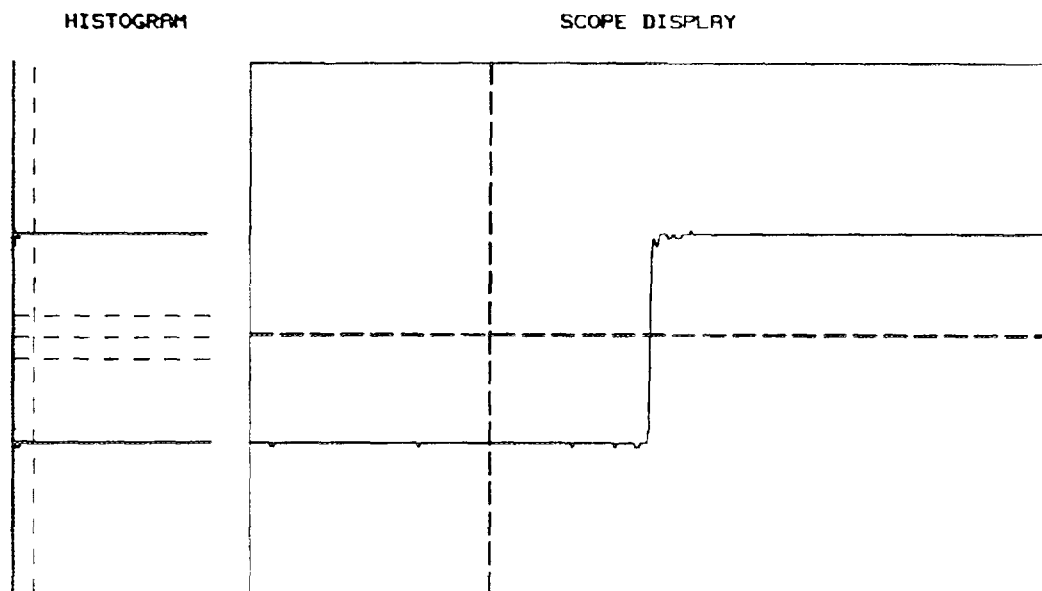
## Edge Find

Edge Find is used to place a time marker on the requested edge of a signal. If you are using automatic parametric measurements, the time markers will be placed on the signal's 50% point. If you are using edge find routines, the voltage markers determine the placement of the time markers. The following algorithm shows this in more detail.

1. Finds the Top – Base and moves the voltage markers to the 10, 50 or 90% points for automatic measurements. Edge measurements utilize existing voltage markers if not performing automatic measurements.
2. Selects a hysteresis level above and below each voltage marker. This limit is  $\pm 2$  A/D levels (1 minor division) and allows edges to be measured on a noisy signal.
3. If points lie between the two limits then a linear regression (curve fit) algorithm is invoked. Where this curve intersects the marker is the edge location.
4. If no points lie between the limits, a linear interpolation is performed using the closest points. This condition only occurs when there is little horizontal resolution (i.e., sweep speed too slow). The edge is considered to be the point where the voltage marker crosses the straight line generated by the linear interpolation.
5. Precision Edge-Find expands the target edge by decreasing the timebase range until the slope of the edge is 45 degrees.

The horizontal resolution (time per pixel) is determined by dividing the sweep range (time per 10 divisions) by 500. If the edge is represented by only one or two horizontal points, the instrument cannot accurately find the edge as in figure A-4. This problem can be surmounted by increasing the horizontal resolution to the point where five or more points are digitized on the edge itself.

## COARSE EDGE FIND



*Figure A-4. Edge of Waveform incorrectly found.  
(Increase Horizontal Resolution)*



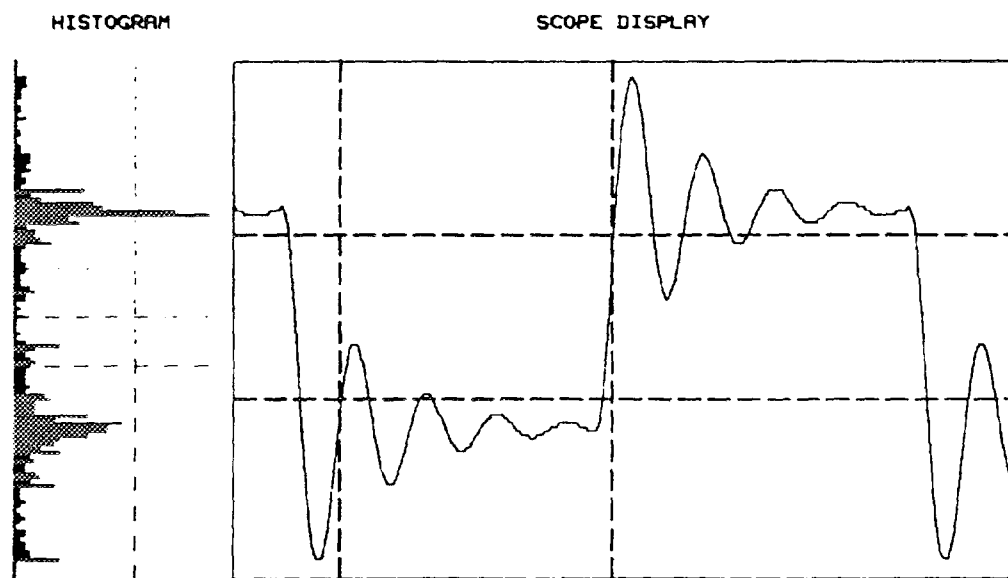
## Risetime

Risetime is measured from the first 10% point of a waveform to the first 90% point. The measurement invokes Top – Base, moves the voltage markers to the computed 10 and 90% points and performs an Edge Find, placing two time markers on the edge.

Accuracy can be improved by following the steps mentioned for the Top – Base and Edge Find routines. The Top and Base must be accurately found to allow precise positioning of the 10 and 90% markers. The horizontal resolution must be adequate to provide the curve-fit routines with sufficient data.

Risetime measurements are performed on the first positive on-screen transition. The measurement is made from the first on-screen 10% crossing to the first 90% crossing. Notice that the risetime is incorrectly measured in figure A-5. The risetime of this signal can be measured accurately if the signal is expanded to give increased horizontal resolution, which moves the unwanted 10% crossings off screen. A second solution is to use negative delay to accomplish the same task.

### RISE TIME GOTCHAS



*Figure A-5. Risetime Incorrectly Measured  
(Increase Horizontal Resolution)*

---

## Overshoot

One of the most commonly reported "bugs" is that the instrument measures overshoot at the bottom of the waveform and preshoot at the top. The definition of overshoot is the maximum excursion that occurs immediately after a transition. The polarity of the transition is irrelevant. If the first edge on screen is positive, overshoot will measure at the top of the waveform. If the first on-screen transition is negative, overshoot is measured at the bottom of the waveform. This conforms to IEEE standards.

## Program Examples

---

This chapter contains example programs using the command set for the HP 54120T. In general, these programs use the long form of the command and a separate output statement for clarity. To optimize speed, switch to the short form of the command.

Throughout these examples, the oscilloscope is assumed to be at address 7, the hardcopy devices at address 1, and the system bus at 700.

All programs were developed on an HP series 200 computer using HP BASIC 4.0. Some examples use the BASIC command "ENTER 2." This pauses program execution until the "ENTER" key is pressed 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 oscilloscope to finish an operation such as hardcopy output or an acquisition.

|  |      |
|--|------|
| Basic Commands .....                       | B-2  |
| TDR cal .....                              | B-5  |
| Pause Parameter Measurements .....         | B-8  |
| Fast Voltage Measurements .....            | B-10 |
| Graphics Printer .....                     | B-13 |
| Sending a Waveform to the Controller ..... | B-16 |
| Digitizing a Waveform on Channel 1 .....   | B-18 |
| Voltage Histogram .....                    | B-21 |
| Envelope Waveform Measurements .....       | B-24 |
| Custom Marker Risettime Measurements ..... | B-27 |

## Basic Commands

```
10 ! This program demonstrates the basic command structure used to program
20 ! the instrument. The TDR step generator is used as the test signal,
30 ! the SHORT should be removed from the CHANNEL 1 input prior to running
40 ! the program.
50
60
70 CONTROL CRT,12;1 !turn off the softkeys
80 ASSIGN @Scope TO 707
90 ASSIGN @Fast TO 707;FORMAT OFF
100 REAL Hi_prec(1:16),Lo_prec(1:16)
110 DIM Image$[32]
120
130
140 ! ***** setup the display *****
150 OUTPUT @Scope;"*RST" !put scope in known default state
160 OUTPUT @Scope;"DISPlay:GRATicule FRAME;FORMat 1"
170 OUTPUT @Scope;"VIEW CHANnel1;BLANK WMEMory2;BLANK WMEMory4"
180 OUTPUT @Scope;"ACQuire:TYPe AVERage;COUNT 8;POINts 1024"
190 OUTPUT @Scope;"DISPlay:ROW 2;COL 0;COLOr 7;ATTRibute ENABle"
200
210
220 ! ***** turn on the step generator *****
230 OUTPUT @Scope;"NETWork:REFLection:STEP CHAN1"
240 OUTPUT @Scope;"NETWork:REFLection:PRESet"
250
260
270 ! ***** scale the timebase *****
280 OUTPUT @Scope;"TIMebase:GRATe 500e3;RANGe 500e-8"
290
300
310 BEEP
320 WAIT 1
330
340
350 ! ***** measure the period and get 1 period on screen *****
360 OUTPUT @Scope;"DISP:STRing ""MEASURE PERIOD and SCALE TIMEBASE""
370 OUTPUT @Scope;"MEASure:PERiod?"
380 ENTER @Scope;Period
390 OUTPUT @Scope;"TIMebase:RANGe ";Period*1.1
400
410
420 BEEP
```

```

430 WAIT 1
440
450
460 ! ***** find the 1st negative edge and center waveform *****
470 OUTPUT @Scope;"DISPlay:TEXT BLANK;STRing ""CENTER the WAVEFORM""
480 OUTPUT @Scope;"MEASure:ESArt -1;TStArt?"
490 ENTER @Scope;Neg_edge
500 Delay_time=Neg_edge-(.05*Period) !delay needed to center signal
510 OUTPUT @Scope;"TImebase:DElay ";Delay_time
520 OUTPUT @Scope;"DISPlay:VMARker OFF;TMARker OFF" !turn off the markers
530 OUTPUT @Scope;"*SAV 0" !save this setup for later use
540
550
560 BEEP
570 WAIT 1
580
590
600 ! ***** move the waveform horizontally *****
610 OUTPUT @Scope;"DISPlay:TEXT BLANK;STRing ""TIMEBASE DELAY""
620 FOR I=Delay_time TO 4.E-6 STEP 1.0E-7
630     OUTPUT @Scope;"TImebase:DElay ";I
640     WAIT .29
650 NEXT I
660
670
680 BEEP
690 WAIT 1
700
710
720 ! ***** move the signal vertically *****
730 OUTPUT @Scope;"DISPlay:TEXT BLANK;STRing ""VERTICAL OFFSET""
740 OUTPUT @Scope;"*RCL 0" !recall setup
750 FOR I=-3.2E-1 TO 5.00E-1 STEP 3.0E-2
760     OUTPUT @Scope;"CHANnel:OFFSet ";I
770     WAIT .3
780 NEXT I
790
800
810 BEEP
820 WAIT 1
830
840
850 ! ***** examine the waveform top *****
860 OUTPUT @Scope;"DISPlay:TEXT BLANK;STRing ""VERTICAL SENSITIVITY""
870 OUTPUT @Scope;"*RCL 0" !recall setup
880 OUTPUT @Scope;"MEASure:VTOp?" !find the pulse top
890 ENTER @Scope;VOLT_top
900 OUTPUT @Scope;"CHANnel:OFFSet ";VOLT_top !move pulse top to center screen
910 FOR I=8.0E-2 TO 0.E+0 STEP -(4.E-3)
920     OUTPUT @Scope;"CHANnel:RANGe ";I*8

```

```

930     WAIT .5
940     NEXT I
950     OUTPUT @Scope;"ACQuire:COUNT 64"
960     OUTPUT @Scope;"CHANnel:RANGe 8E-3"
970
980
990     BEEP
1000    WAIT 1
1010
1020
1030    !***** measure full parametrics *****
1040    OUTPUT @Scope;"DISPlay:TEXT BLANK;STriNg ""PARAMETRIC MEASUREMENTS""
1050    OUTPUT @Scope;"*RCL 0"
1060    DATA "FREQ","PER ","PWID","NWID","RIS ","FALL","VAMP","VPP ","PRES"
1070    DATA "OVER","DUT ","VRMS","VMAX","VMIN","VTOP","VBAS"
1080    OUTPUT @Scope;"TImebase:RANGe ";6*Period
1090
1100    !***** precision measurements *****
1110    OUTPUT @Scope;"SYSTem:HEADer OFF;LONGform OFF"
1120    OUTPUT @Scope;"MEASure:SOURce CHANnel;PRECision FINE;ALL?"
1130    ENTER @Scope;Hi_prec(*)
1140
1150    ! ***** coarse measurements *****
1160    OUTPUT @Scope;"MEASure:PREC COARse;ALL?"
1170    ENTER @Scope;Lo_prec(*)
1180    PRINT CHR$(128)
1190    PRINT USING "11X,4A,12X,6A,10X,5A";"FINE","COARSE","DELTA"
1200    FOR I=1 TO 16
1210        READ Head$
1220        Image$="4A,3X,2(SD.5DE,5X),SD.2DE"
1230        PRINT USING Image$;Head$,Hi_prec(I);Lo_prec(I);Hi_prec(I)-Lo_prec(I)
1240    NEXT I
1250
1260
1270    END

```

## TDR cal

```
10  !-----
20  !   This program does a TDR cal and makes measurements on the live
30  !   reflection waveform. As part of the calibration process,
40  !   prompts are written to the oscilloscope display by the controller.
50  !
60  !   Scope setup: There should be no signals connected to the scope
70  !   inputs. A short and a 50 ohm ref. will be needed.
80  !-----
90  !
100 REAL Min,Max,Distance,Refl_percent,Impedance,Delta_t
110 !
120 !
130 CLEAR 707                                !Initialize scope HP-IB
140                                !interface registers.
150 !
160 OUTPUT 707;"ACQUIRE:TYPE AVERAGE;COUNT 16" !Select avg by 16 mode
170                                !Most accurate results
180                                !obtained in average mode.
190 !
200 !
210 ! Text written to the scope display by the controller will be in row 15
220 ! starting at column 0 in the beige highlight color (0). Row 15 is
230 ! where the scope normally writes prompts and advisories.
240 !
250 !
260 OUTPUT 707;"DISPLAY:ROW 15;COLUMN 0;COLOR 0"
270 !
280 !-----
290 ! Setup Channel 1 as the TDR channel, then set timebase to 500 ps/div
300 !-----
310 !
320 !
330 OUTPUT 707;"NETWORK:REFLECTION:STEP CHANNEL1" !Chan1 is reflection chan.
340 !
350 OUTPUT 707;"NETWORK:REFLECTION:PRESET;:TIMEBASE:RANGE 5 ns"
360 !
370 !
380 !-----
390 ! Begin TDR calibration
400 !-----
410 !
420 ! Prompt user with text on scope display to connect a short at ref plane
430 !
440 !
```

```

450 OUTPUT 707;"DISP:STR 'Connect short and push ENTER on computer keyboard'"
460
470 ENTER 2                                !Cause program to pause. Push
480                                         !ENTER on computer keyboard to
490                                         !continue.
500 OUTPUT 707;"DISPLAY:TEXT BLANK"        !Erase prompt from scope screen
510
520 OUTPUT 707;"NETWORK:CALIBRATE SHORT"    !Do the TDR short (0 ohm) cal
530
540
550 ! Prompt user with text on scope display to connect 50 ohms at ref plane
560
570
580 OUTPUT 707;"DISPLAY:TEXT BLANK"        !Erase prompt scope writes to
590                                         !screen for 50 ohm cal 1st.
600
610 OUTPUT 707;"DISP:STR 'Connect 50 ohms at ref plane and press ENTER again'"
620
630 ENTER 2                                !Wait for ENTER key to be pushed
640
650 OUTPUT 707;"NETWORK:CALIBRATE FIFTY"    !Do TDR fifty ohm cal
660
670
680 !-----
690 ! TDR calibration is now complete
700 !-----
710 !
720 ! Prompt user with text on scope display to connect a test device at ref
730 ! plane (0 or 50 ohms will work). Best results will be obtained if the
740 ! waveform is allowed to converge before the ENTER key is pushed.
750
760
770 OUTPUT 707;"DISP:STR 'Connect test device at ref plane and press ENTER '"
780
790 ENTER 2                                !Wait for user to connect device
800
810 OUTPUT 707;"DISPLAY:TEXT BLANK"        !Erase text written to screen
820
830
840 !-----
850 ! Select Channel 1 as the reflection measurement source
860 !-----
870 !
880 ! If Ch3 were the reflection channel, it would be selected as the source.
890 ! To make reflection measurements on normalized TDR waveforms, Wmemory1
900 ! would be selected as the reflection measurement source.
910
920
930 OUTPUT 707;"NETWORK:REFLECTION:SOURCE CHANNEL1" !Ch1 = reflection
940                                         !measurement source

```



```

950
960 OUTPUT 707;"NETWORK:REFLECTION:Cursors 500 ps" !Move network cursor to
970 !500 ps from ref plane on
980 !channel 1.
990
1000 OUTPUT 707;"NETWORK:REFLECTION:PARAMETERS?" !Measure min & max
1010 !percent reflection
1020
1030 ENTER 707;Max,Min !Read measurement results
1040
1050
1060 !-----
1070 ! Make same measurements made by the TDR cursor from the front panel
1080 !-----
1090 !
1100 ! NOTE: Distance measurements are only as accurate as the currently
1110 ! specified dielectric constant. Assume dielectric constant
1120 ! is 1 for this example (corresponds to vacuum).
1130
1140
1150 OUTPUT 707;"NETWORK:DIELECTRIC 1" !Dielectric constant = 1
1160
1170 OUTPUT 707;"NETW:REFL:DISTANCE?;IMPEDANCE?;RHO?;CURSOR?"
1180
1190 ENTER 707;Distance,Impedance,Refl_percent,Delta_t !Read measurement results
1200
1210
1220 LOCAL 707 !Return scope to local
1230 !mode
1240 END

```

## Pulse Parameter Measurements

```

10      ! This program will obtain a stable display of a waveform input to
20      ! channel 1. Up to 16 pulse parameter measurements will then be
30      ! made (depending on the waveform). At the end, the voltage markers
40      ! are placed at the 15 and 85 % levels on the waveform.
50      !
60      !      Required scope setup:      Input signal to channel 1  640 mV p-p
70      !                                     Frequency of input  50 Hz
80      !                                     Good results will be obtained with sine
90      !                                     waves and square waves.
100     !                                     External trigger signal must be supplied
110     !
120     !
130     CLEAR 707                                     !Initialize scope HP-IB interface
140     !
150     ALLOCATE REAL Results(0:15)                   !Will hold measurement results
160     !
170     OUTPUT 707;"AUTOSCALE"                         !Set up vertical channel,
180     !                                     !timebase, and trigger
190     !
200     OUTPUT 707;"ACQUIRE:TYPE AVERAGE;COUNT 8" !Select avg mode for more
210     !                                     !repeatable results
220     !
230     OUTPUT 707;"SYSTEM:HEADER OFF"                 !Don't return header with meas.
240     !                                     !results. Header should always be
250     !                                     !off when reading data into a
260     !                                     !numeric array. Header strings may
270     !                                     !confuse controller input formatter
280     !
290     OUTPUT 707;"MEASURE:SOURCE CHANNEL1"           !Do measurements on channel 1
300     !
310     OUTPUT 707;"MEASURE:PRECISION FINE"            !Fine= Expand timebase as necessary
320     !                                     !to make timing measurements
330     !                                     !as accurate as possible.
340     OUTPUT 707;"MEASURE:ALL?"                      !Measure up to 16 signal parameters
350     !
360     ENTER 707;Results(*)                           !Read results into Results array
370     !
380     !
390     ! Order of measurement results in Results array:
400     !
410     !      Freq, period, +width, -width, risetime, falltime, Vamp, Vpp,
420     !      preshoot, overshoot, dutycycle, VRMS, Vmax, Vmin, Vtop, Vbase

```

```

430  !
440  ! For example: Results(0) = Frequency  Results(1) = Period ...
450
460  ! Print some of the measurement results
470
480  PRINT "Frequency = ",Results(0)," Hz"
490  PRINT "Risetime = ",Results(4)," seconds"
500
510
520  ! Set voltage markers at 15 and 85%
530
540  OUTPUT 707;"MEASURE:PBASE 15;PTOP 85;VMARKERSET"
550
560  LOCAL 707                                !Return scope to local mode
570  END

```

## Fast Voltage Measurements

```

10  !-----
20  ! This program makes very fast and very accurate voltage measurements
30  ! at specific points in time by taking full advantage of the 54120's
40  ! sequential sampling technique. The 54120's timebase can be
50  ! programmed to acquire data at one specific point in time. Since data
60  ! is acquired only at the specified time, this is a very efficient way
70  ! to make voltage measurements. This capability is provided with the
80  ! "voltage at a time" (VTIME?) query.
90  !
100 ! Voltage and time markers will intersect on scope CRT to show where
110 ! voltage measurement was made.
120 !
130 ! Scope setup: Channel 1 = Periodic waveform 50 Hz, 640 mV p-p
140 !           External trigger signal connected
150 !
160 ! Variables used in this program:
170 !     Time = Time at which voltage measurements will be made
180 !     Delay= Current scope delay setting
190 !     Voltage= Voltage measured at the time held in the Time variable
200 !     Range= Current timebase range (= Time/div * 10)
210 !-----
220                                     !
230                                     !
240 REAL Time,Delay,Voltage,Range
250                                     !
260 CLEAR 707                          ! Initialize scope HP-IB interface
270                                     !
280 OUTPUT 707;"AUTOSCALE"             ! Setup timebase, vertical channel, & trigger
290                                     !
300                                     !
310 OUTPUT 707;"ACQUIRE:TYPE NORMAL"  ! Use persistence display mode
320 OUTPUT 707;"DISPLAY:PERSISTENCE 300E-3" ! Select 300 ms persistence time
330                                     ! Persistence is the fastest
340                                     ! acquisition mode for VTIME?
350                                     ! measurements.
360                                     !
370 OUTPUT 707;"TIMEBASE:RANGE?;DELAY?" !Request current settings
380                                     !
390 ENTER 707;Range,Delay               !Read current Range and Delay
400                                     !
410 Time=Delay+(Range/10)               ! Make measurement at current delay + 1 division
420                                     !
430 WAIT .1                             ! Wait .1 s for data to build up for next example

```

```

440 !
450 !
460 !-----!
470 ! The FASTEST way to make a voltage measurement is to use the VTIME?
480 ! query with the measurement precision coarse. In this mode, the
490 ! scope will return the voltage value on screen which was acquired
500 ! closest to the specified time. See documentation for VTIME?
510 ! in command summary section of programming manual for details.
520 !-----!
530 !
540 !
550 OUTPUT 707;"MEASURE:PRECISION COARSE;SOURCE CHANNEL1" !Use coarse
560 !precision. Measure
570 !data on channel 1.
580 !
590 !
600 OUTPUT 707;"MEASURE:VTIME? ";Time !Note space after "?" is mandatory
610 !
620 ENTER 707;Voltage !Read voltage at "Time"
630 !
640 !
650 !-----!
660 ! To make measurements which are still very fast but which are also
670 ! precisely at the specified time, change measurement precision to
680 ! fine. If the measurement source is a channel and precision is
690 ! fine, data is automatically reacquired precisely at the specified
700 ! time, regardless of what data is on screen. See description
710 ! of VTIME? in programming manual for restrictions.
720 !-----!
730 !
740 !
750 Time=Time+(Range/10) ! Measure V at Delay + 2 div
760 !
770 OUTPUT 707;"MEASURE:PRECISION FINE;VTIME? ";Time !Change to fine precision
780 !and start measurement
790 !
800 ENTER 707;Voltage ! Read voltage at "Time"
810 !
820 !
830 !
840 !-----!
850 ! To make the MOST ACCURATE VTIME measurements possible in the minimum
860 ! amount of time, select average by 2048 mode. During the VTIME
870 ! measurement, 2048 samples will be acquired and averaged at the
880 ! specified time. Observe that because the scope can acquire data
890 ! exactly where desired on every trigger, the measurement is made
900 ! very quickly even with a large number of averages selected.
910 !-----!
920 !
930 !

```

```

940  OUTPUT 707;"ACQUIRE:TYPE AVERAGE;COUNT 2048" !Average by 2048 mode
950                                         !
960  Time=Time+(Range/10)                !Measure V at Delay + 3 div
970                                         !
980  OUTPUT 707;"MEASURE:VTIME? ";Time    !Request voltage at "Time"
990                                         !
1000 ENTER 707;Voltage                   !Read voltage at "Time"
1010                                         !
1020                                         !
1030 LOCAL 707                          !Return scope to local
1040                                         !
1050  END

```

## Graphics Printer

```

10  ! -----
20  !   This program sends data from the active scope display to a
30  !   graphics printer. A service request (SRQ) is used to signal
40  !   the completion of the data transfer to the printer.
50  !
60  !   The scope should already be setup with waveform data on screen.
70  !   The graphics printer should be connected via HP-IB and should
80  !   be set at address 1. The program assumes a black and white printer.
90  ! -----
100 !
110 CLEAR 707                      ! Clear scope HP-IB interface
120 !
130 OUTPUT 707;"*CLS"              ! Clear all scope status registers
140 !
150 !
160 ! -----
170 !   Use the service request (SRQ) capabilities of the scope to determine
180 !   when the data transfer to the printer is complete. Attempting to
190 !   send commands to the 54120 while the data transfer to the printer is
200 !   in progress will cause errors.
210 !
220 !   The following code will set up the scope to generate a service request
230 !   when the data transfer to the printer is completed.
240 ! -----
250 !
260 !
270 ON INTR 7 GOSUB Srq_serv ! Call Srq_serv if interrupt comes in over HP-IB
280 !
290 OUTPUT 707;"*ESE 1"          ! Enable setting of the event status bit in
300                               ! status byte register when "operation complete"
310                               ! event occurs.
320 !
330 OUTPUT 707;"*SRE 32"        ! Generate a SRQ when the event status bit=1
340 !
350 Done=0                       ! Done will be set = 1 by Srq_serv when data
360                               ! transfer to printer is complete.
370 !

```

```

380  !-----
390  !   Tell scope to print the active display and scale factors on
400  !   a black and white printer and to send a form feed after the
410  !   printout is completed.
420  !-----
430                                     !
440  OUTPUT 707;"HARDCOPY:SOURCE PMEMORY0,FACTORS"  !Print active display and
450                                               !associated scale factors
460                                     !
470  OUTPUT 707;"HARDCOPY:PRINTER DEFAULT;PAGE AUTOMATIC" !Specify black &
480                                               !white printer and request
490                                               !a formfeed at end of
500                                               !printout.
510                                     !
520  OUTPUT 707;"PRINT?;*OPC"      !Tell scope to buffer the data to be sent to
530                               !the printer and to set the operation
540                               !complete bit in the event status register
550                               !when data transfer to the printer is
560                               !complete.
570                                     !
580  SEND 7;UNT UNL                ! Unaddress bus. Assert ATN line
590  SEND 7;TALK 7                 ! Address the scope to talk
600  SEND 7;LISTEN 1              ! Address the printer to listen
610  SEND 7;DATA                  ! Negate the ATN line (starts data transfer)
620                                     !
630  ENABLE INTR 7;2              ! Tell controller to accept SRQ interrupts
640                                     !
650                                     !
660  Print_wait:                  !
670  IF Done=0 THEN GOTO Print_wait !Wait for data transfer to printer to
680                               !finish. Done set=1 in Srq_serv when
690                               !service request is received.
700                                     !
710                                     !
720  LOCAL 707                    ! Return scope to local mode
730  GOTO End_prog                ! Exit program
740                                     !
750                                     !
760  !-----
770  !   The following is the service request (SRQ) interrupt service routine
780  !-----
790                                     !
800                                     !
810  Srq_serv:                    !
820                                     !
830  !   Since "operation complete" is the only event enabled in the

```



```

840 ! event status enable register there is no need to examine the
850 ! event status register to determine the cause of the service request.
860 !
870 Stat=SPOLL(707) ! Perform a serial poll of the scope to clear the SRQ
880 ! Stat will hold the scope status byte
890 !
900 Done=1 ! Signal that transfer to printer is done
910 !
920 OUTPUT 707;"*CLS" ! Clear all scope status registers
930 RETURN
950 End_prog:
960 !
970 END

```

---

## Sending a Waveform to the Controller

```
10      ! This sample program digitizes a waveform on channel 1 and
20      ! transfers it from the 54120 to the controller.
30      !
40      !
50      ! Required scope setup:  Signal input to channel 1  640 mV p-p
60      !                               Frequency  50 Hz
70      !                               External trigger input connected
80
90
100     ASSIGN @Scope TO 707;FORMAT OFF          ! Turn off BASIC input formatters
110                                           ! Accept word binary data "as is"
120                                           !
130     CLEAR 707                                !Initialize scope HP/IB interface
140                                           !
150     ALLOCATE INTEGER Waveform(0:499)         !Will hold raw waveform data
160     DIM Voltages(0:499)                     !Waveform data values in Volts
170     INTEGER Yref                             !Described in detail when used
180     REAL Yor,Yinc                             !
190
200
210     OUTPUT 707;"AUTOSCALE"                   !Setup vertical, timebase & trigger
220
230     OUTPUT 707;"ACQUIRE:TYPE NORM;POINTS 500" !Change scope to persistence
240                                           !display mode (fastest mode
250                                           !for digitizing a waveform
260                                           !with the DIGITIZE command).
270                                           !Waveform record length=500 pts
280
290     OUTPUT 707;"DIGITIZE CHANNEL1"           !Acquire data on channel 1 and
300                                           !store it in waveform memory 1.
310
320     OUTPUT 707;"SYSTEM:HEADER OFF"          !Turn off headers in query replies
330                                           !They provide no useful data for this
340                                           !example & increase data transfer time
350
360
370     ! Next commands request data in waveform memory 1 to be transferred
380     ! from the oscilloscope to the controller.  Data will be sent in
390     ! word format for maximum transfer speed.
400
410
```

```

420 OUTPUT 707;"WAVEFORM:SOURCE WMEMORY1;FORMAT WORD"
430 OUTPUT 707;"WAVEFORM:DATA?" !Actual request for data
440
450
460 ! Read data from oscilloscope into controller
470
480 ENTER 707 USING "#,6A";Length$ !#= no linefeed terminator
490 ! necessary
500 ! 6A= read 6 ASCII characters
510 ! into Length$
520 ! Characters will be:
530 ! "#3500 "
540
550 ENTER @Scope;Waveform(*) ! Read waveform data
560
570 ENTER 707 USING "-K,B";End$ ! Read characters until a
580 ! linefeed is encountered then
590 ! read the last byte (the
600 ! linefeed char) into End$
610
620
630
640 ! Read parameters from scope needed to convert raw waveform data to volts
650 !
660 ! YINCREMENT = Voltage difference between consecutive data values
670 ! YORIGIN = Voltage at mid-screen = Offset voltage
680 ! YREFERENCE = Data value where Y origin occurs
690
700
710 OUTPUT 707;"WAVEFORM:YINCREMENT?;YORIGIN?;YREFERENCE?"
720 ENTER 707;Yinc,Yor,Yref ! Read values returned by scope
730
740
750 ! Convert raw waveform data values to voltages and store in Voltages array
760
770
780 FOR I=0 TO 499
790 Voltages(I)=((Waveform(I)-Yref)*Yinc)+Yor
800 NEXT I
810
820
830 LOCAL 707 ! Return 54120 to local
840 END ! operation

```

## Digitizing a Waveform on Channel 1

```

10  !-----
20  ! This program digitizes a waveform on Ch1, transfers it from the scope
30  ! (waveform memory 1) to the controller, and then from the controller
40  ! back into the scope (waveform memory 4). The sections actually
50  ! involved with the transfers are written to work with any length
60  ! waveform of type average or normal.
70  !
80  ! Scope setup: A stable display of signal input to channel 1
90  !
100 !-----
110                                     !
120 PRINTER IS 1                      !PRINTs go to computer display
130                                     !
140 ALLOCATE REAL Preamble(0:10)      !Will hold waveform data preamble
150                                     !Preamble holds scaling info
160 CLEAR 707                        !Initialize scope HP-IB interface
170                                     !
180 OUTPUT 707;"SYSTEM:HEADER ON;LONGFORM OFF" !Shortforms of headers will
190                                     !be returned by scope.
200 OUTPUT 707;"ACQUIRE:TYPE NORMAL;POINTS 1024" !Acquire a "persistence" type
210                                     !waveform, 1024 pts long.
220 OUTPUT 707;"DIGITIZE CHANNEL1"      !Ch1 waveform data - Wmem1
230                                     !
240                                     !
250 !-----
260 ! Transfer waveform data from scope (wmemory1) to controller
270 !-----
280                                     !
290                                     !
300 OUTPUT 707;"WAVEFORM:SOURCE WMEMORY1" !Data transfer from wmemory1
310                                     !
320 OUTPUT 707;"WAVEFORM:PREAMBLE?"      !Tell scope to send preamble
330 ENTER 707;Preamble(*)               !Read preamble
340 PRINT Preamble(*)                   !Print it on computer screen
350                                     !
360 Length=Preamble(2)                  !Length=# points in waveform
370                                     !data record. See WAVEFORM
380                                     !subsystem in programming
390                                     !manual for preamble format.
400 ALLOCATE INTEGER Wfm(1:Length)      !Create waveform array of
410                                     !appropriate length

```

```

420 OUTPUT 707;"WAVEFORM:FORMAT WORD;DATA?"      !Tell scope to send data in
430                                              !binary word format
440 ENTER 707 USING "#,11A,D";First11$,Bytes      !Read header= ":wav:data #X"
450                                              !First11$ = 1st 11 characters
460                                              !Bytes = X = # ASCII bytes
470                                              !          used to specify
480                                              !          # waveform data
490                                              !          bytes which
500                                              !          follow.
510
520
530 IF Bytes=3 THEN ENTER 707 USING "#,3A";Length$ !Read 3 ASCII characters
540                                              !which specify number of
550                                              !waveform data bytes to
560                                              !follow.
570 IF Bytes=4 THEN ENTER 707 USING "#,4A";Length$ !Read 4 ASCII characters
580                                              !which specify number of
590                                              !waveform data bytes to
600                                              !follow.
610
620 PRINT First11$&VAL$(Bytes)&Length$             !Print header
630
640 ! Read in the waveform data bytes. The waveform data will be in binary
650 ! format so the BASIC input formatter must be turned off. To do this,
660 ! a special "path" (called @Scope below) must be assigned to talk with
670 ! the scope.
680
690 ASSIGN @Scope TO 707                          !Assign special path
700 ASSIGN @Scope;FORMAT OFF                      !Turn off BASIC input
710                                              !formatter.
720 ENTER @Scope;Wfm(*)                          !Read waveform data
730 ASSIGN @Scope;FORMAT ON                      !Turn BASIC input formatter
740                                              !back on.
750 ENTER 707 USING "-K,B";End$                  !Read linefeed
760
770
780 !-----
790 ! Transfer waveform data back to scope (wmemory4) from controller
800 !-----
810
820
830 ! Put scope in dual screen display mode and turn on waveform memories
840 ! 1 and 4 so they can be compared before and after the data transfer.
850
860
870 OUTPUT 707;"DISPLAY:FORMAT 2;:VIEW WMEMORY1;:VIEW WMEMORY4"
880
890
900 OUTPUT 707;"WAVEFORM:SOURCE WMEMORY4"        !Data will be sent to wmemory4
910

```

```

920  OUTPUT 707;"WAVEFORM:PREAMBLE ";Preamble(*) !Send in new wmem4 preamble
930
940
950  ! Send header and waveform data to scope (wmemory4)
960
970
980  OUTPUT 707 USING "#,K";First11$&VAL$(Bytes)&Length$ !Send header =
990                                     ! :wav:data #42048
1000
1010  ASSIGN @Scope;FORMAT OFF           !Turn off output formatter.
1020                                     !Scope expects binary data.
1030
1040  OUTPUT @Scope;Wfm(*)               !Send out waveform data
1050
1060  ASSIGN @Scope;FORMAT ON           !Turn output formatter back on
1070
1080  OUTPUT 707 USING "#,K";CHR$(10)    !Output a linefeed to complete
1090                                     !the binary data transfer.
1100
1110
1120  LOCAL 707                         !Return scope to local
1130  DEALLOCATE Wfm(*)                  !Release memory to system
1140  DEALLOCATE Preamble(*)
1150  END

```

## Voltage Histograms

```

10  !-----
20  ! This example program acquires a voltage histogram, measures the mean
30  ! and standard deviation of the whole histogram and then of a subsection
40  ! of the histogram, transfers the histogram data up to the controller,
50  ! and finally plots the histogram data on the controller display.
60  !
70  ! Scope setup : Disconnect all front panel inputs.
80  !               TDR step output is used as a signal source
90  !
100 !-----
110 !
120 CLEAR 707                                !Clear 54120 HP-IB interface
130 !
140 PRINTER IS 1                             !PRINT commands send data to
150                                         !controller display.
160 !
170 OUTPUT 707;"network:reflection:preset"    !Step generator on. Ch1=TDR chan
180                                         !Setup display parameters
190 OUTPUT 707;"timebase:range 5 ns"          !Expand time scale to 500 ps/div
200 !
210 !-----
220 ! Prepare to generate a voltage histogram. Specify source=channel1,
230 ! number of samples to acquire in time window, and time window itself.
240 !-----
250 !
260 OUTPUT 707;"histogram:source channel1"      !Use data from channel1
270 !
280 OUTPUT 707;"acquire:type histogram;count 25000" !Acquire histogram with
290                                         !25000 samples in window
300 !
310 OUTPUT 707;"histogram:window time,16ns,19ns" !Voltage histogram of data
320                                         !between 16 ns and 19 ns
330                                         !on channel 1.
340 !
350 OUTPUT 707;"digitize channel1;view wmemory5" !Acquire histogram data
360                                         !and display it on CRT
370 !
380 !
390 !-----
400 ! Histogram acquisition is complete. Measure mean and sigma (standard
410 ! deviation) of the whole distribution.
420 !-----
430 !

```

```

440                                     !
450 OUTPUT 707;"histogram:mean?;sigma?"    !Tell scope to measure mean & sigma
460                                     !
470 ENTER 707;Mean,Sigma                  !Read results
480 PRINT "          Mean=",Mean,"Sigma=",Sigma !Write mean & standard deviation
490                                     !to controller display.
500                                     !
510 !-----
520 ! Compute the mean and standard deviation of the portion of the histogram
530 ! between 150 mV and 250 mV.
540 !-----
550                                     !
560 OUTPUT 707;"histogram:reference 150mV,250mV" !Only use histogram data
570                                     !between 150 mV and 250 mV
580                                     !for histogram measurements
590                                     !
600 OUTPUT 707;"histogram:mean?;sigma?"    !Ask for mean & standard
610                                     !deviation.
620 ENTER 707;Mean,Sigma                  !Read results from scope
630 PRINT "          Mean=",Mean,"Sigma=",Sigma !Display results
640                                     !
650 !-----
660 ! Transfer histogram data from 54120 to controller
670 !-----
680                                     !
690 ! Histogram data is in Wmemory5. In this example, data is transferred in
700 ! ASCII format to simplify the code. Long format should be used if data
710 ! transfer time must be minimized. Long format transfer is very similar
720 ! to word format. See example programs which transfer data in word format
730                                     !
740 OUTPUT 707;"waveform:source wmemory5;format ascii;"
750                                     !
760 OUTPUT 707;"system:header off"          !No headers to precede data
770                                     !
780 OUTPUT 707;"waveform:data?"            !Ask for histogram data
790                                     !
800 ALLOCATE REAL Histogram(1:256)         !Create array to hold data
810                                     !
820 ENTER 707;Histogram(*)                 !Read data into array
830                                     !
840                                     !
850 !-----
860 ! Plot the histogram data on the controller display.
870 !-----
880                                     !
890 ! Determine the largest entry in the histogram array so
900 ! the controller display can be scaled properly
910                                     !
920 Max=-1                                  !Invalid value. Min legal value=0
930 FOR I=1 TO 256                          !Check all 256 histogram entries.

```



```

940
950   IF Histogram(I)Max THEN Max=Histogram(I) !If current histogram entry is
960                                           !larger than largest entry so
970                                           !far- It becomes the new max
980   NEXT I
990
1000 ! Plot the histogram
1010
1020 GINIT                                !Initialize graphics
1030 GCLEAR                              !Clear any previous data from screen
1040 GRAPHICS ON                          !Make sure graphics are on
1050 WINDOW -Max/20,Max,-25,256          !Scale the computer screen
1060
1070 MOVE 0,0
1080 DRAW 0,256                          !Draw vertical arm of graticule
1090 MOVE 0,0
1100 DRAW Max,0                          !Draw horizontal arm of graticule
1110
1120 CSIZE 5,.6                          !Character size for labels
1130 MOVE .3*Max,-20                     !Label location along x axis
1140 LABEL "Number of samples"           !Label for horizontal axis
1150 MOVE -Max/100,120                   !Put next label 1/2 way up y axis
1160 DEG                                 !Angle for LDIR command in degrees
1170 LDIR 90                             !Write label parallel to y axis
1180 LABEL "Volts"
1190
1200 FOR I=1 TO 256
1210   IF Histogram(I)=0 THEN             !Don't plot anything if entry=0
1220     MOVE 0,I                         !Move to correct spot on Y axis
1230     DRAW Histogram(I),I             !Plot actual histogram value
1240   END IF
1250 NEXT I
1260
1270 LOCAL 707                           !Return scope to local mode
1280 DEALLOCATE Histogram(*)             !Release memory back to system
1290 END

```

---

## Envelope Waveform Measurements

```
10  !-----
20  ! This program digitizes a waveform of type envelope, transfers the
30  ! envelope waveform data from the scope to the controller, and converts
40  ! the waveform data values to voltages.
50  !
60  ! Scope setup: External signal supplied to Ch1 640 mV p-p and 50 Hz
70  ! External trigger signal must be supplied
80  !
90  !-----
100 !
110 CLEAR 707                                !Initialize scope HP-IB interface
120 !
130 OUTPUT 707;"AUTOSCALE"                    !Setup Ch1 vertical scale, timebase
140 !and trigger.
150 !
160 !
170 !-----
180 ! Acquire a waveform of TYPE ENVELOPE. The acquire COUNT specifies how
190 ! many samples to take at each time point on the waveform before
200 ! considering the acquisition at that point to be complete. The COUNT
210 ! command provides a way of specifying how long the scope should wait to
220 ! allow the waveform envelope to build up.
230 !
240 ! In this example, 64 samples will be taken and used to create the
250 ! envelope at each of the 256 time POINTS. Note that a shorter waveform
260 ! record (fewer points) will reduce the amount of time required for
270 ! for data acquisition and data transfer to the controller.
280 !-----
290 !
300 !
310 OUTPUT 707;"ACQUIRE:TYPE ENVELOPE;COUNT 64;POINTS 256" !Specify a waveform
320 !envelope 256 pts
330 !long. Take 64
340 !samples at each
350 !point.
360 OUTPUT 707;"DIGITIZE CHANNEL1"            !Acquire waveform
370 !envelope.
380 !
390 !
400 !-----
```

```

410 ! Transfer the waveform envelope from the scope to the controller.
420 ! The envelope actually consists of 2 waveform data arrays. Waveform
430 ! memory1 holds the minimum value sampled at each time point on Ch1.
440 ! Waveform memory3 holds the maximum value sampled at each time point.
450 ! The memories are time correlated. The value point in wmem1 is the
460 ! minimum value found at the 1st time point and the 1st value in wmem3 is
470 ! the maximum value found at the 1st time point.
480 !
490 ! Both waveform data arrays are transmitted one at a time over the
500 ! HP-IB in response to the WAVEFORM:DATA? query. The array of minimum
510 ! values is sent first. The value returned from a WAVEFORM:POINTS? query
520 ! is the number of points in each of the arrays.
530 !-----
540 !
550 !
560 OUTPUT 707;"WAVEFORM:SOURCE WMEMORY1;FORMAT ASCII" !Because Wmem1 is of
570 !type envelope, the
580 !scope knows to send
590 !both Wmem1 & 3 even
600 !though only Wmem1 is
610 !listed as the source.
620 !Data will be sent in
630 !ASCII format.
640 OUTPUT 707;"WAVEFORM:POINTS?" !Request length of waveform data arrays
650 !
660 ENTER 707;Array_len !Array_len = # of values in each array
670 !
680 ! Allocate memory for the minimum and maximum waveform data arrays
690 !
700 ALLOCATE REAL Min(1:Array_len),Max(1:Array_len)
710 !
720 !
730 OUTPUT 707;"SYSTEM:HEADER OFF" !Always turn headers off when reading
740 !data from scope into numeric variables.
750 !
760 OUTPUT 707;"WAVEFORM:DATA?" !Tell scope to send waveform data
770 !
780 ENTER 707 USING "#,-K";Min(*) !Read minimum waveform data array
790 !#= Read values only until array is full
800 ! No linefeed terminator is necessary.
810 !-K = Don't require a LF between values
820 !
830 ENTER 707 USING "-K";Max(*) !Read maximum waveform data array
840 !
850 !
860 !-----
870 ! Read parameters from scope needed to convert raw waveform data values
880 ! to voltages.
890 !
900 ! Yinc = Voltage difference between consecutive data values

```

```

910 !   Yorg = Voltage at mid-screen = Offset voltage
920 !   Yref = Data value where Y origin (Yorg) occurs
930 !-----
940                                     !
950                                     !
960 REAL Yinc,Yorg,Yref                !Store scale factors as real values
970                                     !
980 OUTPUT 707;"WAVEFORM:YINCREMENT?;YORIGIN?;YREFERENCE?" !Request scale
990                                     !factors.
1000 ENTER 707;Yinc,Yorg,Yref          !Read scale factors
1010                                     !sent by scope
1020                                     !
1030                                     !
1040 !-----
1050 !   Convert raw waveform data values in Min and Max arrays to voltages
1060 !-----
1070                                     !
1080                                     !
1090 FOR I=1 TO Array_len
1100     Min(I)=((Min(I)-Yref)*Yinc)+Yorg    !Convert Min(I) to volts
1110     Max(I)=((Max(I)-Yref)*Yinc)+Yorg    !Convert Max(I) to volts
1120 NEXT I
1130                                     !
1140                                     !
1150 LOCAL 707                          !Return scope to local mode
1160 END

```

## Custom Marker Risetime Measurements

```

10  ! -----
20  ! This example program uses the voltage and time markers to make a
30  ! custom risetime measurement and a channel to channel time interval
40  ! measurement.
50  !
60  ! Scope setup: Periodic signal 50 Hz and 640 mV p-p input to
70  !             channels 1 and 2. An external trigger signal must
80  !             be supplied.
90  !
100 ! Variables used:
110 !   Risetime = Time between 15% & 85% levels of 1st rising edge on Ch1.
120 !   Delta_t  = Time delay between 50% levels of 1st rising edges on
130 !             Channels 1 and 2
140 ! -----
150
160
170 REAL Risetime,Delta_t
180
190 CLEAR 707                      !Initialize scope HP-IB interface
200
210 OUTPUT 707;"*RST"              !Reset scope to default setup
220
230 OUTPUT 707;"AUTOSCALE"        !Setup timebase, vertical scale, & trigger
240
250 OUTPUT 707;"DISPLAY:GRATICULE FRAME" !Change to frame graticule to make
260                                     !it easier to see V & T markers
270
280
290 ! -----
300 ! This block of code does a custom risetime measurement between the
310 ! 15% and 85% voltage levels of the 1st rising edge on channel 1.
320 ! -----
330
340
350 OUTPUT 707;"MEASURE:SOURCE CHANNEL1" !Do measurement on channel 1
360
370 OUTPUT 707;"DISPLAY:VMARKER ON"      !Always turn Vmarkers on before using
380
390 OUTPUT 707;"MEASURE:PBASE 15;PTOP 85;VMARKERSET" !Set Vmarkers to
400                                                     !15% & 85% levels

```

```

410
420 ! Move start time marker to intersect with Vmarker1 at the 15% level
430 ! of rising edge 1, move stop time marker to intersect with Vmarker2
440 ! at the 85% level of rising edge 1, and ask scope to return the time
450 ! between the time markers.
460
470 OUTPUT 707;"MEASURE:EDELTA? 1,1"
480
490 ! Read time between time markers (15% to 85% risetime)
500
510 ENTER 707;Risetime
520
530
540 !-----
550 ! The next block of code does a channel to channel time interval
560 ! measurement such as might be used to measure propagation delay
570 ! through a device.
580 !-----
590
600
610 OUTPUT 707;"MEASURE:SOURCE CHANNEL1,CHANNEL2" !Slave Vmarker1 and Start
620 !time marker to channel 1.
630 !Slave Vmarker2 and Stop
640 !time marker to channel 2.
650
660 OUTPUT 707;"MEASURE:VRELATIVE 50;VMARKERSET" !Set Vmarkers at 50% levels
670 !of waveforms on assigned
680 !channels
690
700 ! Move the start time marker to intersect with Vmarker1 at the 50% level
710 ! of the 1st rising edge on channel1. Move the stop time marker to
720 ! intersect with Vmarker2 at the 50% level of the 1st rising edge
730 ! on channel 2. Tell scope to return the time between the time markers.
740
750 OUTPUT 707;"MEASURE:EDELTA? 1,1"
760
770 ENTER 707;Delta_t !Read time delay between time markers. If Ch1
780 !were the signal at the input of a device and
790 !Ch2 were the signal at the output, Delta_t would
800 !be the propagation delay through the device.
810
820 LOCAL 707 !Return scope to local mode
830
840 END

```

# Index

---

## A

ACQUIRE SUBSYSTEM 8-1 ... 8-7, 6-5, 19-1  
    BANDwidth command/query 8-3  
    COMPLet command/query 8-4  
    COUNT command/query 8-5  
    POINTs query 8-6  
    TYPE command/query 8-7, 19-1  
active display 6-8, 11-14, 11-28, 13-6  
active waveform display 6-11  
active display memory 11-16  
ADD command, function 12-4, 12-1  
addressing, HP-IB 1-3, 2-1  
addressed to talk with nothing to say 3-3  
Addressed to talk with no listeners on the bus 3-3  
ALL? query, measure 15-7 ... 15-8  
alpha arguments 7-9  
    < arbitrary ASCII response data > 3-25  
    < arbitrary block program data > 3-18  
ASCII format 19-5, 19-3  
ATN (Attention) 2-1  
attenuation factor 18-6  
Attribute Byte 11-7  
ATTRibute command/query, display 11-6 ... 11-8  
Auto Level Set key 15-31  
Automatic Top-Base A-1  
AUToscale command, root level commands 6-4, 6-12  
Average (mode) 8-5, 8-7, 19-1, 19-2  
Averaging Mode 8-1  
Axes 11-18

## B

Bandwidth 8-1  
BANDwidth command/query, acquire 8-3  
baseline (sweep) 17-5  
BLANK command, root level commands 6-4, 10-1  
BLINK command/query, display 11-9  
BRIGHtness command/query, display 11-10, 11-11  
Buffer Deadlock 3-4  
Bus Commands 2-3

## C

CALibrate command, network 16-4  
CALIBRATE SUBSYSTEM 9-1 ... 9-4  
    DATA command query 9-2 ... 9-3  
    GAIN command 9-4, 9-1  
calibration waveform 16-8, 16-14  
Cal String 9-2  
CHANNEL < N > SUBSYSTEM 10-1 ... 10-5  
    OFFSet command/query 10-3  
    PROBe command/query 10-4  
    RANGe command/query 10-5  
    < character program data > 3-14  
    < character response data > 3-23  
Clear Display 6-7  
\*CLS (Clear Status) command, common commands 5-3, 3-29  
CME (command error bit) 3-29  
Color bits 11-7  
COLOR command/query, display 11-11 ... 11-12, 11-10  
COLUMn command/query, display 11-13, 11-20, 11-29

Histogram (mode) 8-5, 8-7, 19-1, 19-2 19-16, 19-17

HISTOGRAM SUBSYSTEM 14-1 ... 14-13

LLIMit command/query 14-4

MEAN? query 14-5

PDELta? query 14-6

PLLimit command/query 14-7

PULimit command/query 14-8

REFerence command/query 14-9

SIGMA? query 14-10

SOURce command/query 14-11

ULIMit command/query 14-12

WINDow command/query 14-13

horizontal axis 17-1

HP-IB status (status annunciators) 2-3

HPGL 6-9

HP Raster Graphics Standard 6-10

Hue 11-26

## I

IDN? (Identification Number) query, commands 5-8, 6-11

IEEE 488.1 2-1

IEEE 488.2 common commands 4-2

IEEE 488.2 Standard 3-1

Infinite (Persistence) 11-23

Infinity Representation 4-5

Input Buffer 3-1

input sensitivity 10-4

Interface Capabilities 2-1

Interface Clear (IFC) 2-3

Interface Select Code 1-3

Interrupted Condition 3-4

Introduction to Programming an Oscilloscope 1-1

INVerse command/query, display 11-19

INVErt command, function 12-4, 12-1

## K

KEY command/query, system subsystem 7-7 ... 7-8

Key Codes 7-7, 7-8

## L

LCL (Local) Event Register 6-7

LCL (local bit) 3-29

learn string 5-9, 7-10

LER? (Local Event Register) query, root level commands 6-7

LEVel command/query, trigger 18-4

LINE command, display 11-20, 11-6, 11-13 11-19, 11-24, 11-25, 11-32

LLIMit command/query, histogram 14-4

LLO (Local Lockout) 2-2

local bit 5-3

Local and Local Lockout, Remote 2-2

Longform/Shortform 1-7

LONG format 19-5, 19-3

LONGform command/query, system commands 7-9

LRN? (Learn) query, common commands 5-9, 7-10

Luminosity 11-26

## M

MASK command/query, display 11-21 ... 11-22

MAV (Message Available bit) 3-29, 5-14

MAX command, function 12-5, 12-1

MEAN? query, histogram 14-5

mean 14-5, 14-10

MEASURE SUBSYSTEM 15-1 ... 15-40

ALL? query 15-7 ... 15-8

CURSor? query 15-8



DUTYcycle? query 15-9  
 EDELta? query 15-10  
 ESTArt command/query 15-11  
 ESTOp command/query 15-12  
 FALLtime? query 15-13  
 FREQuency? query 15-14  
 NWIDth? query 15-15  
 OVERshoot? query 15-16  
 PBASe command/query 15-17  
 PERiod? query 15-18  
 PRECision command/query 15-19  
 PREShoot? query 15-20  
 PTOp command/query 15-21  
 PWIDth? query 15-22  
 RISetime? query 15-23  
 SOURce command/query 15-24 ... 15-25 15-1  
 TDELta? query 15-25, 15-1  
 TSTArt command/query 15-26  
 TSTOp command/query 15-27  
 TVOLt? query 15-28  
 VAMPLitude? query 15-29  
 VBASE? query 15-30  
 VDELta? query 15-30, 15-1  
 VFIFty command 15-31  
 VMARkerset command 15-31  
 VMAX? query 15-32  
 VMIN? query 15-32  
 VPP? query 15-33  
 VRELative command/query 15-34 ... 15-35  
 VRMS? query 15-36  
 VSTArt command/query 15-37  
 VSTOp command/query 15-38  
 VTIME? query 15-39 ... 15-40  
 VTOP? query 15-40  
 MENU command/query, root level commands 6-8  
 menu select keys 7-8  
 Message Communication and System Functions 3-1  
 message exchange 3-1  
 MERGe command, root level commands 6-8  
 MIN command, function 12-5, 12-1

MLA (My Listen Address) 2-2  
 Mnemonic Truncation 4-1  
 MODE command/query, timebase 17-5  
 MODE command/query, trigger 18-5  
 MSG (message bit) 3-29  
 MSS (master summary status bit) 3-29  
 MTA (My Talk Address) 2-2  
 Multiple Queries 1-13

## N

Network Preset 6-12  
 NETWORK SUBSYSTEM 16-1 ... 16-18  
   CALibrate command 16-4  
   DIElectric command/query 16-5  
   RFLection:CURSor command/query 16-6  
   REFLection:DISTance? query 16-7  
   REFLection:IMPedance? query 16-7  
   REFLection:NORMAlize command 16-8  
   REFLection:PARAmeters? query 16-8  
   REFLection:PRESet command 16-9  
   REFLection:RHO? query 16-9  
   REFLection:SOURce command/query 16-10  
   REFLection:STEP command/query 16-10  
   REFLection:TIME? query 16-11  
   RISetime command/query 16-11  
   TRANsmission:CURSor command/query 16-12  
   TRANsmission:GAIN? query 16-13  
   TRANsmission:NORMAlize command 16-14  
   TRANsmission:PARAmeters? query 16-15  
   TRANsmission:SOURce command/query 16-16  
   TRANsmission:STEP command/query 16-16  
   TRANsmission:TIME? query 16-17  
   VELocity command/query 16-18  
 New Line Character 3-19, 6-6  
   <NL> (newline) 3-19  
 Normal (Persistence) Mode 8-1  
 Normal (mode) 8-5, 8-7, 19-1, 19-2  
 normalization filter 16-11  
 normalized waveform 16-8, 16-10, 16-16  
 Notation Conventions and Definitions 4-6

<nr1 numeric response data> 3-23

<nr3 numeric response data> 3-24

Number of Averages 8-1

Numeric Variables 1-13

NWIDth? query, measure 15-15

## O

OFFSet command/query, channel 10-3

OFFSet command/query, function 12-6

ONLY command, function 12-7, 12-1

\*OPC (Operation Complete) command/query  
common commands 5-10, 3-30

OPC (operation complete bit) 3-29

Open (calibrate) 16-4

OTA (Other Talk Address) 2-2

OUTPUT C-12 ... C-21

output queue 3-1, 5-10

Overlapped Commands 4-6

OVERshoot? query, measure 15-16

OVERshoot A-6

## P

PAGE command/query, hardcopy 13-3

Parallel Poll 3-32

parametric measurements 15-1

Parser 3-2

PBAsE command/query, measure 15-17

PDELta? query, histogram 14-6

PEN command/query, hardcopy 13-4

period 15-1

PERiod? query, measure 15-18

PERSistence command/query, display 11-23

Persistence (Normal) Mode 8-1

pixel data 11-14

pixel memory 6-7, 11-14

PLLimit command/query, histogram 14-7

PLOT? query, root level commands 6-9, 13-1

POINts command/query, acquire 8-6

POINts query, waveform 19-11, 19-2 19-3, 19-5

PON (power on bit) 3-29

preamble (waveform) 19-1

PREamble command/query, waveform 19-12 ...  
19-13

PRECision command/query, measure 15-19

Preset Reflect Channel key 16-9

PREShoot? query, measure 15-20

PRINT C-22 ... C-30

PRINT? query, root level commands 6-10, 13-1

PRINter command/query, hardcopy 13-5

PRiority command/query, display 11-24

probe attenuation factor 10-4, 18-6

PROBe command/query, channel 10-4

PROBe command/query, trigger 18-6

< program data > 3-14

< program data separator > 3-18

Program Examples B-1 ... B20

< program header separator > 3-19

< program message > 3-9, 3-2

Program Message Terminator 3-19, 1-8

< program message unit > 3-9

< program message unit separator > 3-11

Program Results 1-17

Programming and Documentation Conventions  
4-1

Programming Syntax 1-2

propagation delay 16-15

Protocols, IEEE 488.2 3-1

PTOP command/query, measure 15-21

PULimit command/query, histogram 14-8

pulse width 15-1

PWIDth? query, measure 15-22

## Q

Quad 6-13

Query Command 1-6

Query Error 3-4

query message 3-2

< query message unit > 3-10

< query program header > , < command  
program header > 3-11

quoted string 7-3

QYE (query error bit) 3-29

## R

RANGe command/query, channel 10-5  
RANGe command/query, function 12-8  
RANGe command/query, timebase 17-6  
RANGe? query, waveform 19-14  
raw data 19-4  
\*RCL (Recall) command, common commands 5-10  
REFerence command/query, histogram 14-9  
REFerence command/query, timebase 17-7, 17-3  
reflection channel 16-10  
reflection cursor 16-11  
REFlection:CURSOR command/query network 16-6  
REFlection:DIStance? query, network 16-7  
REFlection:IMPedance? query, network 16-7  
reflection normalization 16-11  
REFlection:NORMAlize command, network 16-8  
REFlection:PARAmeters? query, network 16-8  
REFlection:PRESet command, network 16-9  
reflection ratio 16-9  
REFlection:RHO? query, network 16-9  
REFlection:SOURce command/query network 16-10  
REFlection:STEP command/query, network 16-10  
REFlection:TIME? query, network 16-11  
Remote, Local and Local Lockout 2-2  
REN (Remote Enable) 2-2  
Repeat Loop 5-18  
Reset Conditions 5-11 ... 5-12  
< response data > 3-21  
< response data separator > 3-25  
< response header > 3-21  
< response header separator > 3-26  
< response message > 3-21, 3-2  
< response message terminator > 3-26

< response message unit > 3-21  
< response message unit separator > 3-26  
risetime 15-1, A-6  
RISetime command/query, network 16-11  
RISetime query, measure 15-23  
rms voltage 15-1  
Root Level Commands 6-1 ... 6-13, 4-2, 4-8  
    AUToscale command 6-4, 6-12  
    BLANK command 6-4  
    DIGitize command 6-5 ... 6-6, 19-1  
    EOI command/query 6-6  
    ERASe command 6-7  
    LER? query 6-7  
    MENU command/query 6-8  
    MERGe command 6-8  
    PLOT? query 6-9, 13-1  
    PRINT? query 6-10, 13-1  
    RUN command 6-11  
    SER command 6-11  
    STOP command 6-12  
    STORE command 6-12  
    TER? query 6-13  
    VIEW command 6-13, 5-14, 6-4, 10-1, 14-1  
ROW command/query, display 11-25 11-20, 11-29  
ROC (request control bit) 3-29  
RQS (request service) 3-29, 3-30  
\*RST command, common commands 5-11 ... 5-12  
RUN command, root level commands 6-11 5-17, 6-12  
Run From Loop 5-18

## S

Saturation 11-26  
\*SAV (Save) command, common commands 5-10, 5-13  
save/recall register 5-10  
SDC (Selective Device Clear) 2-3  
Selecting Multiple Subsystems 1-8  
SENSitivity command/query, trigger 18-7

- Sequential Commands 4-6
- SER command, root level commands 6-11
- serial number 5-8
- Serial Poll 3-31
- Service Request Enable Register 5-14
- SETColor command/query, display 11-26 ...  
11-27
- SETup command/query, system commands  
7-10, 5-9
- Short (calibrate) 16-4
- Shortform/Longform 1-7
- SIGMa? query, histogram 14-10
- Skew Calibration 9-1, 9-2
- SLOPe command/query, trigger 18-8
- software revision code 5-8
- SOURce command/query, display 11-28
- SOURce command, hardcopy 13-6
- SOURce command/query, histogram 14-11
- SOURce command/query, measure 15-24 ...  
15-25, 15-2
- SOURce command/query, trigger 18-9
- SOURce command/query, waveform 19-14  
19-3, 19-8
- SPEed command/query, hardcopy 13-7
- \*SRE (Service Request Enable)  
command/query, common commands 5-14 ...  
5-15
- standard deviation 14-10
- Standard Event Status Enable Register 5-4 5-5,  
5-6, 5-15
- Standard Event Status Register 5-7, 5-10
- statistical mean 14-5, 14-10
- Status 1-13
- Status Annunciations 2-3
- Status Byte 3-30
- Status Byte Register 5-3, 5-4, 5-14, 5-16
- Status Reporting 3-28
- \*STB? (Status Byte) query, common  
commands 5-3, 5-16
- step 18-4
- step generator 16-9, 16-10, 16-16
- STOP command, root level commands 6-12

- STORE command, root level commands 6-12
- STRing command, display 11-29, 11-6, 11-13  
11-19, 11-24, 11-25, 11-32
- < string program data > 3-17
- < string response data > 3-24
- String Variable 1-12
- Subsystem commands 4-2, 4-8
- SUBTract command, function 12-9, 12-1
- < suffix program data > 3-16
- Suffix Multiplier 3-16
- Suffix Unit 3-16
- Syntax Diagrams 3-5, 4-7
- SYSTEM SUBSYSTEM 7-1 ... 7-10
  - DSP command/query 7-3
  - ERRor? query 7-4 ... 7-5
  - HEADer command/query 7-6
  - KEY command/query 7-7 ... 7-8
  - LONGform command/query 7-9
  - SETup command/query 7-10, 5-9

## T

- TDELta? query, measure 15-25, 15-2
- TDR step 18-8
- TER? query, root level commands 6-13
- TEXT command, display 11-30
- Thru (calibrate) 16-4
- timebase delay 17-3
- Time/Div 6-4
- time domain reflectometry (TDR) 16-1
- time domain transmission (TDT) 16-1
- time histograms 19-2
- TIMEBASE SUBSYSTEM 17-1 ... 17-7
  - DELay command/query 17-3
  - GRATe command/query 17-4
  - MODE command/query 17-5
  - RANGe command/query 17-6
  - REFerence command/query 17-7
- time measurements 15-2
- TMARker command/query, display 11-31, 15-1
- transmission cursor 16-13, 16-17

TRANsmission:CURSor command/query  
     network 16-12  
 TRANsmission:GAIN? query, network 16-13  
 TRANsmission:NORMAlize command network  
     16-14  
 TRANsmission:PARAmeters? query, network  
     16-15  
 TRANsmission:SOURce command/query  
     network 16-16  
 TRANsmission:STEP command/query network  
     16-16  
 TRANsmission:TIME? query, network 16-17  
 Tree Traversal Rules 4-4  
 \*TRG command, common commands 5-17  
 TRG (trigger bit) 3-29, 3-30  
 TRG Event Register 6-13, 3-30  
 trigger bit 3-30, 5-3  
 TRIGGER SUBSYSTEM 18-1 ... 18-9  
     HFReject command/query 18-3  
     LEVel command/query 18-4  
     MODE command/query 18-5  
     PROBe command/query 18-6  
     SENSitivity command/query 18-7  
     SLOPe command/query 18-8  
     SOURce command/query 18-9  
 Truncation Rule 4-1  
 TSTArt command/query, measure 15-26  
 TSTOp command/query, measure 15-27  
 TST? query, common commands 15-18  
 TVOLt? query, measure 15-28  
 TYPE command/query, acquire 8-7  
 TYPE command/query, waveform 19-15 19-1,  
     19-3

## U

ULIMit command/query, histogram 14-12  
 UNDERline command/query, display 11-32  
 unsynchronized (sweep) 17-5  
 UNT (Untalk) 2-2

Unterminated Condition 3-4  
 Upper/Lower Case Equivalence 3-8  
 URQ (User Request) 3-29, 5-4, 5-6

## V

VALid? query, waveform 19-15  
 VAMPlitude? query, measure 15-29  
 variable levels functions 15-34  
 variable persistence mode 8-1  
 VBASE? query, measure 15-30  
 VDELta? query, measure 15-30, 15-2  
 VELOCITY command/query, network 16-18  
 velocity constant 16-5, 16-18  
 VERSus command, function 12-9, 12-1  
 vertical gain calibration 9-4  
 VFIFty command, measure 15-31  
 VIEW command, root level commands 6-13  
     5-14, 6-4, 10-1, 14-1  
 Vmarker1,2 15-2, 15-24  
 VMARker command/query, display 11-33, 15-1  
 VMARkerset command, measure 15-31  
 VMAX? query, measure 15-32  
 VMIN? query, measure 15-32  
 voltage histograms 19-2  
 VPP? query, measure 15-33  
 VRELative command/query, measure 15-34 ...  
     15-35  
 VRMS? query, measure 15-36  
 VSTArt command/query, measure 15-37  
 VSTOp command/query, measure 15-38  
 VTIME? query, measure 15-39 ... 15-40  
 VTOP? query, measure 15-40

## W

\*WAI command, common commands 5-19  
 waveform data 19-1  
 waveform preamble 19-4, 19-12  
 WAVEFORM SUBSYSTEM 19-1 ... 19-18  
     COUNT? query 19-8, 19-2  
     DATA command/query 19-8 ... 19-9 19-3, 19-4  
     FORMat command/query 19-10

POINTs? query 19-11, 19-2, 19-5  
 PREAmble command/query 19-12 ... 19-13  
 RANGE? query 19-14  
 SOURce command/query 19-14, 19-3, 19-8  
 TYPE command/query 19-15, 19-3  
 VALid? query 19-15  
 XINCrement? query 19-16  
 XORigin? query 19-16  
 XREFerence? query 19-17  
 YINCrement? query 19-17  
 YORigin? query 19-18  
 YREFerence? query 19-18  
 < white space > 3-8  
 WINDow command/query, histogram 14-13  
 WORD format 19-5

## X

X axis 17-1  
 XINCrement? query, waveform 19-16  
 XORigin? query, waveform 19-16  
 XREFerence? query, waveform 19-17  
 X vs Y 12-9

## Y

Y axis 10-1, 19-14  
 YINCrement? query, waveform 19-17  
 YORigin? query, waveform 19-18  
 YREFerence? query, waveform 19-18