

Errata

Title & Document Type: 54505B/06B/10B/12B Digitizing Oscilloscope Programming Reference

Manual Part Number: 54512-90905

Revision Date: May 1992

HP References in this Manual

This manual may contain references to HP or Hewlett-Packard. Please note that Hewlett-Packard's former test and measurement, semiconductor products and chemical analysis businesses are now part of Agilent Technologies. We have made no changes to this manual copy. The HP XXXX referred to in this document is now the Agilent XXXX. For example, model number HP8648A is now model number Agilent 8648A.

About this Manual

We've added this manual to the Agilent website in an effort to help you support your product. This manual provides the best information we could find. It may be incomplete or contain dated information, and the scan quality may not be ideal. If we find a better copy in the future, we will add it to the Agilent website.

Support for Your Product

Agilent no longer sells or supports this product. You will find any other available product information on the Agilent Test & Measurement website:

www.tm.agilent.com

Search for the model number of this product, and the resulting product page will guide you to any available information. Our service centers may be able to perform calibration if no repair parts are needed, but no other support from Agilent is available.

Programming Reference

**HP 54505B and HP 54506B
500 MSa/s Digitizing Oscilloscopes
and
HP 54510B and HP 54512B
1 GSa/s Digitizing Oscilloscopes**



© Copyright Hewlett-Packard Company 1992
Microsoft® is a U.S. registered trademark of Microsoft Corporation

Publication 54512-90905

Printed in the U.S.A. May 1992

Printing History

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

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

Edition 1

May 1992

54512-90905

List of Effective Pages

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

Pages

Effective Date

Product Warranty

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

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

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

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

- Limitation of Warranty** The foregoing warranty shall not apply to defects resulting from improper or inadequate maintenance by the Buyer, Buyer-supplied software or interfacing, unauthorized modification or misuse, operation outside of the environmental specifications for the product, or improper site preparation or maintenance.
- NO OTHER WARRANTY IS EXPRESSED OR IMPLIED. HEWLETT-PACKARD SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.
- Exclusive Remedies** THE REMEDIES PROVIDED HEREIN ARE BUYER'S SOLE AND EXCLUSIVE REMEDIES. HEWLETT-PACKARD SHALL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER BASED ON CONTRACT, TORT, OR ANY OTHER LEGAL THEORY.
- Assistance** Product maintenance agreements and other customer assistance agreements are available for Hewlett-Packard products.
- For assistance, contact your nearest Hewlett-Packard Sales and Service Office.
- Certification** Hewlett-Packard Company certifies that this product met its published specifications at the time of shipment from the factory. Hewlett-Packard further certifies that its calibration measurements are traceable to the United States National Institute of Standards and Technology, to the extent allowed by the Institute's calibration facility, and to the calibration facilities of other International Standards Organization members.
- Safety** This product has been designed and tested according to International Safety Requirements. To ensure safe operation and to keep the product safe, the information, cautions, and warnings in this manual must be heeded.

Contents

Introduction

How to Use This Manual

Chapter 1:

Introduction to Programming an Instrument

| | |
|--|------|
| Introduction | 1-1 |
| Talking to the Instrument | 1-2 |
| Program Message Syntax | 1-3 |
| Output Command | 1-3 |
| Device Address | 1-4 |
| Instructions | 1-4 |
| Instruction Header | 1-4 |
| White Space (Separator) | 1-5 |
| Program Data | 1-5 |
| Header Types | 1-5 |
| Combining Commands from the Same Subsystem | 1-7 |
| Duplicate Mnemonics | 1-7 |
| Query Command | 1-8 |
| Program Header Options | 1-9 |
| Program Data Syntax Rules | 1-10 |
| Character Program Data | 1-10 |
| Numeric Program Data | 1-11 |
| Embedded Strings | 1-11 |
| Program Message Terminator | 1-12 |
| Selecting Multiple Subsystems | 1-12 |

Chapter 2:

Programming an Instrument

| | |
|---|-----|
| Introduction | 2-1 |
| Initialization | 2-1 |
| Autoscale | 2-2 |
| Setting Up the Instrument | 2-2 |
| Example Program | 2-3 |
| Program Overview | 2-4 |
| Using the Digitize Command | 2-4 |
| Receiving Information from the Instrument | 2-6 |
| Response Header Options | 2-7 |

| | |
|---|------|
| Response Data Formats | 2-8 |
| String Variables | 2-9 |
| Numeric Variables | 2-10 |
| Definite-Length Block Response Data | 2-11 |
| Multiple Queries | 2-12 |
| Instrument Status | 2-12 |

Chapter 3:

Interface Functions

| | |
|--|-----|
| Introduction | 3-1 |
| Interface Capabilities | 3-1 |
| Command and Data Concepts | 3-1 |
| Addressing | 3-2 |
| Communicating Over the Bus (HP 9000 Series 200/300 Controller) | 3-2 |
| Remote, Local and Local Lockout | 3-3 |
| Bus Commands | 3-4 |
| Device Clear | 3-4 |
| Group Execute Trigger (GET) | 3-4 |
| Interface Clear (IFC) | 3-4 |
| Status Annunciators | 3-4 |

Chapter 4:

Programming and Documentation Conventions

| | |
|--|------|
| Introduction | 4-1 |
| Truncation Rules | 4-1 |
| Front Panel to Command Cross-Reference | 4-2 |
| The Command Tree | 4-10 |
| Command Types | 4-10 |
| Tree Traversal Rules | 4-14 |
| Examples | 4-14 |
| Infinity Representation | 4-16 |
| Sequential and Overlapped Commands | 4-16 |
| Response Generation | 4-16 |
| Notation Conventions and Definitions | 4-17 |
| Syntax Diagrams | 4-18 |
| Program Examples | 4-18 |
| Command Set Organization | 4-19 |

| | | |
|-------------------|---|------|
| Chapter 5: | Example Programs | |
| | Introduction | 5-1 |
| | Order of Commands | 5-2 |
| | HP Basic Initialize Program | 5-4 |
| | HP Basic Digitize Program | 5-6 |
| | Microsoft QuickBASIC Initialize Program | 5-9 |
| | Microsoft QuickBASIC Digitize Program | 5-11 |
| | Microsoft QuickBASIC Initialize Program | 5-16 |
| | Microsoft QuickC Initialize Program | 5-19 |
| | Microsoft QuickC Digitize Program | 5-21 |
| | Microsoft QuickC Initialize Program | 5-26 |

| | | |
|-------------------|------------------------|------|
| Chapter 6: | Common Commands | |
| | Introduction | 6-1 |
| | Common Commands | 6-3 |
| | *CLS | 6-4 |
| | *DMC | 6-5 |
| | *EMC | 6-6 |
| | *ESE | 6-7 |
| | *ESR | 6-9 |
| | *GMC | 6-11 |
| | *IDN | 6-12 |
| | *LMC | 6-13 |
| | *LRN | 6-14 |
| | *OPC | 6-15 |
| | *OPT | 6-16 |
| | *PMC | 6-17 |
| | *RCL | 6-18 |
| | *RST | 6-19 |
| | *SAV | 6-22 |
| | *SRE | 6-23 |
| | *STB | 6-25 |
| | *TRG | 6-27 |
| | *TST | 6-28 |
| | *WAI | 6-29 |

Chapter 7:**Root Level Commands**

| | |
|--------------------|------|
| Introduction | 7-1 |
| AUToscale | 7-5 |
| BEEPer | 7-6 |
| BLANk | 7-7 |
| BNC | 7-8 |
| DIGitize | 7-9 |
| ERASe | 7-11 |
| LER | 7-12 |
| LTER | 7-13 |
| MENU | 7-14 |
| MERGe | 7-15 |
| PLOT | 7-16 |
| PRINt | 7-17 |
| RUN | 7-18 |
| SERial | 7-19 |
| STATus | 7-20 |
| STOP | 7-21 |
| STORe | 7-22 |
| TER | 7-23 |
| VIEW | 7-24 |

Chapter 8:**System Subsystem**

| | |
|--------------------|------|
| Introduction | 8-1 |
| DATE | 8-3 |
| DSP | 8-4 |
| ERRor | 8-5 |
| HEADer | 8-8 |
| KEY | 8-9 |
| LONGform | 8-11 |
| PIMacro | 8-12 |
| SETup | 8-13 |
| TIME | 8-15 |

| | | |
|-------------------|---------------------------------|------|
| Chapter 9: | Acquire Subsystem | |
| | Introduction | 9-1 |
| | Acquire Type and Count | 9-3 |
| | (Normal) Persistence Mode | 9-3 |
| | Averaging Mode | 9-3 |
| | Envelope Mode | 9-4 |
| | Rawdata Mode | 9-4 |
| | COMPLETE | 9-5 |
| | COUNT | 9-7 |
| | POINTs | 9-8 |
| | TYPE | 9-10 |

| | | |
|--------------------|----------------------------|------|
| Chapter 10: | Calibrate Subsystem | |
| | Introduction | 10-1 |
| | DATA:ASCii | 10-2 |
| | SETup | 10-3 |
| | TNULI | 10-4 |

| | | |
|--------------------|--------------------------|-------|
| Chapter 11: | Channel Subsystem | |
| | Introduction | 11-1 |
| | COUPling | 11-4 |
| | DISPlay | 11-5 |
| | ECL | 11-6 |
| | HFReject | 11-7 |
| | LFReject | 11-8 |
| | OFFSet | 11-9 |
| | PROBe | 11-10 |
| | RANGe | 11-11 |
| | SETup | 11-12 |
| | TTL | 11-13 |

| | | |
|--------------------|----------------------------------|-------|
| Chapter 12: | Display Subsystem | |
| | Introduction | 12-1 |
| | COLumn | 12-5 |
| | CONNect | 12-6 |
| | DATA | 12-7 |
| | FORMat | 12-9 |
| | GRATicule | 12-10 |
| | INVerse | 12-11 |
| | LINE | 12-12 |
| | MASK | 12-13 |
| | PERsistence | 12-15 |
| | ROW | 12-17 |
| | SCReen | 12-18 |
| | SETup | 12-19 |
| | SOURce | 12-20 |
| | STRing | 12-21 |
| | TEXT | 12-22 |
| | MARKer TMARKer VMARker | 12-23 |

| | | |
|--------------------|---------------------------|-------|
| Chapter 13: | Function Subsystem | |
| | Introduction | 13-1 |
| | ADD | 13-5 |
| | DIFF | 13-6 |
| | DISPLay | 13-7 |
| | FFT | 13-8 |
| | FREQuency | 13-9 |
| | INTEgrate | 13-10 |
| | INVert | 13-11 |
| | LEVel | 13-12 |
| | MAGNify | 13-13 |
| | MULTiPLY | 13-14 |
| | OFFSet | 13-15 |
| | ONLY | 13-16 |
| | PEAK | 13-17 |
| | POINts | 13-18 |

| | |
|----------------|-------|
| RANGe | 13-21 |
| SETup | 13-22 |
| SPAN | 13-23 |
| SUBTract | 13-24 |
| VERSus | 13-25 |
| WINDow | 13-26 |

Chapter 14: Hardcopy Subsystem

| | |
|-----------------------|------|
| Introduction | 14-1 |
| LENGth | 14-4 |
| MODE | 14-5 |
| PAGE | 14-6 |
| PLOT:AREA | 14-7 |
| PLOT:INITialize | 14-8 |
| PLOT:PEN COLor | 14-9 |

Chapter 15: Measure Subsystem

| | |
|---------------------------------|-------|
| Introduction | 15-1 |
| Measurement Setup | 15-10 |
| User-Defined Measurements | 15-10 |
| Measurement Error | 15-10 |
| Making Measurements | 15-11 |
| ALL | 15-13 |
| COMPare | 15-14 |
| CURSor | 15-16 |
| DEFine | 15-17 |
| DELay | 15-19 |
| DESTination | 15-20 |
| DUTYcycle | 15-22 |
| ESTArt | 15-23 |
| ESTOp | 15-25 |
| FALLtime | 15-27 |
| FREQuency | 15-28 |
| LIMittest | 15-29 |
| LOWer | 15-30 |
| MODE | 15-31 |
| NWIDth | 15-32 |
| OVERshoot | 15-33 |
| PERiod | 15-34 |

| | |
|----------------------------|-------|
| POSTfailure | 15-35 |
| PREShoot | 15-36 |
| PWIDth | 15-37 |
| RESults | 15-38 |
| RISetime | 15-39 |
| SCRatch | 15-40 |
| SOURce | 15-41 |
| STATistics | 15-42 |
| TDELta | 15-43 |
| TMAX | 15-44 |
| TMIN | 15-45 |
| TSTArt | 15-46 |
| TSTOp | 15-47 |
| TVOLt | 15-48 |
| UNITs | 15-50 |
| UPPer | 15-51 |
| VACRms | 15-52 |
| VAMPLitude | 15-53 |
| VAVerage | 15-54 |
| VBASe | 15-55 |
| VDCRms | 15-56 |
| VDELta | 15-57 |
| VFIFty | 15-58 |
| VMAX | 15-59 |
| VMIN | 15-60 |
| VPP | 15-61 |
| VRELative | 15-62 |
| VRMS | 15-64 |
| VSTArt | 15-65 |
| VSTOp | 15-66 |
| VTIME | 15-67 |
| VTOP | 15-68 |
| WCOMpare | 15-69 |
| WCOMpare:ALLowance | 15-70 |
| WCOMpare:COMpare | 15-71 |
| WCOMpare:DESTination | 15-72 |
| WCOMpare:POSTfailure | 15-74 |
| WCOMpare:TEST | 15-75 |

| | | |
|--------------------|---------------------------|------|
| Chapter 16: | Timebase Subsystem | |
| | Introduction | 16-1 |
| | DELAy | 16-3 |
| | MODE | 16-4 |
| | RANGE | 16-5 |
| | REFerence | 16-6 |
| | SAMPlE | 16-7 |
| | SAMPlE | 16-8 |

| | | |
|--------------------|--------------------------------|-------|
| Chapter 17: | Trigger Subsystem | |
| | Introduction | 17-1 |
| | Trigger Mode | 17-6 |
| | The EDGE Trigger Mode | 17-8 |
| | The Pattern Trigger Mode | 17-9 |
| | The State Trigger Mode | 17-10 |
| | The Delay Trigger Mode | 17-11 |
| | The TV Trigger Mode | 17-12 |
| | The Glitch Trigger Mode | 17-14 |
| | CENTERed | 17-15 |
| | CONDition | 17-16 |
| | COUPLing | 17-18 |
| | DELAy | 17-19 |
| | DELAy:SLOPe | 17-20 |
| | DELAy:SOURce | 17-21 |
| | FIELD | 17-22 |
| | GLITCh | 17-23 |
| | GLITCh:CENTERed | 17-24 |
| | GLITCh:HOLDoff | 17-25 |
| | GLITCh:LEVel | 17-26 |
| | GLITCh:SOURce | 17-27 |
| | GLITCh:WIDTh | 17-28 |
| | HOLDoff | 17-29 |
| | LEVel | 17-30 |
| | LINE | 17-31 |
| | LOGic | 17-32 |
| | MODE | 17-33 |
| | OCCurrence | 17-34 |
| | OCCurrence:SLOPe | 17-35 |

| | |
|-------------------------|-------|
| OCCurrence:SOURce | 17-36 |
| PATH | 17-37 |
| POLarity | 17-38 |
| PROBe | 17-39 |
| QUALify | 17-40 |
| SENSitivity | 17-41 |
| SETup | 17-42 |
| SLOPe | 17-43 |
| SOURce | 17-44 |
| STANdard | 17-45 |

Chapter 18:

Waveform Subsystem

| | |
|---|-------|
| Introduction | 18-1 |
| Waveform Data and Preamble | 18-3 |
| Data Acquisition Types | 18-4 |
| Realtime Mode | 18-4 |
| Repetitive Modes | 18-5 |
| Rawdata | 18-6 |
| Data Conversion | 18-8 |
| Conversion from Data Value to Voltage | 18-8 |
| Conversion from Data Value to Time | 18-8 |
| Data Format for HP-IB Transfer | 18-9 |
| WORD Format | 18-9 |
| BYTE Format | 18-10 |
| COMPRESSED Format | 18-10 |
| ASCII Format | 18-10 |
| DATA | 18-11 |
| FORMat | 18-13 |
| POINts | 18-14 |
| PREamble | 18-15 |
| SOURce | 18-17 |
| TYPE | 18-18 |
| XINCrement | 18-19 |
| XORigin | 18-20 |
| XREFerence | 18-21 |
| YINCrement | 18-22 |
| YORigin | 18-23 |
| YREFerence | 18-24 |

| | | |
|--------------------|----------------------------------|-------|
| Chapter 19: | Waveform Memory Subsystem | |
| | Introduction | 19-1 |
| | DISPlay | 19-3 |
| | GET | 19-4 |
| | PROTect | 19-5 |
| | SETup | 19-6 |
| | XOFFset | 19-7 |
| | XRANge | 19-8 |
| | YOFFset | 19-9 |
| | YRANge | 19-10 |

| | | |
|--------------------|-------------------------------|------|
| Chapter 20: | Pixel Memory Subsystem | |
| | Introduction | 20-1 |
| | CLEar | 20-2 |
| | DISPlay | 20-3 |
| | MERGe | 20-4 |
| | SETup | 20-5 |

| | | |
|--------------------|----------------------------------|------|
| Chapter 21: | Multiple Memory Subsystem | |
| | Introduction | 21-1 |
| | DISPlay | 21-2 |
| | FNUMber | 21-3 |
| | SOURce | 21-4 |
| | STORe | 21-5 |

| | | |
|--------------------|-----------------------------|-------|
| Chapter 22: | Sequential Subsystem | |
| | Introduction | 22-1 |
| | DISPlay | 22-3 |
| | EXCLude | 22-4 |
| | INCLude | 22-6 |
| | NPOints | 22-8 |
| | NSEGments | 22-9 |
| | SETup | 22-11 |
| | SNUMber | 22-12 |
| | SOURce | 22-13 |

| | | |
|--------------------|-------------------------|-------|
| Chapter 23: | Marker Subsystem | |
| | Introduction | 23-1 |
| | DISPlay | 23-3 |
| | SETup | 23-4 |
| | XDELta | 23-5 |
| | X1Position | 23-6 |
| | X2Position | 23-7 |
| | X1Y1source | 23-8 |
| | X2Y2source | 23-9 |
| | YDELta | 23-10 |
| | Y1Position | 23-11 |
| | Y2Position | 23-12 |

| | | |
|--------------------|---------------------------------|------|
| Chapter 24: | Status Reporting | |
| | Introduction | 24-1 |
| | Bit Definitions | 24-3 |
| | Operation Complete (*OPC) | 24-4 |
| | Trigger Bit (TRG) | 24-4 |
| | Serial Poll | 24-5 |
| | Using Serial Poll | 24-5 |

| | |
|--------------------|-----------------------|
| Chapter 25: | Error Messages |
|--------------------|-----------------------|

| | | |
|--------------------|-----------------------------|------|
| Chapter 26: | Algorithms | |
| | Introduction | 26-1 |
| | Measurement Setup | 26-1 |
| | Making Measurements | 26-1 |
| | Automatic Top-Base | 26-2 |
| | Edge Definition | 26-2 |
| | Algorithm Definitions | 26-3 |
| | delay | 26-3 |
| | + width | 26-4 |
| | - width | 26-4 |
| | Period | 26-5 |
| | Frequency | 26-5 |
| | Duty Cycle | 26-5 |

| | |
|---------------------|------|
| Rise time | 26-5 |
| Fall time | 26-5 |
| Vmax | 26-5 |
| Vmin | 26-5 |
| Vp-p | 26-5 |
| Vtop | 26-5 |
| Vbase | 26-5 |
| Vamp | 26-5 |
| Vavg | 26-6 |
| Vrms (ac) | 26-6 |
| Vrms (dc) | 26-6 |
| Integrate | 26-6 |
| Differentiate | 26-6 |

Chapter 27:

Message Communication and System Functions

| | |
|-------------------------------|-------|
| Introduction | 27-1 |
| Protocols | 27-1 |
| Functional Elements | 27-1 |
| Protocol Overview | 27-2 |
| Protocol Operation | 27-3 |
| Protocol Exceptions | 27-3 |
| Syntax Diagrams | 27-5 |
| Syntax Overview | 27-5 |
| Device Listening Syntax | 27-8 |
| Device Talking Syntax | 27-21 |
| Common Commands | 27-27 |

Quick Reference Guide

Index

Introduction

The HP 54505B/54506B 500 MSa/s and HP 54510B/54512B 1 GSa/s Digitizing Oscilloscopes are general purpose repetitive and realtime oscilloscopes. Full HP-IB programmability is incorporated into both oscilloscopes and may be used in a broad range of HP-IB applications, from high-speed ATE to device characterization in research and development environments. This manual explains how to program the oscilloscope over HP-IB and lists the commands and queries associated with this instrument.

How to Use This Manual

This manual is divided into 27 chapters. The first two chapters introduce you to the programming syntax and some basic programming concepts to help get you started programming.

Chapter 3 describes the interface functions and some general concepts of HP-IB.

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

Chapter 5 contains **example programs** using the command set from the oscilloscope.

Chapters 6 through 23 list the commands and queries associated with the oscilloscope, including their corresponding arguments and returned formats.

Chapter 24 describes the **status reporting** features of the oscilloscope that are available over the HP-IB.

Chapter 25 lists the **error messages** that are returned by the parser on the oscilloscope.

Chapter 26 provides details on how automatic measurements are calculated and offers some tips on how to improve results.

Chapter 27 describes the operation of instruments that operate in compliance with the **IEEE 488.2** standard.

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

Introduction to Programming an Instrument

Introduction

Chapters 1 and 2 introduce you to the basics of remote programming. The programming instructions explained in this manual conform to the IEEE 488.2 Standard Digital Interface for Programmable Instrumentation. These programming instructions provide a means of remotely controlling the oscilloscopes. There are four basic operations that can be done with a controller and an oscilloscope via HP-IB. You can:

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

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

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 6 through 23.

Chapter 2 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. Refer to chapter 15, "Measure Subsystem" for information on sending measurement data to the instrument.



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

Talking to the Instrument

In general, computers acting as controllers communicate with the instrument by sending and receiving messages over a remote interface. Instructions for programming the oscilloscope normally appear as ASCII character strings embedded inside the output statements of a "host" language available on your controller. The input statements of the host language are used to read in responses from the oscilloscope.

For example, HP 9000 Series 200/300 BASIC uses the OUTPUT statement for sending commands and queries to the oscilloscope. After a query is sent, the response is usually read in using the ENTER statement.

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

The following BASIC statement sends a command which turns the command headers on:

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

The < device address > represents the address of the device being programmed. Each of the other parts of the above statement are explained in the following pages.

Program Message Syntax

To program the instrument remotely, you must have an understanding of the command format and structure expected by the instrument. The IEEE 488.2 syntax rules govern how individual elements such as headers, separators, program data, and terminators may be grouped together to form complete instructions. Syntax definitions are also given to show how query responses are formatted. Figure 1-1 shows the main syntactical parts of a typical program statement.

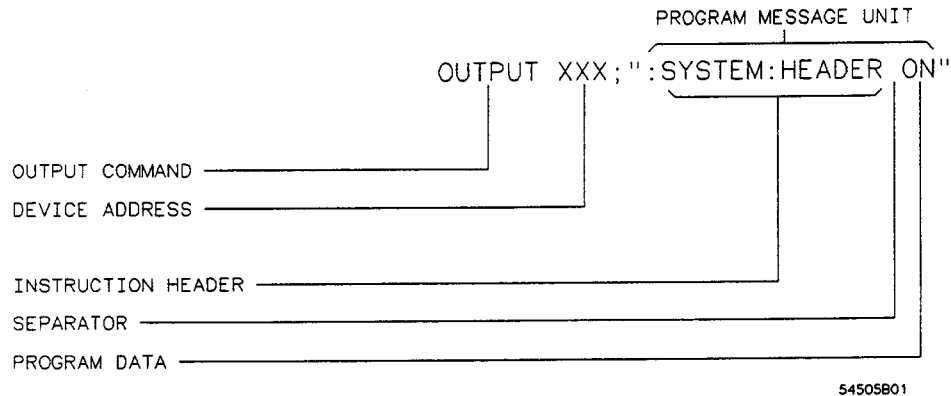


Figure 1-1. Program Message Syntax

Output Command

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

Device Address The location where the device address must be specified is also dependent on which host language you are using. In some languages, this may be specified outside the output command. In BASIC, this is always specified after the keyword OUTPUT. The examples in this manual assume the oscilloscope is at device address 707. When writing programs, the address you use varies according to how you have configured the bus for your application.

Instructions Instructions (both commands and queries) normally appear as a string embedded in a statement of your host language, such as BASIC, Pascal, or C. The only time a parameter is not meant to be expressed as a string is when the instruction's syntax definition specifies <block data >. There are only a few instructions which use block data.

Instructions are composed of two main parts:

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

Instruction Header The instruction header is one or more mnemonics separated by colons (:) that represent the operation to be performed by the instrument. The command tree in figure 4-1 illustrates how all the mnemonics can be joined together to form a complete header (see chapter 4, "Programming and Documentation Conventions").

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

White Space (Separator)

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

Program Data

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

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

Header Types

There are three types of headers:

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

Simple Command Header

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

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

When program data must be included with the simple command header (for example, :DIGITIZE CHAN1), white space is added to separate the data from the header. The syntax is:

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

Compound Command Header

Compound command headers are a combination of two or more program mnemonics. The first mnemonic selects the subsystem, and the last mnemonic selects the function within that subsystem. Additional mnemonics may appear between the subsystem mnemonic and the function mnemonic when there are additional levels within the subsystem that must be traversed. The mnemonics within the compound message are separated by colons. For example:

To execute a single function within a subsystem:

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

(For example :SYSTEM:LONGFORM ON)

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

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

(For example :TRIGGER:DELAY:SOURCE CHAN1)

Common Command Header

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

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

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

Combining Commands from the Same Subsystem

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

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

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

Duplicate Mnemonics

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

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

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

```
:TIMEBASE:RANGE 1
```

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

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

Query Command

Command headers immediately followed by a question mark (?) are queries. After receiving a query, the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the answer is transmitted across the bus to the designated listener (typically a controller). For example, the query :TIMEBASE:RANGE? places the current time base setting in the output queue. In BASIC, the controller input statement:

```
ENTER < device address > ;Range
```

passes the value across the bus to the controller and places it in the variable Range.

Query commands are used to find out how the instrument is currently configured. They are also used to get results of measurements made by the instrument, with the query actually activating the measurement. For example, the command :MEASURE:RISETIME? instructs the instrument to measure the rise time of your waveform and place the result in the output queue.

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

Note

Sending another command or query before reading the result of a query causes the output buffer to be cleared and the current response to be lost. This also generates an error in the error queue.

Program Header Options

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

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

```
:SYSTEM:HEADER ON - long form
```

```
:SYST:HEAD ON - short form
```

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



The rules for the short form syntax are shown in chapter 4, "Programming and Documentation Conventions."

Program Data Syntax Rules

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

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

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

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

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

There are two main types of program data which are used in HP 54505B, HP 54506B, HP 54510B, and HP 54512B commands: character and numeric program data.

Character Program Data

Character program data is used to convey parameter information as alpha or alphanumeric strings. For example, the TIMEBASE:MODE command can be set to auto, trigger, or single. The character program data in this case may be AUTO, TRIGGER, or SINGLE. :TIMEBASE:MODE SINGLE sets the time base mode to single.

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

Numeric Program Data

Some command headers require program data to be expressed numerically. For example, `:TIMEBASE:RANGE` requires the desired full scale range to be expressed numerically.

For numeric program data, you have the option of using exponential notation or using suffix multipliers to indicate the numeric value. The following numbers are all equal:

$$28 = 0.28E2 = 280e-1 = 28000m = 0.028K = 28e-3K.$$

When a syntax definition specifies that a number is an integer, that means that the number should be whole. Any fractional part would be ignored, truncating the number. Numeric data parameters which accept fractional values are called real numbers. For more information see chapter 27, "Message Communication and System Functions."

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

Embedded Strings

Embedded strings contain groups of alphanumeric characters which are treated as a unit of data by the oscilloscope. For example, the line of text written to the advisory line of the instrument with the `:SYSTEM:DSP` command. Embedded strings may be delimited with either single (') or double (") quotes. These strings are case-sensitive and spaces act as legal characters just like any other character.

Program Message Terminator

The program instructions within a data message are executed after the program message terminator is received. The terminator may be either an NL (New Line) character, an EOI (End-Of-Identify) asserted, or a combination of the two. All three ways are equivalent with the exact encodings for the program terminators listed in chapter 27, "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).



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

Selecting Multiple Subsystems

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

```
<program mnemonic><data>;<program mnemonic><data><terminator>  
:CHANNEL1:RANGE 0.4;;:TIMEBASE:RANGE 1
```



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

Programming an Instrument

Introduction

This chapter deals mainly with how to set up the instrument, how to retrieve setup information and measurement results, how to digitize a waveform, and how to pass data to the controller. Refer to chapter 15, "Measure Subsystem" for information on sending measurement data to the instrument.



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

Initialization

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

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

When you are using HP-IB, CLEAR also resets the oscilloscope's parser. The parser is the program which reads in the instructions which you send it.

After clearing the interface, initialize the instrument to a preset state. For example:

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



The actual commands and syntax for initializing the instrument are discussed in chapter 6, "Common Commands."

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

Autoscale

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

The syntax for Autoscale is:

```
:AUTOSCALE<terminator>
```

Setting Up the Instrument

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

```
:CHANNEL1:RANGE 0.64;OFFSET 0.25<terminator>
```

```
:TIMEBASE:RANGE 1E-6;DELAY 20E-9;MODE TRIGGERED <terminator>
```

```
:TRIGGER:LEVEL 0.25;SLOPE POSITIVE<terminator>
```

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

Example Program

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

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

Program Overview

Line 10 initializes the instrument interface to a known state.

Line 20 initializes the instrument to a preset state.

Line 30 selects the Probe mode for the BNC output on the rear panel.

Lines 40 through 70 set the time base mode to triggered with the horizontal time at $50 \mu\text{s}/\text{div}$ with 0 s of delay referenced at the center of the graticule.

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

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

Line 150 captures the measurement data.

Line 160 runs the measurement and acquires the data.

Using the Digitize Command

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



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

When the DIGITIZE command is sent to an instrument, the specified channel signal is digitized with the current ACQUIRE parameters. To transfer waveform data across the bus, you must setup the WAVEFORM:SOURCE and WAVEFORM:FORMAT parameters for the waveform data prior to sending the :WAVEFORM:DATA? query.

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

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

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

After receiving the :WAVEFORM:DATA? query, the instrument will start passing the waveform information when addressed to talk.

Digitized waveforms are passed from the instrument to the controller by sending a numerical representation of each digitized point. The format of the numerical representation is controlled with the :WAVEFORM:FORMAT command and may be selected as ASCII, WORD, BYTE, or COMPRESSED.

Note 

A digitize operation may be aborted by pressing the "LOCAL" key, then any other key on the front panel, or by sending a Device Clear over the bus. Digitize operations with :ACQUIRE:TYPE RAWDATA can only be aborted with a Device Clear.

Receiving Information from the Instrument

After receiving a query (command header followed by a question mark), the instrument interrogates the requested function and places the answer in its output queue. The answer remains in the output queue until it is read or another command is issued. When read, the answer is transmitted across the bus to the designated listener (typically a controller). The input statement for receiving a response message from an instrument's output queue typically has two parameters; the device address, and a format specification for handling the response message. For example, to read the result of the query command `:SYSTEM:LONGFORM?` you would execute the BASIC statement:

```
ENTER <device address> ;Setting$
```

where `< device address >` represents the address of your device. This would enter the current setting for the `LONGFORM` command in the string variable `Setting$`.

All results for queries sent in a program message must be read before another program message is sent. For example, when you send the query `:MEASURE:RISETIME?`, you must follow that query with an input statement. In BASIC, this is usually done with an `ENTER` statement.

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

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

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

Response Header Options

The format of the returned ASCII string depends on the current settings of the SYSTEM HEADER and LONGFORM commands. The general format is:

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

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

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

The following examples show some possible responses for a **:MEASURE:FREQUENCY?** frequency measurement query:

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



A command or query may be sent in either long form or short form, or in any combination of long form and short form. The HEADER and LONGFORM commands only control the format of the returned data and have no effect on the way commands are sent.

Refer to chapter 8, "System Subsystem" for information on turning the HEADER and LONGFORM commands on and off.

Response Data Formats

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

The following examples are possible responses to the :TRIGGER:SLOPE? query:

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



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

String Variables

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

```
ENTER <device address>;Result$
```

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



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

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



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

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

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

After running this program, the controller will typically display:

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

Numeric Variables

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



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

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

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

After running this program, the controller will typically display:

.8

Definite-Length Block Response Data

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

For example, for transmitting 80 bytes of data, the syntax would be:

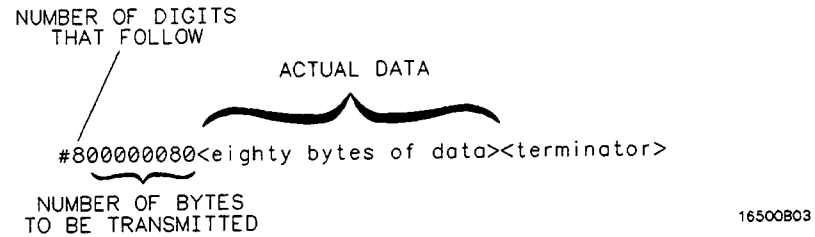


Figure 2-1. Definite-Length Block Response Data

The "8" states the number of digits that follow, and "00000080" states the number of bytes to be transmitted.

Multiple Queries

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

```
ENTER 707;Results$
```

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

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

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

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



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

Instrument Status

Status registers track the current status of the instrument. By checking the instrument status, you can find out whether an operation has been completed, whether the instrument is receiving triggers, and more. Chapter 24, "Status Reporting" explains how to check the status of the instrument.

Interface Functions

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 54505B, HP 54506B, HP 54510B, and HP 54512B, as defined by IEEE 488.1, are SH1, AH1, T5, L4, SR1, RL1, DC1, DT1, C0, and E2.

Command and Data Concepts

The HP-IB has two modes of operation:

- command mode.
- data mode.

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

The bus is in the data mode when the ATN line is false. The data mode is used to convey device-dependent messages across the bus. The device-dependent messages include all of the instrument commands and responses found in chapters 6 through 23 of this manual.

Addressing

By using the front-panel controls, the instrument can be placed in either the talk-only mode or addressed (talk/listen) mode (see your front-panel reference). Talk-only mode should be used when you want the instrument to talk directly to a printer without the aid of a controller. Addressed mode is used when the instrument operates 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 talk and which listen.
- An instrument may be talk addressed, listen addressed, or unaddressed by the controller.

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

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

Communicating Over the Bus (HP 9000 Series 200/300 Controller)

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

Interface Select Code (Selects Interface)

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

Instrument Address (Selects Instrument)

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

$$\text{DEVICE ADDRESS} = (\text{Interface Select Code} * 100) + (\text{Instrument Address})$$

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

For the HP 54505B, HP 54506B, HP 54510B, and HP 54512B, the instrument address is typically set to "7" at the factory. This address can be changed in the HP-IB menu of the Utility menu.



The examples in this manual assume the oscilloscope is at device address 707.

Remote, Local and Local Lockout

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



Cycling the power also restores local control, but this also resets certain HP-IB states.

The instrument is placed in the remote mode by setting the REN bus control line true, and then addressing the instrument to listen. The instrument is placed in the local lockout mode by sending the local lockout command (LLO). The instrument can be returned to the local mode by either setting the REN line false, or sending the go-to-local command (GTL) to the instrument.

Bus Commands

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

Device Clear

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

Group Execute Trigger (GET)

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

Interface Clear (IFC)

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

Status Annunciators

The oscilloscope displays the HP-IB status on the screen. The message indicates whether the instrument is in the remote mode, whether talk or listen is addressed, and whether the instrument has requested service. When the instrument is in the local mode only the SRQ annunciator is displayed.

Programming and Documentation Conventions

Introduction

This chapter covers conventions which are used in programming the instrument, as well as conventions used in the remainder of this manual. This chapter contains a detailed description of the command tree and command tree traversal. For more information on command syntax refer to chapter 27, "Message Communication and System Functions."

Truncation Rules

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

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

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

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

Some examples of how the truncation rule is applied to various commands are shown in table 4-1.

Table 4-1. Mnemonic Truncation

| Long Form | Short Form |
|-----------|------------|
| RANGE | RANG |
| PATTERN | PATT |
| TIME | TIME |
| DELAY | DEL |

Front Panel to Command Cross-Reference

Table 4-2 lists the front-panel functions for the oscilloscope in alphabetical order with their corresponding programming commands.

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|-------------------------------|---------------------------|-----------------------|
| 1 2 3 4 EXT | Channel menu | :BLANk |
| 1 2 3 4 EXT | Channel menu | :CHANnel[:DISPlay] |
| 1 2 3 4 EXT | Channel menu | :VIEW |
| 1 2 3 4 EXT | Delay Trigger menu (edge) | :TRIGger:DELay:SOURce |
| 1 2 3 4 EXT LINE | Edge Trigger menu | :TRIGger:SOURce |
| 1 2 3 4 EXT | State Trigger menu | :TRIGger:SOURce |
| 1 M Ω 50 Ω DC | Channel menu | :TRIGger:COUPling |
| 525 625 lines | TV Trigger menu | :TRIGger:LINE |
| $\Delta t \Delta V$ | Marker menu | :MARKer:XDELta |
| $\Delta t \Delta V$ | Marker menu | :MEASure:TDELta |
| $\Delta t \Delta V$ | Marker menu | :MARKer:YDELta |
| $\Delta t \Delta V$ | Marker menu | :MEASure:VDELta |
| $\Delta t \Delta V$ | front-panel key | :MENU:DELta |
| Δt markers | Marker menu | :DISPlay:TMARker |
| Δt markers | Marker menu | :MARKer[:DISPlay] |
| Δt markers | Marker menu | :MEASure:ESTArt |
| Δt markers | Marker menu | :MEASure:ESTOp |
| Δt markers | Marker menu | :MARKer:X1Position |
| Δt markers | Marker menu | :MEASure:TSTArt |
| Δt markers | Marker menu | :MARKer:X2Position |
| Δt markers | Marker menu | :MEASure:TSTOp |
| ΔV markers | Marker menu | :DISPlay:VMARker |
| ΔV markers | Marker menu | :MARKer[:DISPlay] |
| ΔV markers | Marker menu | :MARKer:Y1Position |
| ΔV markers | Marker menu | :MEASure:VSTArt |
| ΔV markers | Marker menu | :MARKer:Y2Position |
| ΔV markers | Marker menu | :MEASure:VSTOp |
| - subtract | Waveform Math menu | :FUNction:SUBTract |
| - WIDTH | front-panel key | :MEASure:NWIDth |
| # of avg | Display menu (avg) | :ACQuire:COUNt |
| # of points | Timebase menu | :SEQuential:NPOints |
| # of points | Waveform Math menu (FFT) | :FUNction:POINts |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|---------------------------|--------------------------------|---------------------------|
| # of screens | Display menu | :DISPlay:FORMat |
| # of segments | Display menu (avg single-shot) | :ACQuire:TYPE |
| # of segments | Timebase menu | :SEQuential:NSEGments |
| % volts | Define Meas menu (meas def) | :MEASure:UNITs |
| + add | Waveform Math menu | :FUNCTion:ADD |
| + WIDTH | front-panel key | :MEASure:PWIDth |
| ↑ X X X | Delay Trigger menu (state) | :TRIGger:LOGic |
| ↑ X X X | State Trigger menu | :TRIGger:LOGic |
| AC BNC | Utility menu | :BNC |
| ac dc | Channel menu | :CHANnel:COUPling |
| add + | Waveform Math menu | :FUNCTion:ADD |
| address | Utility menu (HP-IB menu) | (front panel only) |
| addressed talk only | Utility menu (HP-IB menu) | program language specific |
| add to memory | Waveform Save menu (pixel) | :MERGe |
| add to memory | Waveform Save menu (pixel) | :PMEMory:MERGe |
| adjust | Edge Trigger menu | :TRIGger:LEVel |
| adjust | Glitch Trigger menu | :TRIGger:GLITch:LEVel |
| after fail | Define Meas menu (compare) | :MEAS:WCOM:POSTfailure |
| after fail | Define Meas menu (meas limit) | :MEASure:POSTfailure |
| allowance | Define Meas menu (compare) | :MEAS:WCOM:ALLowance |
| attenuation | Utility menu (probe cal menu) | (front panel only) |
| AUTOSCALE | front-panel key | :AUToscale |
| auto trig'd | Trigger menu | :TIMEbase:MODE |
| avg env norm | Display menu | :ACQuire:TYPE |
| axes grid off frame | Display menu | :DISPlay:GRATicule |
| BW lim | Channel menu | :CHANnel:HFRject |
| center | Edge Trigger menu | :TRIGger:CENTerEd |
| center | Glitch Trigger menu | :TRIGger:GLITch:CENTerEd |
| center freq | Waveform Math menu (FFT) | :FUNCTion:FREQ |
| CHAN | front-panel key | :MENU:CHANnel |
| channel | Utility menu (probe cal menu) | :CALibrate:TNUlI |
| CLEAR DISPLAY | front-panel key | :ERASe pmem0 |
| clear memory | Waveform Save menu (pixel) | :ERASe |
| clicker | Utility menu | :BEEPer |
| clock | Delay Trigger menu (state) | :TRIGger:DELay:SOURce |
| clock | State Trigger menu | :TRIGger:SOURce |
| CLR MEAS | front-panel key | :MEASure:SCRatch |
| color | Utility menu (HP-IB menu) | :HARDcopy:PLOT:COLor |
| compare | Define Meas menu (compare) | :MEAS:WCOMpare:COMParE |
| connect dots | Display menu | :DISPlay:CONNect |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|---|-----------------------------|----------------------------|
| continuous | Define Meas menu (meas) | (see specific measurement) |
| count | Delay Trigger menu | :TRIGger:OCCurrence |
| date | Utility menu (system menu) | :SYSTem:DATE |
| dc ac | Channel menu | :CHANnel:COUPLing |
| define | Define Meas menu | :MEASure:DEFine |
| DEFINE MEAS | front-panel key | :MENU:MEASure |
| delay + width - width | Define Meas menu (meas def) | :MEASure:DEFine |
| DELAY | front-panel key | :MEASure:DELay |
| delay | Timebase menu | :TIMebase:DELay |
| delay | Delay Trigger menu | :TRIGger:DELay |
| delay | Waveform Save menu | :WMEMory:XOFFset |
| | (waveform) | |
| delay tv edge pattern state | Trigger menu | :TRIGger:MODE |
| delta t (x) delta v (y) $\Delta t \Delta V$ | Marker menu | :MARKer:XDELta |
| delta t (x) delta v (y) $\Delta t \Delta V$ | Marker menu | :MEASure:TDELta |
| delta t (x) delta v (y) $\Delta t \Delta V$ | Marker menu | :MARKer:YDELta |
| delta t (x) delta v (y) $\Delta t \Delta V$ | Marker menu | :MEASure:VDELta |
| delta t (x) delta v (y) $\Delta t \Delta V$ | Marker menu | :MENU DELTa |
| delta t (x) markers | Marker menu | :DISPlay:TMARker |
| delta t (x) markers | Marker menu | :MARKer[:DISPlay] |
| delta t (x) markers | Marker menu | :MEASure:ESTArt |
| delta t (x) markers | Marker menu | :MEASure:ESTOp |
| delta t (x) markers | Marker menu | :MARKer:X1Position |
| delta t (x) markers | Marker menu | :MEASure:TSTArt |
| delta t (x) markers | Marker menu | :MARKer:X2Position |
| delta t (x) markers | Marker menu | :MEASure:TSTOp |
| delta V (y) markers | Marker menu | :DISPlay:VMARker |
| delta V (y) markers | Marker menu | :MARKer[:DISPlay] |
| delta V (y) markers | Marker menu | :MARKer:Y1Position |
| delta V (y) markers | Marker menu | :MEASure:VSTArt |
| delta V (y) markers | Marker menu | :MARKer:Y2Position |
| delta V (y) markers | Marker menu | :MEASure:VSTOp |
| device mode | Utility menu (HP-IB menu) | :HARDcopy:MODE |
| device mode | Utility menu | :PLOT |
| device mode | Utility menu | :PRINt |
| diff | Waveform Math menu | :FUNCTion:DIFF |
| DISPLAY | front-panel key | :MENU DISPlay |
| display off | Waveform Math menu | :FUNCTion[:DISPlay] |
| display off | Waveform Math menu | :BLANk |
| display off | Waveform Save menu | :BLANk |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|-------------------------------------|--------------------------------|-------------------------|
| display off | Waveform Save menu (pixel) | :PMEMory[:DISPlay] |
| display off | Waveform Save menu (waveform) | :WMEMory[:DISPlay] |
| display on | Waveform Math menu | :FUNction[:DISPlay] |
| display on | Waveform Math menu | :VIEW |
| display on | Waveform Save menu | :VIEW |
| display on | Waveform Save menu (pixel) | :PMEMory[:DISPlay] |
| display on | Waveform Save menu (waveform) | :WMEMory[:DISPlay] |
| DUTY CY | front-panel key | :MEASure:DUTYcycle |
| ECL | Channel menu | :CHANnel:ECL |
| edge pattern state delay tv | Trigger menu | :TRIGger:MODE |
| enter changes | Display menu (seq single-shot) | front panel only |
| env norm avg | Display menu | :ACQuire:TYPE |
| exclude | Display menu (seq single-shot) | :SEQuential:EXCLude |
| execute changes | Display menu (seq single-shot) | front panel only |
| fail if > | Define Meas menu (meas limit) | :MEASure:COMPare |
| failure # | Waveform Save menu (multiple) | :MMEMory:FNUMber |
| FALLTIME | front-panel key | :MEASure:FALLtime |
| FFT | Waveform Math menu | :FUNction:FFT |
| field | TV Trigger menu | :TRIGger:FIELD |
| FINE | front-panel key | (front panel only) |
| f1 f2 f3 f4 | Waveform Math menu | :FUNction < n > |
| form feed | Utility menu (HP-IB menu) | :HARDcopy:PAGE |
| frame axes grid off | Display menu | DISPlay:GRATICule |
| freq span | Waveform Math menu (FFT) | :FUNction:SPAN |
| FREQ | front-panel key | :MEASure:FREQUency |
| from | Define Meas menu (meas def) | :MEASure:DEFine |
| from | Display menu (seq single-shot) | :SEQuential:EXCLude |
| from | Display menu (seq single-shot) | :SEQuential:INCLude |
| grid off frame axes | Display menu | DISPlay:GRATICule |
| HARDCOPY | front-panel key | :PLOT |
| HARDCOPY | front-panel key | :PRINT |
| H X X X | Delay Trigger menu (pattern) | :TRIGger:LOGic |
| H X X X | Pattern Trigger menu | :TRIGger:LOGic |
| holdoff | Edge Trigger menu | :TRIGger:HOLDoff |
| holdoff | Glitch Trigger menu | :TRIGger:GLITCh:HOLDoff |
| holdoff | Pattern Trigger menu | :TRIGger:HOLDoff |
| holdoff | State Trigger menu | :TRIGger:HOLDoff |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|-------------------------------------|--------------------------------|-----------------------------|
| holdoff | TV Trigger menu | :TRIGger:HOLDoff |
| horix magnify | Waveform Math menu (FFT) | :FUNction:MAGNify |
| HP-IB menu | Utility menu | (see specific functions) |
| include | Display menu (seq single-shot) | :SEQuential:INCLude |
| initialize | Utility menu (HP-IB menu) | :HARDcopy:PLOT:INITialize |
| int | Waveform Math menu | :FUNction:INTegrate |
| interpolation | Utility menu (system menu) | front panel only |
| inv | Waveform Math menu | :FUNction:INVert |
| level | Edge Trigger menu | :TRIGger:LEVel |
| level | Glitch Trigger menu | :GLITCh:TRIGger:LEVel |
| level | TV Trigger menu | :TRIGger:LEVel |
| LF rej | Channel menu | :CHANnel:LFReject |
| limit | Define Meas menu | :MEASure:COMPare |
| line | TV Trigger menu | :TRIGger:OCCurrence |
| LOCAL | front-panel key | :SYSTem:KEY |
| lower threshold | Define Meas menu (meas def) | :MEASure:LOWer |
| magnify | Waveform Math menu | :FUNction:MAGNify |
| marker menu on off | Marker menu | :MARKer[:DISPlay] |
| mask menu | Waveform Save menu | front panel only |
| measure | Define Meas menu | (see specific functions) |
| measurements | Define Meas menu (meas def) | (see specific measurements) |
| multiply X | Waveform Math menu | :FUNction:MULTIply |
| noise reject | Edge Trigger menu | :TRIGger:SENSitivity |
| nonvolatile | Waveform Save menu | :STORE |
| norm avg env | Display menu | :ACQuire:TYPE |
| off on | Channel menu | :CHANnel[:DISPlay] |
| off frame axes grid | Display menu | DISPlay:GRATICule |
| offset | Channel menu | :CHANnel:OFFSet |
| offset | Waveform Math menu | :FUNction:OFFSet |
| on fail, save | Define Meas menu (compare) | :MEAS:WCOMP:DESTination |
| on fail, save | Define Meas menu (meas limit) | :MEASure:DESTination |
| or if < | Define Meas menu (meas limit) | :MEASure:COMPare |
| paper length | Utility menu (HP-IB menu) | :HARDcopy:LENGth |
| pattern state delay tv edge | Trigger menu | :TRIGger:MODE |
| peak search | Waveform Math menu (FFT) | :FUNction:PEAK |
| pen | Utility menu (HP-IB) | :HARDcopy:PLOT:PEN |
| PERIOD | front-panel key | :MEASure:PERiod |
| persistence | Display menu (norm) | :DISPlay:PERSiStence |
| pixel | Waveform Save menu | :MERGe |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|-----------------------|--|-----------------------------|
| polarity | TV Trigger menu | :TRIGger:POLarity |
| plot | Utility menu (HP-IB menu) | :HARDcopy:PLOT:AREA |
| present | State Trigger menu | :TRIGger:CONDition |
| probe | Channel menu | :CHANnel:PROBe |
| probe cal menu | Utility menu | (see specific functions) |
| protect | Waveform Save menu | :MMEMory:PROTect |
| protect | Waveform Save menu (multiple) | :WMEMory:PROTect |
| protect | Waveform Save menu (waveform) | :WMEMory:PROTect |
| qualify on | Delay Trigger menu | :TRIGger:QUALify |
| qualify on | TV Trigger menu | :TRIGger:QUALify |
| realtime repetitive | Timebase menu | :TIMEbase:SAMPle |
| RECALL | front-panel key | *RCL |
| RECALL / CLEAR | front-panel keys | *RST |
| reference | Timebase menu | :TIMEbase:REFerence |
| registration form | Utility menu (system menu) | front panel only |
| repetitive realtime | Timebase menu | :TIMEbase:SAMPle |
| RISETIME | front-panel key | :MEASure:RISetime |
| rms | Define Meas menu (meas) | :MEASure:VACRms |
| rms | Define Meas menu (meas) | :MEASure:VDCRms |
| rms | Define Meas menu (meas) | :MEASure:VRMS |
| RUN/STOP | front-panel key | :RUN |
| RUN/STOP | front-panel key | :STOP |
| SAVE | front-panel key | *SAV |
| saved source | Waveform Save menu (multiple) | :MMEMory:SOURce |
| scale-delay | Waveform Save menu | :WMEMory:XOFFset |
| scale-timebase | Waveform Save menu | :WMEMory:XRANge |
| s/div | Timebase menu | :TIMEbase:RANGe |
| search level | Waveform Math menu (FFT) | :FUNCTion:LEVel |
| seconds s | Utility menu (probe cal menu) | :CALibrate:TNUlI |
| seconds per division | Timebase menu | :TIMEbase:RANGe |
| segment # | Display menu (norm seq single-shot) | :SEQuential:NSEGments |
| self cal menu | Utility menu | (front panel only) |
| selftest menu | Utility menu | *TST |
| sensitivity | Waveform Math menu | :FUNCTion:RANGe |
| sequential | Timebase menu | :SEQuential[:DISPlay] |
| service menu | Utility menu | (front panel only) |
| set | Define Meas menu (meas limit) | (see specific measurements) |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|-------------------------------------|----------------------------------|-----------------------------|
| SHOW | front-panel key | :MENU SHOW |
| SINGLE | front-panel key | :TIMEbase:MODE |
| slope | Delay Trigger menu | :TRIGger:DELAy:SLOPe |
| slope | Edge Trigger menu | :TRIGger:SLOPe |
| source | Display menu (seq single-shot) | :SEQuential:SOURce |
| source | Edge Trigger menu | :TRIGger:SOURce |
| source | Marker menu | :MARKer:X1Y1source |
| source | Marker menu | :MARKer:X2Y2source |
| source | TV Trigger menu | :TRIGger:SOURce |
| source | Waveform Save menu (waveform) | :STORe |
| source-state | Glitch Trigger menu | :GLITCh:TRIGger:SOURce |
| standard | Define Meas menu (meas def) | :MEASure:MODE |
| start cal | Utility menu (probe cal menu) | (front panel only) |
| start marker | Marker menu | :MARKer:X1Position |
| start marker | Marker menu | :MEASure:ESTArt |
| start marker | Marker menu | :MEASure:TSTArt |
| state delay tv edge pattern | Trigger menu | :TRIGger:MODE |
| statistics | Define Meas menu (meas) | :MEASure:STATistics |
| stop marker | Marker menu | :MARKer:X2Position |
| stop marker | Marker menu | :MEASure:ESTOp |
| stop marker | Marker menu | :MEASure:TSTOp |
| store | Waveform Save menu (waveform) | :WMEMory:GET |
| store | Waveform Save menu (waveform) | :STORe |
| store failure | Waveform Save menu (multiple) | :MMEMory:STORe |
| subtract - | Waveform Math menu | :FUNCTion:SUBTract |
| talk only addressed | Utility menu (HP-IB menu) | (program language specific) |
| test | Define Meas menu (compare) | :MEAS:WCOMpare:WTEST |
| test | Define Meas menu (meas limit) | :MEASure:LIMittest |
| test all | Utility menu (selftest menu) | *TST |
| threshold | Define Meas menu (meas def) | :MEASure:DEFine |
| thresholds | Define Meas menu (meas def) | :MEASure:LOWer |
| thresholds | Define Meas menu (meas def) | :MEASure:UPPer |
| TIMEBASE | front-panel key | :MENU TIMEbase |
| timebase | Waveform Save menu (waveform) | :WMEMory:XRANge |
| time | Utility menu (system menu) | :SYSTem:TIME |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|-------------------------------------|----------------------------------|---------------------------|
| time null | Utility menu (probe cal menu) | :CALibrate:TNUll |
| to | Define Meas menu (meas def) | :MEASure:DEFine |
| to | Define Meas menu (compare) | :MEAS:WCOMpare:COMPare |
| to | Display menu (seq single-shot) | :SEQuential:EXCLude |
| to | Display menu (seq single-shot) | :SEQuential:INCLude |
| TRIG | front-panel key | :MENU:TRIGger |
| trig'd auto | Trigger menu | :TIMEbase:MODE |
| trigger on | Delay Trigger menu | :TRIGger:OCCurrence |
| trigger on | Delay Trigger menu | :TRIGger:OCCurrence:SLOPe |
| trigger on | Delay Trigger menu | :TRIG:OCCurrence:SOURce |
| trigger on | TV Trigger menu | :TRIGger:OCCurrence |
| trigger on | TV Trigger menu | :TRIGger:OCCurrence:SLOPe |
| TTL | Channel menu | :CHANnel:TTL |
| tv edge pattern state delay | Trigger menu | :TRIGger:MODE |
| upper threshold | Define Meas menu (meas def) | :MEASure:UPPer |
| user defined | Define Meas menu (meas def) | :MEASure:MODE |
| UTIL | front-panel key | :MENU UTILity |
| V AMP | front-panel key | :MEASure:VAMplitude |
| V AVG | front-panel key | :MEASure:VAverage |
| V BASE | front-panel key | :MEASure:VBASe |
| V/div | Channel menu | :CHANnel:RANGe |
| versus vs (WFORM MATH) | Waveform Math menu | :FUNction:VERSus |
| V marker 1 | Marker menu | :MARKer:Y1Position |
| V marker 1 | Marker menu | :MEASure:VSTArt |
| V marker 2 | Marker menu | :MARKer:Y2Position |
| V marker 2 | Marker menu | :MEASure:VSTOP |
| V MAX | front-panel key | :MEASure:VMAX |
| V MIN | front-panel key | :MEASure:VMIN |
| view failures | Waveform Save menu (multiple) | :MMEMory:FNUMber |
| volatile | Waveform Save menu (pixel) | :MERGe |
| volts % | Define Meas menu (meas def) | :MEASure:UNITs |
| volts per division | Channel menu | :CHANnel:RANGe |
| V P-P | front-panel key | :MEASure:VPP |
| V RMS | front-panel key | :MEASure:VRMS |
| V TOP | front-panel key | :MEASure:VTOP |
| waveform | Waveform Save menu | :STORE |
| when | Delay Trigger menu | :TRIGger:LOGic |
| when | Glitch Trigger menu | :TRIGger:GLITch:CONDition |
| when | Pattern Trigger menu | :TRIGger:CONDition |

Table 4-2. Front-Panel Function to Command Cross-Reference

| Front-Panel Function | Location | Command |
|----------------------|--------------------------|-----------------------|
| when | State Trigger menu | :TRIGger:LOGic |
| width | Glitch Trigger menu | :TRIGger:GLITch:WIDTh |
| window | Waveform Math menu (FFT) | :FUNCTion:WINDow |
| WFORM MATH | front-panel key | :MENU MATH |
| WFORM SAVE | front-panel key | :MENU SAVE |
| X multiply | Waveform Math menu | :FUNCTion:MULTIply |
| x delta | Marker menu | :MARKer:XDELta |
| x delta | Marker menu | :MEASure:TDELta |
| x marker | Marker menu | :DISPlay:TMARker |
| x marker | Marker menu | :MARKer[:DISPlay] |
| x1 marker | Marker menu | :MARKer:X1Position |
| x1 marker | Marker menu | :MEASure:TSTArt |
| x2 marker | Marker menu | :MARKer:X2Position |
| x2 marker | Marker menu | :MEASure:TSTOp |
| y delta | Marker menu | :MARKer:YDELta |
| y delta | Marker menu | :MEASure:VDELta |
| y marker | Marker menu | :DISPlay:VMARker |
| y marker | Marker menu | :MARKer[:DISPlay] |
| y1 marker | Marker menu | :MARKer:Y1Position |
| y1 marker | Marker menu | :MEASure:VSTArt |
| y2 marker | Marker menu | :MARKer:Y2Position |
| v2 marker | Marker menu | :MEASure:VSTOp |

The Command Tree

The command tree in figure 4-1 shows all of the commands in the HP 54505B/54506B/54510B/54512B, and the relationship of the commands to each other. The IEEE 488.2 common commands are not listed as part of the command tree since they do not affect the position of the parser within the tree. When a program message terminator (<NL>, linefeed - ASCII decimal 10) or a leading colon (:) is sent to the instrument, the parser is set to the "root" of the command tree.

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

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

Common Commands

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

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

Example:

```
*RST.
```

Root Level Commands

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

Example:

```
:AUTOSCALE
```

Subsystem Commands

Subsystem commands are grouped together under a common node of the command tree, such as the TIMEBASE commands. Only one subsystem may be selected at any given time. When the instrument is initially turned on, the command parser is set to the root of the command tree, therefore, no subsystem is selected.

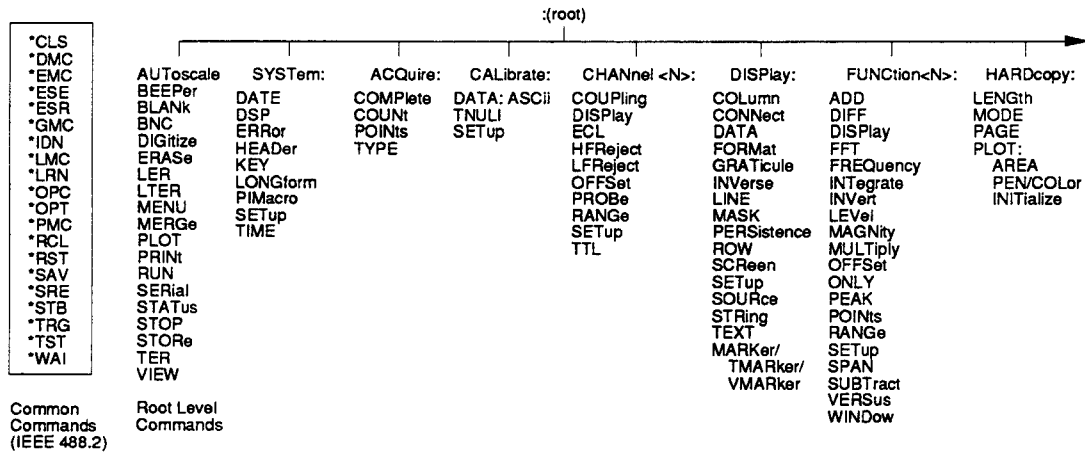


Figure 4-1. HP54505B/54506B/54510B/54512B Command Tree

| MEASure: | TIMEbase: | TRIGger: | WAVeform: | WMEMory: | PMEMory: | MMEMory: | SEQuential: | MARKer: |
|-------------|-----------|-------------|------------|----------|----------|----------|-------------|------------|
| ALL | DELAy | CENTEred | DATA | DISPlay | CLEar | DISPlay | DISPlay | DISPlay |
| COMPare | MODE | CONDiion | FORMat | GET | DISPlay | FNUMber | NPOints | SETup |
| CURSor | RANGE | COUPLing | POINts | PROTEct | MERGE | SOURce | NSEGments | X1Position |
| DEFine | REFerence | DELAy | PREAmble | SETup | SETup | STORe | SNUMber | X2Position |
| DELAy | SAMPle | DELAy: | SOURce | XOFFset | | | SETup | Y1Position |
| DESTination | SETup | SLOPe | TYPE | XRANGE | | | SOURce | Y2Position |
| DUTYcycle | | SOURce | XINCrement | YOFFset | | | EXCLude | X1Y1source |
| ESTArt | | FIELD | XORigin | YRANGE | | | INCLude | X2Y2source |
| ESTOp | | GLITch: | XREFerence | | | | | XDELta |
| FALLtime | | CENTEred | YINCrement | | | | | YDELta |
| FREQuency | | HOLDoff | YORigin | | | | | |
| LIMittest | | LEVel | YREFerence | | | | | |
| LOWer | | SOURce | | | | | | |
| MODE | | WIDTh | | | | | | |
| NWIDth | | HOLDoff | | | | | | |
| OVERshoot | | LEVel | | | | | | |
| PERiod | | LINE | | | | | | |
| POSTfailure | | LOGic | | | | | | |
| PREShoot | | MODE | | | | | | |
| PWIDth | | OCCurrence | | | | | | |
| RESults | | OCCurrence: | | | | | | |
| RISetime | | SLOPe | | | | | | |
| SCRatch | | SOURce | | | | | | |
| SOURce | | PATH | | | | | | |
| STATistics | | POLarity | | | | | | |
| TDELta | | PROBe | | | | | | |
| TMAX | | QUALity | | | | | | |
| TMIN | | SENSitivity | | | | | | |
| TSTArt | | SETup | | | | | | |
| TSTOp | | SLOPe | | | | | | |
| TVOLT | | SOURce | | | | | | |
| UNITs | | STANdard | | | | | | |
| UPPer | | | | | | | | |
| VACRms | | | | | | | | |
| VAMPititude | | | | | | | | |
| VAVerage | | | | | | | | |
| VBASe | | | | | | | | |
| VDCRms | | | | | | | | |
| VDELta | | | | | | | | |
| VFIFty | | | | | | | | |
| VMAX | | | | | | | | |
| VMIN | | | | | | | | |
| VPP | | | | | | | | |
| VRELative | | | | | | | | |
| VRMS | | | | | | | | |
| VSTArt | | | | | | | | |
| VSTOp | | | | | | | | |
| VTIME | | | | | | | | |
| VTOP | | | | | | | | |
| WCOMpare: | | | | | | | | |
| WTEST | | | | | | | | |
| COMPare | | | | | | | | |
| ALLowance | | | | | | | | |
| DESTination | | | | | | | | |
| POSTfailure | | | | | | | | |

Figure 4-1. HP54505B/54506B/54510B/54512B Command Tree (cont)

Tree Traversal Rules

Command headers are created by traversing down the command tree. A legal command header from the command tree in figure 4-1 would be :CHANNEL1:RANGE. This is referred to as a compound header. A compound header is a header made of two or more mnemonics separated by colons. The mnemonic created contains no spaces. The following rules apply to traversing the tree:

- A leading colon or a < program message terminator > (either an < NL > or EOI true on the last byte) places the parser at the root of the command tree. A leading colon is a colon that is the first character of a program header.
- Executing a subsystem command places you in that subsystem until a leading colon or a < program message terminator > is found. In the Command Tree, figure 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 (CHANNEL < N > :). That is the point where the parser resides. Any command below that point can be sent within the current program message without sending the mnemonics which appear above them (for example, OFFSET).

Examples

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

Example 1:

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

Comments:

The colon between CHANNEL1 and RANGE is necessary because CHANNEL1:RANGE is a compound command. The semicolon between the RANGE command and the OFFSET command is the required program message unit separator. The OFFSET command does not need CHANNEL1 preceding it, since the CHANNEL1:RANGE command sets the parser to the CHANNEL1 node in the tree.

Example 2:

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

or

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

Comments:

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

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

Example 3:

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

Comments:

The leading colon before CHANNEL1 tells the parser to go back to the root of the command tree. The parser can then see the CHANNEL1:OFFSET command.

Infinity Representation

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

Sequential and Overlapped Commands

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

Response Generation

As defined by IEEE 488.2, query responses may be buffered for the following conditions:

- When the query is parsed by the instrument.
- When the controller addresses the instrument to talk so that it may read the response.

The oscilloscope buffers responses to a query when the query is parsed.

Notation Conventions and Definitions

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

Conventions

< > Angle brackets enclose words or characters that symbolize a program code parameter or an HP-IB command.

::= "is defined as." For example, **<A> ::= ** indicates that **<A>** can be replaced by **** in any statement containing **<A>**.

| "or." Indicates a choice of one element from a list. For example, **<A> | ** indicates **<A>** or ****, but not both.

... An ellipsis (trailing dots) indicates that the preceding element may be repeated one or more times.

[] Square brackets indicate that the enclosed items are optional.

{ } When several items are enclosed by braces, one, and only one of these elements must be selected.

Definitions

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

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

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

<sp> ::= <white space>

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

Syntax Diagrams

Chapters 6 through 23 contain syntax diagrams showing the proper syntax for each command. All characters contained in a circle or oblong are literals, and must be entered exactly as shown. Words and phrases contained in rectangles are names of items used with the command and are described in the accompanying text of each command. Each line can only be entered from one direction as indicated by the arrow on the entry line. Any combination of commands and arguments that can be generated by following the lines in the proper direction is syntactically correct. An argument is optional if there is a path around it. Where there is a rectangle which contains the word "space," a white space character must be entered. White space is optional in many other places.

Program Examples

The program examples given for each command in chapters 6 through 23 were written on an HP 9000 Series 200/300 controller using the HP BASIC 5.0 programming language. The programs always assume the oscilloscope is at address 707. If a printer is used, it is always assumed to be at address 701.

In these examples, pay special attention to the ways in which the command or query can be sent. The way the instrument is set up to respond to a command or query has no bearing on how you send the command or query. That is, the command or query can be sent using the long form or short form, if a short form exists for that command. You can send the command or query using upper case (capital) letters or lower case (small) letters. Also, the data can be sent using almost any form you wish. If you are sending a channel 1 range value of 100 mV, that value could be sent using a decimal (.1), or an exponential (1e-1 or 1.0E-1), or a suffix (100 mV or 100MV).

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

- Commands in long form using the decimal format.

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

- Commands in short form using an exponential format.

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

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

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



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

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

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

Command Set Organization

The command set for the oscilloscope is divided into common commands, root level commands and 16 sets of subsystem commands. Each of the 18 groups of commands is described in the following chapters. Each of the chapters contain a brief description of the subsystem, a set of syntax diagrams for the commands, and the commands for each subsystem in alphabetic order.

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

The sixteen subsystems in the HP 54505B, HP 54506B, HP 54510B, and HP 54512B are listed below:

SYSTem - controls some basic functions of the oscilloscope.

ACQuire - sets the parameters for acquiring and storing data.

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

CHANnel - controls all Y-axis oscilloscope functions.

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

FUNCTION - controls the waveform math functions of the oscilloscope.

HARDcopy - controls the parameters used during the plotting or printing of waveforms.

MEASure - selects the automatic measurements to be made.

TIMEbase - controls all X-axis oscilloscope functions.

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

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

WMEMory - controls waveform memory functions.

PMEMory - controls pixel memory functions.

MMEMory - controls multiple memory functions.

SEQuential - controls sequential memory mode configuration and display parameters.

MARKer - controls x- and y-marker functions.

Table 4-3 lists the commands for the oscilloscope in alphabetical order with their corresponding subsystem or command type.

Table 4-3. Alphabetic Command Cross-Reference

| Command | Where Used |
|--------------|----------------------|
| ADD | FUNCTION Subsystem |
| ALL | MEASURE Subsystem |
| AUTOScale | Root Level Command |
| BEEPer | Root Level Command |
| BLANK | Root Level Command |
| BNC | Root Level Command |
| CENTERed | TRIGGER Subsystem |
| CLEAR | PMEMORY Subsystem |
| *CLS | Common Command |
| COLumn | DISPLAY Subsystem |
| COMPare | MEASURE Subsystem |
| COMPLete | ACQUIRE Subsystem |
| CONDition | TRIGGER Subsystem |
| CONNect | DISPLAY Subsystem |
| COUNT | ACQUIRE Subsystem |
| COUPLing | CHANNEL Subsystem |
| COUPLing | TRIGGER Subsystem |
| CURSor | MEASURE Subsystem |
| DATA | DISPLAY Subsystem |
| DATA | WAVEFORM Subsystem |
| DATA:ASCii | CALIBRATE Subsystem |
| DATE | SYSTEM Subsystem |
| DEFine | MEASURE Subsystem |
| DELay | MEASURE Subsystem |
| DELay | TIMeBASE Subsystem |
| DELay | TRIGGER Subsystem |
| DELay:SLOPe | TRIGGER Subsystem |
| DELay:SOURce | TRIGGER Subsystem |
| DESTination | MEASURE Subsystem |
| DIFF | FUNCTION Subsystem |
| DIGitize | Root Level Command |
| [DISPlay] | CHANNEL Subsystem |
| [DISPlay] | FUNCTION Subsystem |
| [DISPlay] | MARKER Subsystem |
| [DISPlay] | MMEMORY Subsystem |
| [DISPlay] | PMEMORY Subsystem |
| [DISPlay] | SEQUENTIAL Subsystem |
| [DISPlay] | WMEMORY Subsystem |
| *DMC | Common Command |
| DSP | SYSTEM Subsystem |

Table 4-3. Alphabetic Command Cross-Reference

| Command | Where Used |
|-----------------|----------------------|
| DUTycle | MEASure Subsystem |
| ECL | CHANnel Subsystem |
| ERASe | Root Level Command |
| ERRor | SYSTem Subsystem |
| *EMC | Common Command |
| *ESE | Common Command |
| *ESR | Common Command |
| ESTArt | MEASure Subsystem |
| ESTOp | MEASure Subsystem |
| EXCLude | SEQuential Subsystem |
| FALLtime | MEASure Subsystem |
| FFT | FUNCTion Subsystem |
| FIELD | TRIGger Subsystem |
| FNUMber | MMEMory |
| FORMat | DISPlay Subsystem |
| FORMat | WAVeform Subsystem |
| FREQUency | FUNCTion Subsystem |
| FREQUency | MEASure Subsystem |
| GET | WMEMory Subsystem |
| GLITch | TRIGger Subsystem |
| GLITch:CENTered | TRIGger Subsystem |
| GLITch:HOLDoff | TRIGger Subsystem |
| GLITch:LEVel | TRIGger Subsystem |
| GLITch:SOURce | TRIGger Subsystem |
| GLITch:WIDTh | TRIGger Subsystem |
| *GMC | Common Command |
| GRATicule | DISPlay Subsystem |
| HEADer | SYSTem Subsystem |
| HFReject | CHANnel Subsystem |
| HOLDoff | TRIGger Subsystem |
| *IDN | Common Command |
| INCLude | SEQuential Subsystem |
| INTegrate | FUNCTion Subsystem |
| INVerse | DISPlay Subsystem |
| INVert | FUNCTion Subsystem |
| KEY | SYSTem Subsystem |
| LENGth | HARDcopy Subsystem |
| LER | Root Level Command |
| LEVel | FUNCTion Subsystem |
| LEVel | TRIGger Subsystem |

Table 4-3. Alphabetic Command Cross-Reference

| Command | Where Used |
|-------------------|----------------------|
| LFReject | CHANnel Subsystem |
| LIMittest | MEASure Subsystem |
| LINE | DISPlay Subsystem |
| LINE | TRIGger Subsystem |
| *LMC | Common Command |
| LOGic | TRIGger Subsystem |
| LONGform | SYSTEM Subsystem |
| LOWer | MEASure Subsystem |
| *LRN | Common Command |
| LTER | Root Level Command |
| MAGNify | FUNCTion Subsystem |
| MASK | DISPlay Subsystem |
| MENU | Root Level Command |
| MERGe | PMEMory Subsystem |
| MERGe | Root Level Command |
| MODE | HARDcopy Subsystem |
| MODE | MEASure Subsystem |
| MODE | TIMEbase Subsystem |
| MODE | TRIGger Subsystem |
| MULTiply | FUNCTion Subsystem |
| NPOints | SEQUential Subsystem |
| NSEGments | SEQUential Subsystem |
| NWIDTH | MEASure Subsystem |
| OCCurrence | TRIGger Subsystem |
| OCCurrence:SLOPe | TRIGger Subsystem |
| OCCurrence:SOURce | TRIGger Subsystem |
| OFFSet | CHANnel Subsystem |
| OFFSet | FUNCTion Subsystem |
| ONLY | FUNCTion Subsystem |
| *OPC | Common Command |
| *OPT | Common Command |
| OVERshoot | MEASure Subsystem |
| PAGE | HARDcopy Subsystem |
| PATH | TRIGger Subsystem |
| PEAK | FUNCTion Subsystem |
| PERiod | MEASure Subsystem |
| PERSistence | DISPlay Subsystem |
| PIMacro | SYSTEM Subsystem |
| PLOT | Root Level Command |
| PLOT:AREA | HARDcopy Subsystem |

Table 4-3. Alphabetic Command Cross-Reference

| Command | Where Used |
|-----------------|----------------------|
| PLOT:COLor | HARDcopy Subsystem |
| PLOT:INITialize | HARDcopy Subsystem |
| PLOT:PEN COLor | HARDcopy Subsystem |
| *PMC | Common Commands |
| POINTs | ACQuire Subsystem |
| POINTs | FUNCTion Subsystem |
| POINTs | WAVeform Subsystem |
| POLarity | TRIGger Subsystem |
| POSTfailure | MEASure Subsystem |
| PREamble | WAVeform Subsystem |
| PREShoot | MEASure Subsystem |
| PRINT | Root Level Command |
| PROBe | CHANnel Subsystem |
| PROBE | TRIGger Subsystem |
| PROTECT | MEMory Subsystem |
| PWIDTH | MEASure Subsystem |
| QUALify | TRIGger Subsystem |
| RANGE | CHANnel Subsystem |
| RANGE | FUNCTion Subsystem |
| RANGE | TIMebase Subsystem |
| *RCL | Common Command |
| REFerence | TIMebase Subsystem |
| RESults | MEASure Subsystem |
| RISetime | MEASure Subsystem |
| ROW | DISPlay Subsystem |
| *RST | Common Command |
| RUN | Root Level Command |
| SAMPle | TIMebase Subsystem |
| *SAV | Common Command |
| SCRatch | MEASure Subsystem |
| SCReen | DISPlay Subsystem |
| SENSitivity | TRIGger Subsystem |
| SERial | Root Level Command |
| SETup | CALibrate Subsystem |
| SETup | CHANnel Subsystem |
| SETup | DISPlay Subsystem |
| SETup | FUNCTion Subsystem |
| SETup | MARKer Subsystem |
| SETup | PMEMory Subsystem |
| SETup | SEQUential Subsystem |

Table 4-3. Alphabetic Command Cross-Reference

| Command | Where Used |
|------------|----------------------|
| SETup | SYSTEM Subsystem |
| SETup | TIMEbase Subsystem |
| SETup | TRIGger Subsystem |
| SETup | WMEMory Subsystem |
| SLOPe | TRIGger Subsystem |
| SPAN | FUNcTION Subsystem |
| SNUMber | SEQuential Subsystem |
| SOURce | DISPlay Subsystem |
| SOURce | MEASure Subsystem |
| SOURce | MMEMory Subsystem |
| SOURce | SEQuential Subsystem |
| SOURce | TRIGger Subsystem |
| SOURce | WAVEform Subsystem |
| *SRE | Common Command |
| STANdard | TRIGger Subsystem |
| STATistics | MEASure Subsystem |
| STATus | Root Level Command |
| *STB | Common Command |
| STOP | Root Level Command |
| STORE | MMEMory Subsystem |
| STORE | Root Level Command |
| STRing | DISPlay Subsystem |
| SUBTract | FUNcTION Subsystem |
| TDELta | MEASure Subsystem |
| TER | Root Level Command |
| TEXT | DISPlay Subsystem |
| TIME | SYSTEM Subsystem |
| TMARKer | DISPlay Subsystem |
| TMAX | MEASure Subsystem |
| TMIN | MEASure Subsystem |
| TNULI | CALibrate Subsystem |
| *TRG | Common Command |
| *TST | Common Command |
| TSTArt | MEASure Subsystem |
| TSTOp | MEASure Subsystem |
| TTL | CHANnel Subsystem |
| TVOLt | MEASure Subsystem |
| TYPE | ACQuire Subsystem |
| TYPE | WAVEform Subsystem |
| UNITs | MEASure Subsystem |

Table 4-3. Alphabetic Command Cross-Reference

| Command | Where Used |
|----------------------|--------------------|
| UPPer | MEASure Subsystem |
| VACRms | MEASure Subsystem |
| VAMPlitude | MEASure Subsystem |
| VAVerage | MEASure Subsystem |
| VBASe | MEASure Subsystem |
| VDCRms | MEASure Subsystem |
| VDELta | MEASure Subsystem |
| VERSus | FUNCTion Subsystem |
| VFIFty | MEASure Subsystem |
| VIEW | Root Level Command |
| VMARker | DISPlay Subsystem |
| VMAX | MEASure Subsystem |
| VMIN | MEASure Subsystem |
| VPP | MEASure Subsystem |
| VRELative | MEASure Subsystem |
| VRMS | MEASure Subsystem |
| VSTArt | MEASure Subsystem |
| VSTOp | MEASure Subsystem |
| VTIMe | MEASure Subsystem |
| VTOP | MEASure Subsystem |
| *WAI | Common Command |
| WCOMpare | MEASure Subsystem |
| WCOMpare:WTEST | MEASure Subsystem |
| WCOMpare:COMPare | MEASure Subsystem |
| WCOMpare:ALLowance | MEASure Subsystem |
| WCOMpare:DESTination | MEASure Subsystem |
| WCOMpare:POSTfailure | MEASure Subsystem |
| WINDow | FUNCTion Subsystem |
| XDELta | MARKer Subsystem |
| XINCrement | WAVEform Subsystem |
| XOFFset | WMEMory Subsystem |
| XORigin | WAVEform Subsystem |
| XRANge | WMEMory Subsystem |
| XREFerence | WAVEform Subsystem |
| X1Position | MARKer Subsystem |
| X1Y1source | MARKer Subsystem |
| X2Position | MARKer Subsystem |
| X2Y2source | MARKer Subsystem |
| YDELta | MARKer Subsystem |
| YINCrement | WAVEform Subsystem |

Table 4-3. Alphabetic Command Cross-Reference

| Command | Where Used |
|----------------|---------------------|
| YOFFset | WMEMemory Subsystem |
| YORigin | WAVEform Subsystem |
| YRANge | WMEMemory Subsystem |
| YREFerence | WAVEform Subsystem |
| Y1Position | MARKer Subsystem |
| Y2Position | MARKer Subsystem |
| YREFerence | WAVEform Subsystem |

Example Programs

Introduction

This chapter contains example programs using the command set for the HP 54505B, HP 54506B, HP 54510B, and HP 54512B. In general, the programs use the long form of the command with alpha (as opposed to numeric) arguments. Each command has a separate output statement for clarity. To optimize speed, switch to the concatenated short form numerics.

Throughout these examples, the oscilloscope is assumed to be at address 7, the hardcopy devices at address 1. The HP-IB system bus address is assumed to be at 7, and the national interface address is derived using the IBDEV command. The input signal used is the AC CAL signal from the rear panel of the instrument. This signal is connected to channel 1 through a coaxial cable (or 1:1 probe).

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

- HP BASIC 5.0 running on an HP Series 200/300 controller.
- Microsoft QuickBASIC 4.5 running on IBM PC or compatibles.
- Microsoft QuickC 2.5 running on IBM PC or compatibles.

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

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

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

- **HP-BASIC** contains examples on HP BASIC 5.0 running on an HP Series 200/300 controller.

- **QB-HPIB** contains examples on Microsoft QuickBASIC 4.5 running on IBM PC or compatibles using the HP 82335A HP-IB interface module.
- **QB-NI** contains examples on Microsoft QuickBASIC 4.5 running on IBM PC or compatibles using the National Instruments PCII or PCIA IEEE 488.2 interface module.
- **QC-HPIB** contains examples on Microsoft QuickC 2.5 running on IBM PC or compatibles using the HP 82335A HP-IB interface module.
- **QC-NI** contains examples on Microsoft QuickC 2.5 running on IBM PC or compatibles using the National Instruments PCII or PCIA IEEE 488.2 interface module.

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

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

Order of Commands

The order of the commands in a program may be critical to the task you are trying to accomplish. If your commands do not follow the proper order, later commands may cancel out earlier commands and conditions you thought were setup are no longer valid. Table 5-1 lists the order of some commonly used commands in the TIMEbase, CHANnel, TRIGger, ACQuire, and WAVEform subsystems. For more information, refer to the specific commands in this manual.

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

Table 5-1. Order of Commands

| | First | Second | Third | Fourth | Fifth |
|-------------------|---------------------------------------|--|---|--|---|
| Subsystem: | :TIMEbase | :CHANnel {1 2 3 4} | :TRIGger | :ACQUIRE | :WAVEform |
| First | :MODE AUTO TRIGgered SINGle | :PROBe (ratio to one: .9 - 1000) | :MODE EDGE PATTern STATe DELay GLITCh TV | :TYPE NORMAl AVERAge ENVELOpe RAWDData (len,acq,type) | :FORMat ASCii WORD BYTE COMPRESSED |
| Second | :SAMPLing REALtime REPetitive | :RANGe (Volts/screen) | :SOURce CHANnel1 CHANnel2 CHANnel3 CHANnel4 EXTernal | :COMPLete (0 to 100%) | :SOURce CHANnel1 CHANnel2 CHANnel3 CHANnel4 WAVEform1 WAVEform2 WAVEform3 WAVEform4 |
| Third | :REFerence LEFT CENTer RIGHT | :OFFSet (Volts @ center screen) | :LEVel (trigger@Volts) | :POINts (500/8000) | |
| Fourth | :RANGe (s/screen) | :COUPLing AC DC DCFiFty | :SLOPe POSitive NEGative | :COUNt (1 - 2048) | |
| Fifth | :DELay (delay value) | | | | |

HP Basic Initialize Program

This sample program demonstrates how to initialize the interface and oscilloscope, digitize the input signal (in this case channel 1), then measure frequency and print the result. The program assumes that the AC CAL signal from the rear panel of the instrument is connected to channel 1 through a coaxial cable.

```
10  !INIT                HP Basic program
20  !
30  !MAIN PROGRAM
40  !
50  CLEAR SCREEN
60  PRINT "This example program will perform the following tasks:"
70  PRINT "      a. initialize the interface and scope"
80  PRINT "      b. digitize signal"
90  PRINT "      c. measure and print the frequency"
100 PRINT
110 PRINT "The program assumes the system is configured as:"
120 PRINT "      HP-IB address = 7"
130 PRINT "      scope address = 7"
140 PRINT "      signal attached to channel 1"
150 PRINT
160 PRINT "If the addresses are not correct for your configuration, change"
170 PRINT "the ASSIGN statements in the Initialize function."
180 PRINT
190 PRINT "Press Continue when ready to start program, or Shift/Break to terminate."
200 PAUSE
210 GOSUB Initialize      !initialize interface and scope
220 GOSUB Get_waveform   !digitize signal
230 GOSUB Measure       !measure and print frequency
240 STOP
250 !
260 !INITIALIZE INTERFACE AND SCOPE
270 !
280 Initialize:         !
290 ASSIGN @Scope TO 707 !scope address
300 ASSIGN @Isc TO 7    !HP-IB address
310 CLEAR @Isc          !clear HP-IB interface
320 OUTPUT @Scope;"*RST" !set scope to default config
330 OUTPUT @Scope;":AUTOSCALE" !AUTOSCALE
340 OUTPUT @Scope;":SYST:HEADER OFF" !turn headers off
350 CLEAR SCREEN       !clear screen
360 RETURN
370 !
380 !DIGITIZE waveform to acquire data and stop scope for further
390 !measurement. Measurement is NOT displayed on front panel.
```

```

400      !
410 Get_waveform:      !
420  OUTPUT @Scope;" :WAVEFORM:SOURCE CHAN1" !set source to channel 1
430  OUTPUT @Scope;" :DIGITIZE CHAN1"      !macro to acquire data & stop
440  RETURN
450      !
460      !have scope to a frequency measurement and read results into
470      !controller.
480      !
490 Measure:          !
500  OUTPUT @Scope;" :MEASURE:FREQUENCY?"    !FREQUENCY query
510  ENTER @Scope;Value      !read from scope
520  PRINT "FREQUENCY = ";Value;"Hz"
521  OUTPUT @Scope;" :MEAS:VPP?"            !Vpp query
522  ENTER @Scope;Value
523  PRINT "Vpp = ";Value;"V"
530  RETURN
540  END

```

HP Basic Digitize Program

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

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

```

390 STOP
400 !
410 !INITIALIZE INTERFACE AND SCOPE
420 Initialize: !
430 PRINT "INITIALIZE"
440 ASSIGN @Isc TO 7 !Interface Select Code = 7
450 ASSIGN @Scope TO 707 !scope address
460 CLEAR @Isc !clear HP-IB interface
470 OUTPUT @Scope;"*RST" !set scope to default config
480 OUTPUT @Scope;"AUTOSCALE" !AUTOSCALE
490 OUTPUT @Scope;"system:header off" !turn headers off
500 !
510 !the following 4 commands are the default configuration setting
520 !that the RST sets up; but, they are included here for clarity and
530 !completeness. This ensures scope is configured for DIGITIZE, if
540 !*RST was not done.
550 !
560 OUTPUT @Scope;"WAVEFORM:SOURCE CHAN1" !set source to channel 1
570 OUTPUT @Scope;"ACQUIRE:TYPE NORMAL" !set to normal acquisition mode
580 OUTPUT @Scope;"ACQUIRE:COMPLETE 100" !set complete criteria
590 OUTPUT @Scope;"ACQUIRE:POINTS 500" !set # of pts to 4000
600 !
610 OUTPUT @Scope;"WAVEFORM:FORMAT COMP" !acquired data in word format
620 CLEAR SCREEN
630 RETURN
640 !
650 !
660 !complete scope configuration; DIGITIZE and acquire waveform data
670 !and preamble (voltage/timing) information into computer.
680 !
690 Get_waveform: !
700 OUTPUT @Scope;"DIGITIZE CHAN1" !macro to acquire data & stop
710 OUTPUT @Scope;"WAVEFORM:DATA?" !query scope for data
720 ENTER @Scope USING "#,1A,";One_char$ !strip off header & size
730 IF One_char$="#" THEN Found_pound
740 PRINT "BAD DATA"
750 STOP
760 Found_pound: !read record length from header
770 ENTER @Scope USING "#,1D";Digits !get length of record
780 ENTER @Scope USING "#,&VAL$(Digits)&"D";Length
790 PRINT "reading ";Length;" bytes from scope"
800 !
810 !allocate an array for the waveform data. After the array has been
820 !read in, one extra byte read is done to input the line feed (10)
830 !attached to the end of the scope's output buffer.
840 !
850 ALLOCATE INTEGER Waveform(1:Length)
860 ENTER @Scope USING "#,B";Waveform(*) !read waveform information
870 ENTER @Scope USING "-K,B";End$ !get last byte (line feed)

```

```

880 OUTPUT @Scope;":WAVEFORM:PREAMBLE?"      !query for preamble
890 ENTER @Scope;Preamble(*)                 !read preamble information
900 RETURN
910 !
920 !
930 !plot waveform data to display
940 !
950 Graph:                                     !
960 GCLEAR                                    !initialize graphics
970 CLEAR SCREEN
980 GINIT
990 GRAPHICS ON
1000 VIEWPORT 0,130,35,100
1010 WINDOW 1,Preamble(3),0,255
1020 FRAME
1030 PEN 4
1040 MOVE 0,0
1050 FOR I=1 TO Preamble(3)                   !plot data points
1060     MOVE I,Waveform(I)
1070     DRAW I,Waveform(I)
1080 NEXT I
1090 Srange=Preamble(3)*Preamble(5)/10
1100 Vrange=256*Preamble(8)
1110 Offset=(128-Preamble(10))*Preamble(8)+Preamble(9)
1120 PRINT TABXY(0,18),"Vertical=";Vrange/8;" V/div";TAB(50),"Offset = ";Offset;"V"
1130 PRINT TABXY(0,19),"Time=";Srange;" s/div"
1140 RETURN
1150 !
1160 !save waveform data and preamble information to computer disk
1170 !
1180 Save_waveform:                             !
1190 ON ERROR GOTO 1210
1200 PURGE "WAVESAMPLE"
1210 OFF ERROR
1220 CREATE BDAT "WAVESAMPLE",1,4080
1230 ASSIGN @Path TO "WAVESAMPLE"
1240 OUTPUT @Path;Waveform(*),Preamble(*)
1250 RETURN
1260 !
1270 !retrieve waveform data and preamble information from disk
1280 !
1290 Retrieve_wave:                             !
1300 ASSIGN @Path TO "WAVESAMPLE"
1310 ENTER @Path;Waveform(*),Preamble(*)
1320 RETURN
1330 END

```


Microsoft QuickBASIC Initialize Program

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

```
DECLARE SUB get.waveform ()
DECLARE SUB measure ()
DECLARE SUB Initialize ()
DECLARE SUB send (cmd$)
'
'Filename : \QB-HP-IB\INIT.bas
'
REM $INCLUDE: 'c:\hpib\qbsetup'
DIM SHARED isc&, scope&

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

CALL Initialize                    'initialize interface and scope
CALL get.waveform                  'tell scope to acquire data
CALL measure                       'measure frequency of input signal

END

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

```

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

CALL ioreset(isc&)                'set interface to start-up state
IF pcib.err noerr THEN ERROR pcib.baserr 'check for error
timeout! = 10!
CALL iotimeout(isc&, timeout!)    'set timeout to 10 seconds
IF pcib.err noerr THEN ERROR pcib.baserr 'check for error
CALL ioclear(isc&)                'clear devices attached to the interfaces to known
state
IF pcib.err noerr THEN ERROR pcib.baserr 'check for error
CLS

    send ("*cls")                  'clear status registers
    send ("*RST")                  'reset the scope
    send (":autoscale")            'autoscale the input signal
    send (":system:header off")    'turn headers off
    ,
    'The following acquire subsystem commands are used to specify
    'the acquisition mode, percent completion, and number of points
    'for the DIGITIZE. These are the defaults set by the *RST, but
    'are included here to stress the fact that you can control them.
    ,
    send (":acquire:type normal")  'normal acquisition mode
    send (":acquire:complete 100") '100% completion criteria
    send (":acquire:points 500")   '"500 points acquired
    CLS                             'clear PC screen
END SUB

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

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

```

Microsoft QuickBASIC Digitize Program

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

```
DECLARE SUB initialize ()
DECLARE SUB capture.waveform ()
DECLARE SUB measure ()
DECLARE SUB get.waveform ()
DECLARE SUB save.waveform ()
DECLARE SUB retrieve.waveform ()
DECLARE SUB graph ()
DECLARE SUB integrate ()
DECLARE SUB send (cmd$)
'
'Filename : \QB-HP-IB\DIGI.bas
'
REM $INCLUDE: 'c:\hpib\qbsetup'

DIM SHARED preamble!(10)           'preamble data array
DIM SHARED waveform%(8000)         'waveform data array
DIM SHARED scope&, isc&            'scope/hpib address variable
'
'MAIN PROGRAM
'
CLS                                 'clear screen

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

```

PRINT "      scope address = 7"
PRINT "Set these variables as needed for your system."
PRINT
PRINT "PRESS:  1 to CONTINUE "
PRINT "      2 to TERMINATE"
BEEP
PRINT "?";
LINE INPUT cmd$
IF cmd$ = "2" THEN SYSTEM
isc& = 7                                'HP-IB Interface Select Code
scope& = 707                            'isc& * 100 + scope address

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

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

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

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

send (":waveform:data?")                'query for waveform data
max.len% = 8000                          'max size of data block
actual.len% = 0
CALL ioenterab(scope&, SEG waveform%(1), max.len%, actual.len%, 2)
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR
max.len% = 1                             'get new line char at end of data block
actual.len% = 0
msg$ = SPACE$(max.len%)                 'allocate the space
CALL ioenters(scope&, msg$, max.len%, actual.len%)
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR

END SUB

```

```

SUB graph
SCREEN 9
' set screen mode to EGA 64k
' 640 x 350 x 16
VIEW (1, 1)-(500, 256), , 15
' set viewport and draw the border
WINDOW (0, 0)-(10, 8)
' prepare to draw the grid
COLOR 8
' set color
FOR i% = 1 TO 9
  LINE (i%, 0)-(i%, 8)
NEXT i%
FOR i% = 1 TO 7
  LINE (0, i%)-(10, i%)
NEXT i%
WINDOW (1, 0)-(preamble!(3), 32767)
' use window to set coordinates
COLOR 10
FOR i% = 1 TO preamble!(3)
  PSET (i%, waveform%(i%))
' draw data point
NEXT i%

WIDTH 80, 25
VIEW PRINT 20 TO 24
send (":chan1:range?")
' query scope for vertical sensitivity
CALL ioenter(scope&, vrange!)
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR
LOCATE 20, 1
PRINT vrange! / 8; " V/div"
send (":chan1:offset?")
CALL ioenter(scope&, offset!)
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR
PRINT "Offset = "; offset!; " V"
send (":timebase:range?")
CALL ioenter(scope&, trange!)
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR
LOCATE 20, 41
PRINT trange! / 10; " S/div"
send (":timebase:delay?")
CALL ioenter(scope&, delay!)
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR
LOCATE 21, 41
PRINT " Delay = "; delay!; " S"
PRINT "press a key to see the integral of the signal plotted"
LINE INPUT dummy$
END SUB

SUB initialize
SHARED PCIB.ERR, PCIB.BASERR, NOERR
CLS
CALL ioreset(isc&)
' reset HP-IB interface
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR

```

```

timeout.val! = 10!
CALL iotimeout(isc&, timeout.val!) 'set timeout to 10 second
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR
CALL ioclear(isc&) 'clear interface & scope
IF PCIB.ERR NOERR THEN ERROR PCIB.BASERR
send ("*cls") 'clear status registers
send ("*rst") 'reset scope to default configuration
send (":autoscale") 'perform autoscale to find signal
send (":system:header off") 'for 54500: turn headers off

send (":acquire:type normal") 'acquisition type normal (default)
send (":acquire:points 500") 'set number of points to 500
send (":waveform:source channel1") 'source of data is channel 1 (default)
END SUB

```

```

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

```

END SUB

```

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

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

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

PRINT "Vpp = "; vpp.value! 'print Vpp
PRINT "frequency = "; frequency; " Hz" 'print frequency

```

```

PRINT
BEEP
INPUT "PRESS Enter TO CONTINUE:", a$ 'read keypad
END SUB

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

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

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

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

```

Microsoft QuickBASIC Initialize Program

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

```
DECLARE SUB measure ()
DECLARE SUB digitize ()
DECLARE SUB GPIBERR (MSG$)
DECLARE SUB initialize ()
DECLARE SUB send.to.scope (CMD$)
'Filename : \QB-NI\INIT.BAS
'
REM $INCLUDE: 'C:\GPIB-PC\QBDECL.BAS'

DIM SHARED scope%           'scope address variable
'
'MAIN PROGRAM
'
CALL initialize             'initialize interface & scope
CALL digitize              'have scope acquire a signal
CALL measure               'find Vpp and Frequency
CALL IBLOC(scope%)
  IF (IBSTA% AND EERR) THEN CALL GPIBERR("IBLOC ERROR")
CALL IBONL(scope%, 0)
  IF (IBSTA% AND EERR) THEN CALL GPIBERR("IBONL ERROR")
END

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

SUB GPIBERR (MSG$) STATIC
PRINT MSG$

PRINT "IBSTA=&H"; HEX$(IBSTA%); "< ";
IF IBSTA% AND EERR THEN PRINT " ERR";
IF IBSTA% AND TIMO THEN PRINT " TIMO";
IF IBSTA% AND EEND THEN PRINT " END";
IF IBSTA% AND SRQI THEN PRINT " SRQI";
IF IBSTA% AND RQS THEN PRINT " RQS";
IF IBSTA% AND CMPL THEN PRINT " CMPL";
IF IBSTA% AND LOK THEN PRINT " LOK";
```



```

send.to.scope (":acquire:points 500")      'set # of points to 500
send.to.scope (":acquire:type normal")    'normal acquisition mode
send.to.scope (":acquire:complete 100")   'completion criteria set
PRINT "scope is initialized. Press enter to continue"
LINE INPUT dummy$
END SUB

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

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

```

Microsoft QuickC Initialize Program

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

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

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

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

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

error_handler(int error, char *routine)
{
```

```

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

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

get_waveform()
{
    send (":digitize channel1");
}

measure()
{
    float freq,vpp;

    send (":measure:frequency?"); /* query frequency of input signal */

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

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

```

Microsoft QuickC Digitize Program

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

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

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

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

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

```

printf ("    h. calculate and display integral\n\n");
printf ("The sample program assumes the following address settings:\n");
printf ("    HP-IB interface select code = 7\n");
printf ("    scope address = 7\n");
printf ("Set these constants as needed for your system.\n\n");
printf ("PRESS Enter TO CONTINUE OR Ctrl C TO TERMINATE\n\n");
ch = getch();

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

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

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

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

```

```

    _clearscreen (_GCLEARSCREEN);          /* clear PC's screen */
}

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

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

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

get_waveform()
{
    int length;
    char msg[20];

    send (":waveform:format comp");      /* use the 8-bit byte format */
    send (":waveform:preamble?");        /* ask for info about data */
    length = 10;                          /* totally 10 parameters */
    error_handler(IOENTERA(SCOPE,preamble,&length), "ioentera");

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

save_waveform() { FILE *fp;

```

```

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

retrieve_waveform()
{
    FILE *fp;
    char ch;

printf("Waveform data has been saved to a file. The program will retrieve\n");
printf("the data, graph it, and perform integration. All subsequence operations\n");
printf ("are based on the saved informations. Press <ENTER>to continue.\n");
printf ("%c", ch = getchar());

    fp = fopen("wave.dat", "rb");          /* open the file */
    fread(preamble,sizeof(preamble[0]),10,fp); /* get the preambles */
    /* retrieve the waveform data, preamble[2] contains number of points */
    fread (waveform, sizeof(waveform[0]), (unsigned)preamble[2], fp);
    fclose(fp);          /* close the file */
}

graph_waveform()
{
    int i;
    char ch;
    _setvideomode(_ERESCOLOR);          /* set screen mode to EGA */
    _rectangle (_GBORDER, 0,0, 501,257); /* draw border */
    _setviewport(1,1,500,256);          /* set viewport */
    _setwindow (TRUE, 0,0,10,8);        /* set mapping co-ordinates */
    _setcolor (8);                      /* set color to dark gary */
    for (i=1;i;i++) {                   /* draw the grid */
        _moveto_w(i,0);
        _lineto_w(i,8);
    }
    for (i=1;i;i++) {
        _moveto_w(0,i);
        _lineto_w(10,i);
    }
    _setwindow(TRUE, 1,0,preamble[2],255); /* set mapping co-ordinates */
    _setcolor(10);                      /* set color to bright green */
    for (i=0;i<preamble[2];i++) {       /* draw the pixels */
        _setpixel_w (i,waveform[i]);
    }
    _settextwindow(20,1,24,80);         /* set text window */
    _settextcolor (15);                 /* set color to bright white */
    _settextposition(1,1);              /* position cursor */
    printf ("V/Div = %f V \n", 32*preamble[7]);
}

```



```

printf ("Offset= %11.3e V \n",
        (128 - preamble[9])*preamble[7]+preamble[8]);
_settextposition(1,41);
printf ("S/Div = %11.3e S \n", preamble[2]*preamble[4]/10);
_settextposition(2,41);
printf ("\nPRESS Enter to plot integral\n");
ch = getch();
}

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

  math[0] = 0;                /* starting from zero */
  for (i=0; i<preamble[2]; i++) { /* each elt. = last elt + curr wf data */
    math[i+1] = math[i] + (waveform[i]-preamble[9])*preamble[7] + preamble[8];
  }
  max = math[1];              /* find out the max & min of integrate */
  min = math[1];              /* in order to have the proper scale */
  for (i=1; i<=preamble[2]; i++) {
    if (math[i] > max) {max = math[i];}
    if (math[i] < min) {min = math[i];}
  }
  _setwindow(TRUE, 1, min, preamble[2], max); /* set up proper mapping */
  _setcolor(12);                /* use red color */
  for (i=1; i<=preamble[2]; i++) /* draw the integrate */
    _setpixel_w(i, math[i]);
  _settextposition (3,1);        /* print out information about integrate */
}

```

Microsoft QuickC Initialize Program

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

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

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

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

void main()
{ char ch;
  _clearscreen (_GCLEARSCREEN); /* clear PC's screen */
  printf ("This sample program will perform the following tasks:\n");
  printf (" a. initialize the HP-IB interface and scope\n");
  printf (" b. digitize the signal\n");
  printf (" c. do a Vpp and Frequency measurement on the signal\n");
  printf ("PRESS Enter TO CONTINUE OR Ctrl C TO TERMINATE\n\n");
  ch = getch();
  initialize(); /* initialize scope and interface */
  capture_waveform(); /* digitize signal */ measure();
/* measure Vpp and Freq */
  ibloc(scope);
  exit(0);
}

send (char *cmd)
{
/*this routine is added to made sending commands easier to read */
  ibwrt(scope,cmd,(long)strlen(cmd));
  if (ibsta & ERR)
  {
    gpiberr("ibwrt error");
    exit (1);
  }
}

initialize()
{
  int i;
```

```

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

send (":acquire:points 500"); /* get 500 pts on digitize */
send (":acquire:type normal"); /* normal acq mode */
send (":acquire:complete 100"); /* set completion criteria */
clearscreen(_GCLEARSCREEN); /* clear pc screen */
}

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

measure()
{
    char buffer[15];
    float result;
    char ch;
    send(":measure:frequency?"); /* query freq */
    ibrd(scope,buffer,15);
    if (ibsta & ERR)
    {
        gpiberr("ibrd error");
        exit(1);
    }
    result = (float)atof(buffer);
    printf("\nfrequency = %e Hz \n",result); /*print frequency*/
    send(":measure:vpp?"); /* query vpp */
    ibrd(scope,buffer,15);
    if (ibsta & ERR)
    {
        gpiberr("Ibrd error");
        exit(1);
    }
}

```

```

    }
    result = (float)atof(buffer);
    printf("Vpp = %e V \n",result);
}

gpiberr(char *msg) {
char ch;
/* This routine would notify you that an IB call failed.    */
printf ("%s\n", msg);
printf ( "ibsta=&H%x  ", ibsta);
if (ibsta & ERR ) printf (" ERR");
if (ibsta & TIMO) printf (" TIMO");
if (ibsta & END ) printf (" END");
if (ibsta & SRQI) printf (" SRQI");
if (ibsta & RQS ) printf (" RQS");
if (ibsta & CMPL) printf (" CMPL");
if (ibsta & LOK ) printf (" LOK");
if (ibsta & REM ) printf (" REM");
if (ibsta & CIC ) printf (" CIC");
if (ibsta & ATN ) printf (" ATN");
if (ibsta & TACS) printf (" TACS");
if (ibsta & LACS) printf (" LACS");
if (ibsta & DTAS) printf (" DTAS");
if (ibsta & DCAS) printf (" DCAS");
printf (" \n");
printf ("iberr= %d", iberr);
if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
if (iberr == ECIC) printf (" ECIC <Not CIC>\n");
if (iberr == ENOL) printf (" ENOL <No Listener>\n");
if (iberr == EADR) printf (" EADR <Address error>\n");
if (iberr == EARG) printf (" EARG <Invalid argument>\n");
if (iberr == ESAC) printf (" ESAC <Not Sys Ctrlr>\n");
if (iberr == EABO) printf (" EABO <Op. aborted>\n");
if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
if (iberr == EOIP) printf (" EOIP <Async I/O in prg>\n");
if (iberr == ECAP) printf (" ECAP <No capability>\n");
if (iberr == EFSO) printf (" EFSO File sys. error>\n");
if (iberr == EBUS) printf (" EBUS <Command error>\n");
if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");
printf ("ibcnt= %d\n", ibcnt1);
printf ("\n");
/* Call the ibonl function to disable the hardware
and software.    */
ibonl (scope,0);
printf("Press Enter to continue: ");
printf ("%c", ch = getchar()); /* wait for enter key */
}

```

Common Commands

Introduction

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

The following common commands are implemented in the HP 54505B, HP 54506B, HP 54510B, and HP 54512B:

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

Figure 6-1 lists the syntax diagrams for the common commands.

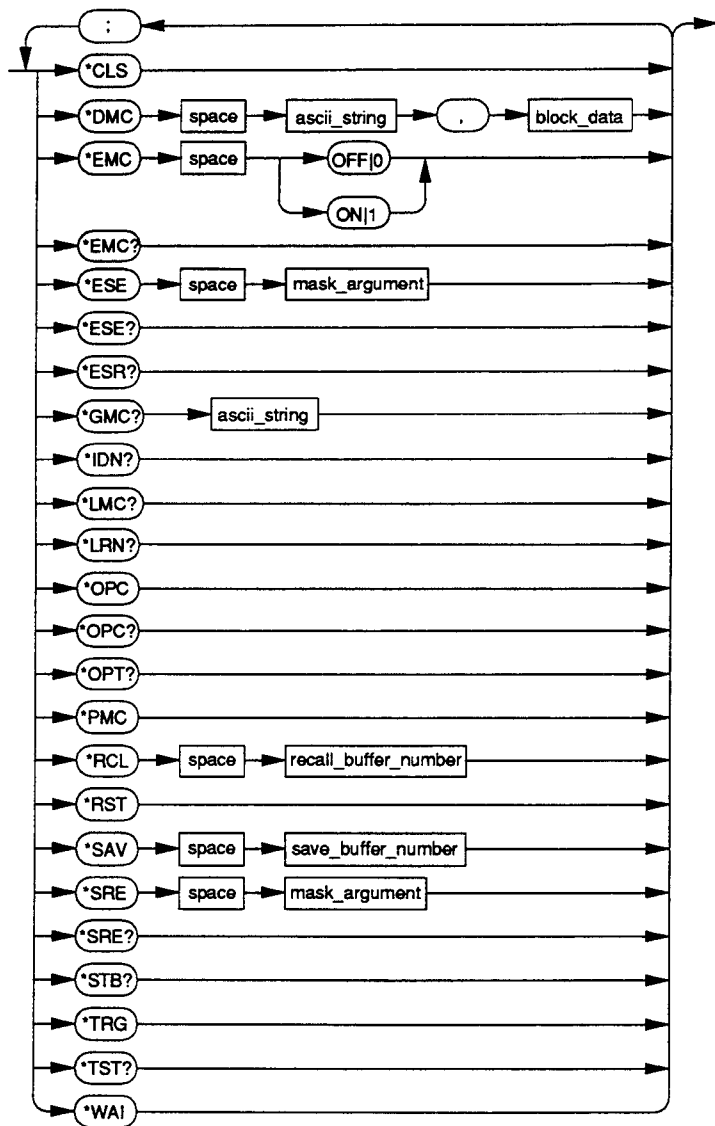


Figure 6-1. Common Commands Syntax Diagram

| | |
|-------------------------------|---|
| mask_argument = | An integer, 0 through 255. This number is the sum of all of the bits in the mask corresponding to conditions that are enabled. Refer to *ESE and *SRE commands for bit definitions in the enable registers. |
| recall_buffer_number = | An integer, 0 through 9. |
| save_buffer_number = | An integer, 1 through 9. |
| ascii_string = | A quoted string. |
| block_data = | block data in IEEE 488.2 # format. |

Figure 6-1. Common Commands Syntax Diagram (continued)

Common Commands

Common commands can be received and processed by the oscilloscope 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 remains in the selected subsystem. For example, if the program message ":ACQUIRE:TYPE AVERAGE; *CLS; COUNT 1024" is received by the instrument, the instrument sets the acquire type and count, then clears the status information without leaving the selected subsystem.

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

Note



Each of the status registers mentioned in this chapter has an enable (mask) register. By setting the bits in the enable register, you can select the status information you wish to use. For a complete discussion of how to read the status registers and how to use the status information available from this instrument refer to chapter 24, "Status Reporting."

*CLS

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

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

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

Command Syntax: *CLS

Example: OUTPUT 707;""CLS"



Refer to chapter 24, "Status Reporting" for a complete discussion of status.

DMC*(Define Macro)****command**

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

Command Syntax: ***DMC** <quoted acsii string>,<binary block data in # format>

Example: **OUTPUT 707;""*DMC 'SETUPTB',#232TIM:RANG 10NS;DELAY 0;REF CENTER"**

*EMC

***EMC** (Enable Macro) **command/query**

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

Command Syntax: *EMC {{ON | 1} | {OFF | 0}}

Example: OUTPUT 707;"*EMC 1"

Query Syntax: *EMC?

Returned Format: {1 | 0}<NL>

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

ESE*(Event Status Enable)****command/query**

The *ESE command sets the bits in the Standard Event Status Enable Register. The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register. A one in the Standard Event Status Enable Register enables the corresponding bit in the Standard Event Status Register. A zero disables the bit. Refer to table 6-1 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>

Where:

<mask> ::= 0 to 255

Example: OUTPUT 707;"*ESE 64"

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

*ESE

Query Syntax: *ESE?

Returned Format: <mask><NL>

Where:

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

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

Table 6-1. Standard Event Status Enable Register

| Event Status Enable Register (High - Enables the ERS 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 |



Refer to chapter 24, "Status Reporting" for a complete discussion of status.

***ESR**

(Event Status Register)

query

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

When you read the Event Status Register, the value returned is the total bit weights of all of the bits that are high at the time you read the byte. Table 6-2 shows each bit in the Event Status Register and its bit weight.



Reading the register clears the Standard Event Status Register.

Query Syntax: *ESR?

Returned Format: <status><NL>

Where:

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

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

GMC*(Get Macro Contents)****query**

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

Query Syntax: *GMC? "<Macro_name>

Returned Format: <block_data><NL>

Where:

<block_data> ::= binary block data in # format

Example: DIM Def\$[50]
OUTPUT 707;""*GMC? 'STORE'
ENTER 707;Def\$
PRINT Def\$

*IDN

*IDN

(Identification Number)

query

The *IDN query identifies the instrument type, serial number, and software version by returning the following string:

```
"HEWLETT-PACKARD,545XXB,<XXXAYYYYY>,<MMDD>"
```

Where:

XXB = 05B (HP 54505B), 06B (HP 54506B), 10B (HP 54510B), or 12B (HP 54512B)
<XXXAYYYYY> ::= the serial number of the instrument.

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

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

Query Syntax: *IDN?

Returned Format: HEWLETT-PACKARD,545XXB,XXXAYYYYY,MMDD<NL>

Example:

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

***LMC**

(Learn Macro)

query

The *LMC query returns the currently defined macro labels.

Query Syntax: *LMC?

Returned Format: <ascii_scring><NL>

Where:

<ascii_scring> ::= string list separated by commas

Example: DIM Labe1\$[50]
OUTPUT 707;"*LMC?"
ENTER 707;Labe1\$
PRINT Labe1\$

*LRN

*LRN

(Learn)

query

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

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



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

Query Syntax: *LRN?

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

Where:

<setup> ::= #800001703<learn string><NL>
<learn string> ::= 1703 data bytes in length.

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

OPC*(Operation Complete)****command/query**

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

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

Command Syntax: *OPC

Example: OUTPUT 707;"*OPC"

Query Syntax: *OPC?

Returned Format: 1<NL>

Example: OUTPUT 707;" :AUTOSCALE;*OPC?"
ENTER 707;0p\$

*OPT

***OPT**

(Option)

query

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

Query Syntax: *OPT?

Returned Format: 0<NL>

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

PMC*(Purge Macro)****command**

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

Command Syntax: *PMC

Where:

Example: OUTPUT 707;"*PMC"

*RCL

***RCL**

(Recall)

command

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



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

Command Syntax: *RCL {0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

Example: OUTPUT 707;*"RCL 3"

***RST**

(Reset)

command

The *RST command places the instrument in a known state. Refer to table 6-3 for the reset conditions.

Command Syntax: *RST

Example: OUTPUT 707;"*RST"

Table 6-3. Reset Conditions

| | |
|------------------------------|--------------|
| Timebase Menu | |
| time/division | 100 μ s |
| delay | 0.00 s |
| reference | cntr |
| sample rate | realtime |
| sequential | off |
| Channel Menu | |
| Channel 1 | on |
| Channel 2 | off |
| Channel 3 (HP 54506B/54512B) | off |
| Channel 4 (HP 54506B/54512B) | off |
| volts/division | 500 mV |
| offset | 0.00 |
| coupling | dc |
| input resistance | 1 M Ω |
| probe attenuation | 1.000:1 |

Table 6-3. Reset Conditions (continued)

| | |
|--|------------|
| Trigger Menu | |
| triggering | auto |
| mode | edge |
| source | Channel 1 |
| level | 0.0 V |
| slope | positive |
| noise reject | off |
| holdoff | 40 ns |
| Display Menu | |
| mode | realtime |
| persistence | minimum |
| # of screens | 1 |
| off/frame/axes/grid | axes |
| connect dots | off |
| $\Delta t/\Delta V$ Menu | |
| Markers | off |
| Waveform Math Menu | |
| f1 | off |
| f2 | off |
| f3 | off |
| f4 | off |
| display | off |
| chan/mem | chan 1 |
| operator | + |
| chan/mem | chan 1 |
| function sensitivity | 1.00 V/div |
| function offset | 0.0 V |

Table 6-3. Reset Conditions (continued)

| | |
|--------------------------------------|------------|
| Waveform Save Menu | |
| waveform/pixel/multiple/mask | waveform |
| nonvolatile | m1 |
| display | off |
| source | chan 1 |
| protect | off |
| Define Meas Menu | |
| measure/define/limit/compare | measure |
| continuous | on |
| statistics | off |
| rms | ac |
| standard/user defined | standard |
| Utility Menu (System Submenu) | |
| clicker | on |
| AC BNC | probe comp |
| interpolation | on |

***SAV**

***SAV**

(Save)

command

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

Command Syntax: *SAV {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9}

Example: OUTPUT 707;""SAV 3"

***SRE**

(Service Request Enable)

command/query

The *SRE command sets the bits in the Service Request Enable Register. The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A one in the Service Request Enable Register enables the corresponding bit in the Status Byte Register. A zero disables the bit. Table 6-4 lists the bits in the Service Request Enable Register and what they mask.

The *SRE query returns the current value.

Command Syntax: *SRE <mask>

Where:

<mask> ::= 0 to 255

Example: OUTPUT 707;"*SRE 16"



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

Query Syntax: *SRE?

Returned Format: <mask><NL>

Where:

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

Example: OUTPUT 707;"*SRE?"
ENTER 707;Value
PRINT Value

Table 6-4. Service Request Enable Register

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

STB*(Status Byte)****query**

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



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

Query Syntax: *STB?

Returned Format: <value><NL>

Where:

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

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

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

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

Command Syntax: *TRG

Example: OUTPUT 707;"*TRG"

*TST

***TST**

(Test)

query

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



Disconnect all front-panel inputs before sending this command.

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

If a test fails, refer to the troubleshooting section of the HP 54505B/54506B and HP 54510B/54512B Service Manual.

Query Syntax: *TST?

Returned Format: <result><NL>

Where:

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

Where:

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

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

***WAI**

(Wait)

command

The ***WAI** command has **no function** in the HP 54505B, HP 54506B, HP 54510B, or HP 54512B, but is parsed for compatibility with other instruments.

Command Syntax: *WAI

Example: OUTPUT 707;""*WAI"

Root Level Commands

Introduction

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

The following Root Level commands are implemented in the HP 54505B, HP 54506B, HP 54510B, and HP 54512B:

- AUToscale
- BEEPer
- BLANK
- BNC
- DIGitize
- ERASe
- LER
- LTER
- MENU
- MERGe
- PLOT
- PRINt
- RUN
- SERial
- STATus
- STOP
- STORe
- TER
- VIEW

Figure 7-1 lists the syntax diagrams for the Root Level commands.

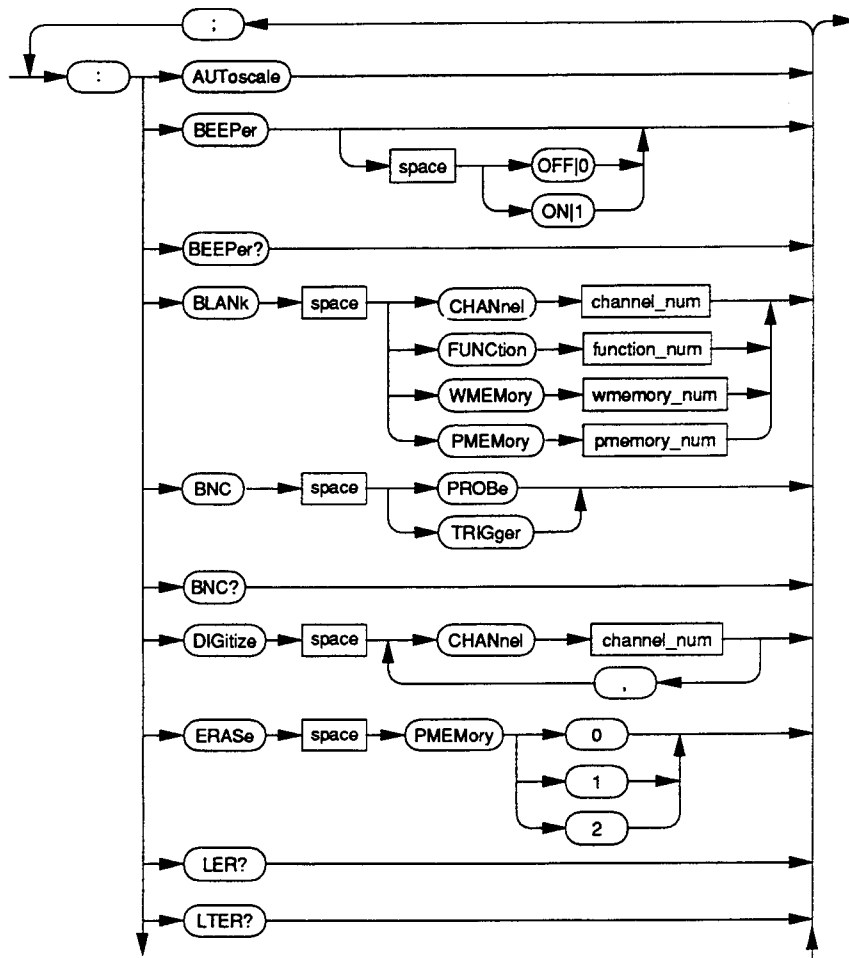


Figure 7-1. Root Level Commands Syntax Diagram

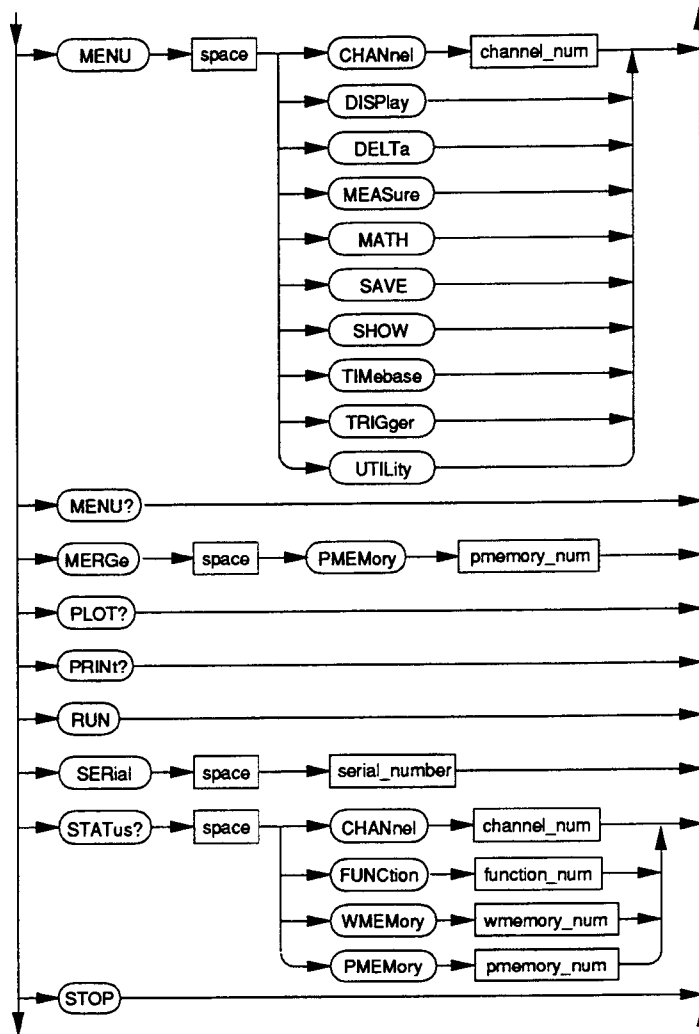
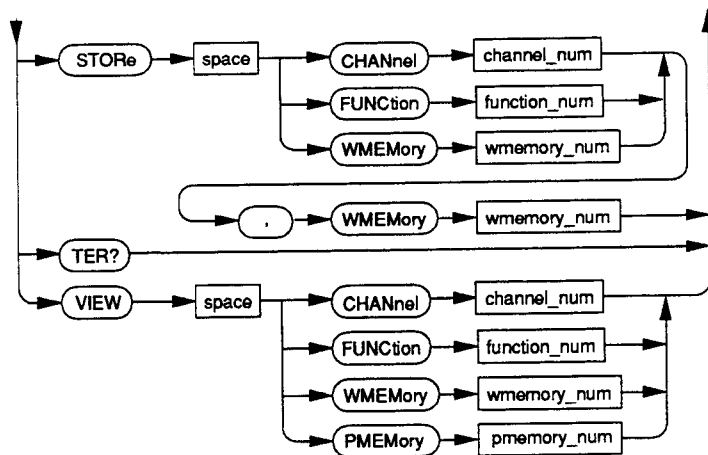


Figure 7-1. Root Level Commands Syntax Diagram (continued)



- channel_num =** an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B).
- function_num =** an integer, 1 through 4.
- wmemory_num =** an integer, 1 through 4.
- pmemory_num =** an integer, 1 or 2.
- serial_number =** a 10 character quoted string.

Figure 7-1. Root Level Commands Syntax Diagram (continued)

AUToscale**command**

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

- The vertical sensitivity.
- The vertical offset.
- The trigger to edge mode.
- The trigger level, holdoff, and slope.
- The sweep speed of the displayed channel.
- The time base reference to center.
- The time base delay to 0.0 s.

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

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

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

Command Syntax: :AUToscale

Example: OUTPUT 707;":AUTOSCALE"

BEEPer

BEEPer

command/query

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

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

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

Example: OUTPUT 707;":BEEPER 1"

Query Syntax: :BEEPer?

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

Example:
DIM Click\$[50]
OUTPUT 707;":BEEPER?"
ENTER 707;Click\$
PRINT Click\$

BLANK**command**

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

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

Command Syntax: :BLANK <display>

Where:

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

Example: OUTPUT 707;":BLANK CHANNEL1"

Note 

CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

BNC

BNC

command/query

The BNC command sets the output mode of the rear-panel BNC to PROBE or TRIGGER. The PROBE mode outputs a square wave signal. The TRIGGER mode outputs a rising edge when an internal trigger occurs.

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

Command Syntax: :BNC {PROBe | TRIGger}

Example: OUTPUT 707;":BNC PROBE"

Query Syntax: :BNC?

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

Example: DIM Mode\${50}
OUTPUT 707;":BNC?"
ENTER 707;Mode\$
PRINT Mode\$

DIGitize**command**

The DIGITIZE command is used to acquire waveform data for transfer over the HP-IB. Sending the command causes an acquisition to take place on the specified channels with the resulting data being placed in the channel buffer. Sending the DIGITIZE command also turns off any unused channels.

The sources for the :DIGITIZE command are channels 1 and 2 (HP 54505B/54510B) or channels 1 through 4 (HP 54506B/54512B).

During a digitize operation, connect the dots, infinite persistence, and variable persistence are disabled so that the screen more accurately reflects the data transferred. Disabling these functions also simplifies and speeds up the digitize operation.

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

The ACQUIRE subsystem commands are used to set up conditions such as TYPE, number of POINTS, and COUNT for the next DIGITIZE command. See the ACQUIRE subsystem for a description of these commands. To determine the actual number of points that are acquired and how the data is transferred, refer to the WAVEFORM subsystem commands.

If you only want data, you can improve the speed by turning the screen off with the command :DISPLAY:SCREEN OFF prior to sending the DIGITIZE command. In this case, nothing is plotted on the screen.

For more information on the DIGITIZE command refer to the section on the DIGITIZE command in chapter 2, "Programming an Instrument."

DIGitize

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

Where:

<N> ::= 1 through 4.

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



CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

ERASe**command**

The ERASE command erases a specified pixel memory.

Erasing pixel memory 0 is a special case, which is the same as pressing the CLEAR DISPLAY key on the front panel. If the scope is running and being triggered and ERASE PMEMORY0 is executed, the instrument momentarily stops acquiring data, clears the screen, then restarts the data acquisition.

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



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

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

Example: OUTPUT 707;":ERASE PMEMORY1"

LER

LER

(Local Event Register)

query

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

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

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

Query Syntax: :LER?

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

Example:
DIM Event\$[50]
OUTPUT 707;" :LER?"
ENTER 707;Event\$
PRINT Event\$

LTER**(Limit Test Event Register)****query**

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

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

Query Syntax: :LTER?

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

Example:
DIM Lmt\$[50]
OUTPUT 707;":LTER?"
ENTER 707;Lmt\$
PRINT Lmt\$

MENU

MENU

command/query

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

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

Command Syntax: :MENU <name>

Where:

<name> ::= {CHANne1 [1|2|3|4] | TIMEbase | CHANne1 | TRIGger | DISPlay | DELTa | MATH | SAVE | MEASure | UTILity | SHOW}

Example: OUTPUT 707;":MENU DISPLAY"

Query Syntax: :MENU?

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

Where:

<name> ::= {CHANne1 [1|2|3|4] | TIMEbase | CHANne1 | TRIGger | DISPlay | DELTa | MATH | SAVE | MEASure | UTILity | SHOW}

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

Note 

CHANne13 and CHANne14 are only available when the oscilloscope is an HP 54506B/54512B.

MERGe**command**

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

The function of this command is similar to the function of the "add to memory" key in the pixel menu of the Waveform Save menu.

Command Syntax: :MERGe {PMEMory1 | PMEMory2}

Example: OUTPUT 707;":MERGE PMEMORY2"

PLOT

PLOT

query

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

Query Syntax: :PLOT?

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

PRINT

query

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

Query Syntax: :PRINT?

```

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

```

RUN

RUN

command

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

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

If the time base mode is set to AUTO or TRIGGERED, the RUN command enables the trigger repeatedly and displays the data it acquires continuously on the screen. This is the same thing that happens when the front panel RUN key is pressed. For a description of the various modes, see the :TIMEBASE:MODE command in chapter 16, "Timebase Subsystem."

Command Syntax: :RUN

Example: OUTPUT 707;":RUN"

SERIAL**(Serial Number)****command**

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

A serial number corresponding to the serial number of the board in the HP 54505B, HP 54506B, HP 54510B, or HP 54512B is loaded at the factory. Do not use this command unless you need to serialize the instrument for a different application.

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

Command Syntax: :SERial <string>

Where:

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

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

STATus

STATus

query

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

Query Syntax: :STATus? <display>

Where:

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

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

Example:
DIM Status\$[50]
OUTPUT 707;":STATus? CHANNEL1"
ENTER 707;Status\$
PRINT Status\$

Note 

CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

STOP**command**

The STOP command stops the data acquisition.

The RUN command must be executed to restart the acquisition.

Command Syntax: :STOP

Example: OUTPUT 707;":STOP"

STORE

STORE

command

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

- The first parameter is the source of the waveform, which can be specified as any channel, function, or waveform memory.
- The second parameter is the destination of the waveform, which can only be waveform memory 1 through 4.

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

where:

<source> ::= {CHANne1{1|2|3| 4} | FUNCtion{1|2|3|4} | WMEMory{1|2|3|4}}
<destination> ::= WMEMory {1|2|3|4}

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

Note 

CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

TER

(Trigger Event Register)

query

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

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

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

Query Syntax: :TER?

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

Example: DIM Trg_event\$ [50]
OUTPUT 707;":TER?"
ENTER 707;Trg_event\$
PRINT Trg_event\$

VIEW

VIEW

command

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

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

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

Command Syntax: `:VIEW <display>`

Where:

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

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



CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

System Subsystem

Introduction

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

The System subsystem contains the following commands:

- DATE
- DSP
- ERRor
- HEADer
- KEY
- LONGform
- PIMacro
- SETup
- TIME

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

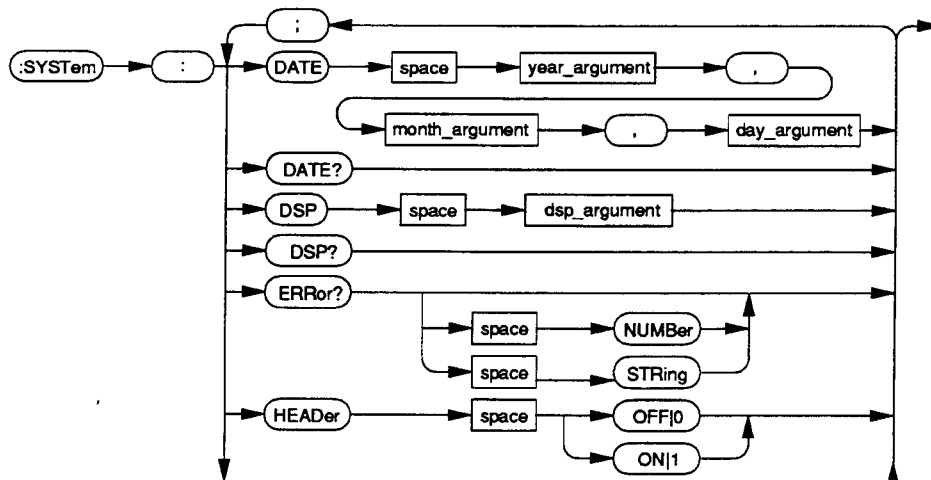
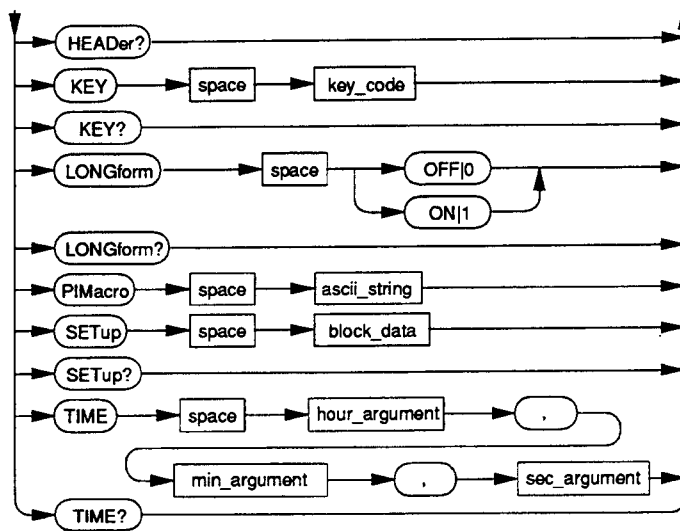


Figure 8-1. System Subsystem Commands Syntax Diagram



- dsp_argument =** any quoted string.
- key_code =** an integer, 1 through 44.
- ascii_string =** any quoted string.
- block_data =** block data in IEEE 488.2 # format.
- year_argument =** an integer, 1990 through 2059.
- month_argument =** an integer, 1 through 12.
- day_argument =** an integer, 1 through 31.
- hour_argument =** an integer, 0 through 23.
- minute_argument =** an integer, 1 through 59.
- second_argument =** an integer, 1 through 59.

Figure 8-1. System Subsystem Commands Syntax Diagram (continued)

DATE

command/query

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

The `SYSTEM:DATE` query returns the current date.

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

Where:

`<year>` ::=1990 to 2059

`<month>` ::=1 to 12

`<day>` ::=1 to 31

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

Query Syntax: `:SYSTem:DATE?`

Returned Format: `[:SYSTem:DATE] "DDMMYYYY"<NL>`

Where:

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

`MMM` ::=three digit alphabetic month

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

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

DSP

DSP

command/query

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

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

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

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

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

Query Syntax: :SYSTem:DSP?

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

Where:

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

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

ERRor

query

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

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

See table 8-1 for the error numbers and descriptions.

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

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

Where:

<error> ::= an integer error code.

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

Example:

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

Table 8-1. Error Messages

| Error Number | Description |
|---------------------|---------------------------------|
| 11 | Questionable horizontal scaling |
| 12 | Edges required not found |
| 13 | Not a 545XXB command |
| 70 | RAM write protected |
| -100 | Command error (unknown command) |
| -101 | Invalid character |
| -102 | Syntax error |
| -103 | Invalid separator |
| -104 | Data type error |
| -105 | GET not allowed |
| -108 | Parameter not allowed |
| -109 | Missing parameter |
| -112 | Program mnemonic too long |
| -113 | Undefined header |
| -121 | Invalid character in number |
| -123 | Numeric overflow |
| -124 | Too many digits |
| -128 | Numeric data not allowed |
| -130 | Suffix error |
| -131 | Invalid suffix |
| -138 | Suffix not allowed |
| -140 | Character data error |
| -141 | Invalid character data |
| -144 | Character data too long |
| -148 | Character data not allowed |
| -150 | String data error |
| -151 | Invalid string data |
| -158 | String data not allowed |

Table 8-1. Error Messages (Continued)

| Error Number | Description |
|--------------|--|
| -160 | Block data error |
| -161 | Invalid block data |
| -168 | Block data not allowed |
| -170 | Expression error |
| -171 | Invalid expression |
| -178 | Expression data not allowed |
| -200 | Execution error |
| -211 | Trigger ignored |
| -221 | Settings conflict |
| -222 | Data out of range |
| -223 | Too much data |
| -310 | System error |
| -350 | Too many errors |
| -400 | Query error |
| -410 | Query INTERRUPTED |
| -420 | Query UNTERMINATED |
| -430 | Query DEADLOCKED |
| -440 | Query UNTERMINATED after indefinite response |

HEADer

HEADer

command/query

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

The `HEADER` query returns the state of the `HEADER` command.

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

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

Query Syntax: `:SYSTem:HEADer?`

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

Where:

1 ::= ON
0 ::= OFF

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

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

With headers set to `ON`; long form `ON`:

```
:CHANNEL1:RANGE 6.40000E-01
```

With headers set to `ON`; long form `OFF`:

```
:CHAN1:RANG 6.40000E-01
```

With headers set to `OFF`:

```
6.40000E-01
```



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

KEY

command/query

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

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

Refer to table 8-2 for key codes.

Note

The :SYSTEM:KEY 39 command will not return the instrument to the "Local" mode when the instrument is in the "Local Lockout" mode.

Command Syntax: :SYSTem:KEY <keycode>

Where:

<keycode> ::= 1 to 44

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

Query Syntax: :SYSTem:KEY?

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

Where:

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

Example: DIM Input\$[50]
OUTPUT 707;":SYSTEM:KEY?"
ENTER 707;Input\$
PRINT Input\$

KEY

Table 8-2. Oscilloscope Front-Panel Key Codes

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

Note



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

LONGform

command/query

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

The `LONGFORM` query returns the state of the `LONGFORM` command.

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

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

Query Syntax: `:SYSTem:LONGform?`

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

Where:

1 ::= ON
0 ::= OFF

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

PIMacro

PIMacro

command

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

Command Syntax: `:SYSTem:PIMacro <quoted ASCII string>`

Where:

`<quoted ASCII string> ::= name of the defined macro`

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

SETup**command/query**

The :SYSTEM:SETUP command sets the oscilloscope as defined by the data in the setup (learn) string sent from the controller. The setup string contains 1703 bytes of setup data. The 1703 bytes do not include the header or "#800001703."

The setup string does not change the HP-IB mode, HP-IB address, fine key setting, waveform format, or waveform source.

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

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

Command Syntax: :SYSTem:SETup <setup>

Where:

<setup> ::= #800001703<setup data string>

SETup

Query Syntax: :SYSTem:SETup?

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

Where:

<setup> ::= #800001703<setup data string>

Example:

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



The logical order for this instruction is to send the query, 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 oscilloscope.

TIME

command/query

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

The `SYSTEM:TIME` query returns the current time.

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

Where:

`<hour>` ::=0 to 23

`<minute>` ::=0 to 59

`<second>` ::=0 to 59

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

Query Syntax: `:SYSTem:TIME?`

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

Where:

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

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

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

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

Acquire Subsystem

Introduction

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

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

This subsystem contains the following commands:

- COMPLETE
- COUNT
- POINTS
- TYPE

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

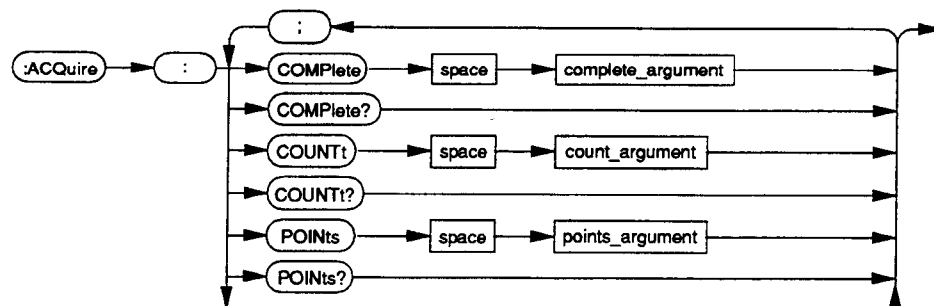
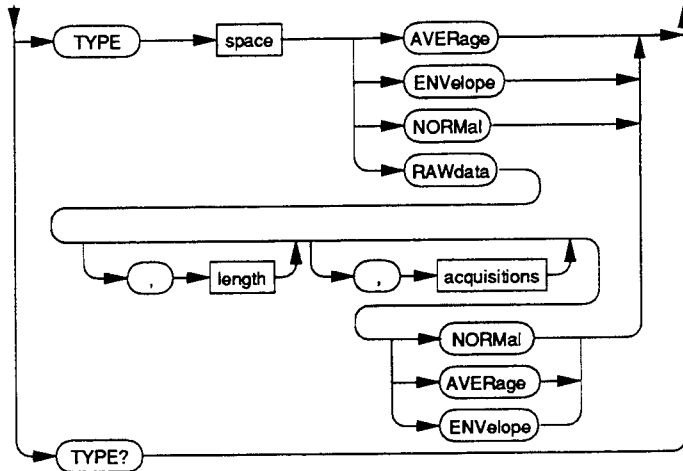


Figure 9-1. Acquire Subsystem Commands Syntax Diagram



- acquisitions =** dependent on the length of the acquisitions and the buffer size.
- complete_argument =** an integer, 0 through 100.
- count_argument =** an integer, 1 to 2048. It specifies the number of values to average for each time point when in the averaged mode or number of hits per time bucket in envelope mode.
- length =** an integer, 4 to 8000.
- points_argument =** 500 in the repetitive mode.
500 or 8000 in the realtime mode.

Figure 9-1. Acquire Subsystem Commands Syntax Diagram (continue)

Acquire Type and Count

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

(Normal) Persistence Mode

The :ACQUIRE:TYPE NORMAL command sets the oscilloscope to the variable persistence mode when the instrument is operating in the repetitive time base mode. You can activate the same mode from the front panel by selecting the Display menu, then setting the display mode to Normal. The persistence time is set over the HP-IB in the DISPLAY subsystem using the :DISPLAY:PERSISTENCE command.

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

Averaging Mode

The :ACQUIRE:TYPE AVERAGE command sets the oscilloscope to the Averaging mode. The averaging mode is available only when the :TIMEBASE:SAMPLE is set to REPETITIVE.

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

Envelope Mode The `:ACQUIRE:TYPE ENVELOPE` command sets the oscilloscope to the Envelope mode. The envelope mode is only available when the `:TIMEBASE:SAMPLE` is set to `REPETITIVE`. In envelope mode, the `ACQUIRE:COUNT` command specifies how many times each time bucket is hit.

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

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

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

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

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

The command `:WAVEFORM:FORMAT` has no effect in the rawdata mode. Data is always transferred in the `WORD` format.

The Rawdata mode is exited using `SEQUENTIAL[:DISPLAY] OFF`.



It is highly recommended that the SEQuential Subsystem (see Chapter 22) be used when using sequential-shot mode (Rawdata mode). Rawdata commands are included only for compatibility between oscilloscopes.

COMPLete

command/query

The :ACQUIRE:COMPLETE command specifies the minimum completion criteria for an acquisition. The parameter determines what percentage of the time buckets need to be "full" before an acquisition is considered complete. If you are in the NORMAL mode, the instrument only needs one data point per time bucket for that time bucket to be considered full. In order for the time bucket to be considered full in the AVERAGE or ENVELOPE mode, a specified number of data points (COUNT) must be acquired.

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



The :ACQUIRE:COMPLETE command specifies the minimum completion criteria for an acquisition. In the realtime mode, and in the repetitive mode at sweep speeds slower than 50 ns per division, 100% of the time buckets are always filled in a single acquisition. This is due to the fast sample rate of the oscilloscope and does not vary with the value specified by the :ACQUIRE:COMPLETE command.

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

COMPLete

Command Syntax: :ACQuire:COMPLete <comp>

Where:

<comp> ::= 0 to 100 percent

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

Query Syntax: :ACQuire:COMPLete?

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

Where:

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

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

COUNT**command/query**

In the average mode, the :ACQUIRE:COUNT command specifies the number of values to be averaged for each time bucket before the acquisition is considered complete for that time bucket. The :ACQUIRE:COUNT command is only valid in the repetitive mode.

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

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

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

The COUNT query returns the currently selected count value.

Command Syntax: :ACquire:COUNT <count>

Where:

<count> ::= 1 to 2048 (depending on the acquisition type)

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

Query Syntax: :ACquire:COUNT?

Returned Format: [:ACquire:COUNT] <count><NL>

Where:

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

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

POINTS

POINTS

command/query

The :ACQUIRE:POINTS command specifies the number of time buckets for each acquisition record. When operating in the repetitive mode, the legal setting is 500. When operating in the real-time mode, the legal settings are 500 and 8000.

If a value is sent in the repetitive mode that is not a legal value, the value is set to 500. If a value is sent in the real-time mode that is not a legal value, the value is set according to one of the following conditions:

- If the value is less than or equal to 1023, the value is set to 500.
- If the value is greater than or equal to 1024, the value is set to 8000.

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



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

Command Syntax: :ACquire:POINTs <points_argument>

Where:

<points_argument> ::= 500 in the repetitive mode.
500 or 8000 in the real-time mode.

Example: OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE" !select sample mode
OUTPUT 707;":ACQ:POINTS 500"

Query Syntax: :ACquire:POINTs?

Returned Format: [:ACquire:POINTs] <points_argument><NL>

Where:

<points_argument> ::= 500 in the repetitive mode.
500 or 8000 in the real-time mode.

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

TYPE

TYPE

command/query

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

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

Note 

To get a correct PREAMBLE with the corresponding 8000 point realtime acquisition, you must send the DIGITIZE command or STOP the acquisition before sending the WAVEFORM:DATA query.

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

Command Syntax: :ACquire:TYPE {NORMa1 | AVERAge | ENVelope |
RAWData[[[,<length>][, <acquisitions>][, {NORMa1 | AVERAge | Envelope}}]}

Where:

<length> ::= 4 to 8000

<acquisitions> ::= dependent on length of acquisitions and buffer size as shown on previous page.

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

Query Syntax: :ACquire:TYPE?

Returned Format: [:ACquire:TYPE] {NORMa1 | AVERAge | ENVelope |
RAWData,<length>,<acquisitions><NL>

Where:

<length> ::= 4 to 8000 (integer - NR1 format).

<acquisitions> ::= dependent on length of acquisitions and buffer size as shown on previous page.

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

Calibrate Subsystem

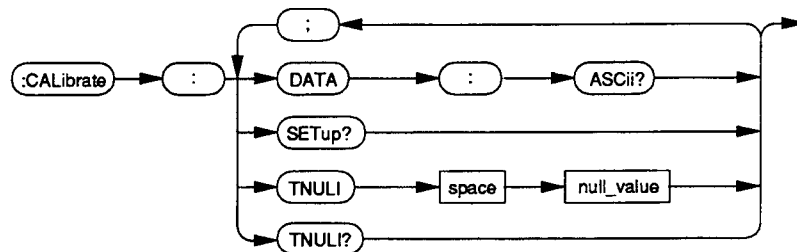
Introduction

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



Note

The time null is set in the Probe Cal menu of the Utility menus. For more information refer to the *HP 54505B/54510B and HP 54506B/54512B Front-Panel Reference*.



null_value =

channel 1 to channel 2 (HP 54505B/54510B) or channel 1 to channel 2 through 4 (HP 54506B/54512B) skew.

Figure 10-1. Calibrate Subsystem Commands Syntax Diagram

DATA:AScii

DATA:AScii

query

The :CALIBRATE:DATA:AScii query returns the instrument's calibration data.

Query Syntax: :CALibrate:DATA:AScii?

Returned Format: [:CALibrate:DATA:AScii] <data>,<data>,...<NL>

Where:

<data> ::= calibration data

Example: DIM Data\$[20000]
OUTPUT 707;" :CAL:DATA:AScii?"
ENTER 707;Data\$
PRINT Data\$

SETup**query**

The :CALIBRATE:SETUP query returns the current settings for the Calibrate Subsystem commands.

Query Syntax: :CALibrate:SETup?

Returned Format: :CALibrate:TNUL<null_value2>,<null_value3>,<null_value4><NL>

Example: DIM Stp\$ [100]
OUTPUT 707;":CAL:SET?"
ENTER 707;Stp\$
PRINT Stp\$

Note 

<null_value3> and <null_value4> returned only when the oscilloscope is an HP 54506B/54512B.

TNULI

TNULI

command/query

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

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

Command Syntax: `:CALibrate:TNULL <null_value2>,<null_value3>,<null_value4>`

Where:

`<null_value2>` ::= channel 1 to channel 2 skew (ALL)
`<null_value3>` ::= channel 1 to channel 3 skew (HP 54506B/54512B only)
`<null_value4>` ::= channel 1 to channel 4 skew (HP 54506B/54512B only)

Example: `OUTPUT 707;":CAL:TNULL 5NS`


Query Syntax: `:CALibrate:TNULL?`

Returned Format: `[:CALibrate:TNULL] <null_value2>,<null_value3>,<null_value4><NL>`

Where:

`<null_value2>` ::= channel 1 to channel 2 skew (exponential - NR3 format)
`<null_value3>` ::= channel 1 to channel 3 skew (exponential - NR3 format)
`<null_value4>` ::= channel 1 to channel 4 skew (exponential - NR3 format)

Example: `DIM N11$[50]
OUTPUT 707;":CALIBRATE:TNULL?"
ENTER 707;N11$
PRINT N11$`

Note  `<null_value3>` and `<null_value4>` applicable only when the oscilloscope is an HP 54506B/54512B.

Channel Subsystem

Introduction

The CHANNEL subsystem commands control the channel display and vertical or Y-axis of the oscilloscope. Each channel is independently programmable for all offset, probe, coupling, and range functions. The channel number specified in the command selects the channel that is affected by the command.

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

The Channel subsystem contains the following commands:

- COUPling
- DISPlay
- ECL
- HFReject
- LFReject
- OFFSet
- PROBe
- RANGe
- SETup
- TTL

Figure 11-1 lists the syntax diagrams for the Channel subsystem commands.

Note 

Commands for channels 3 and 4 are only available when the oscilloscope is an HP 54506B/54512B.

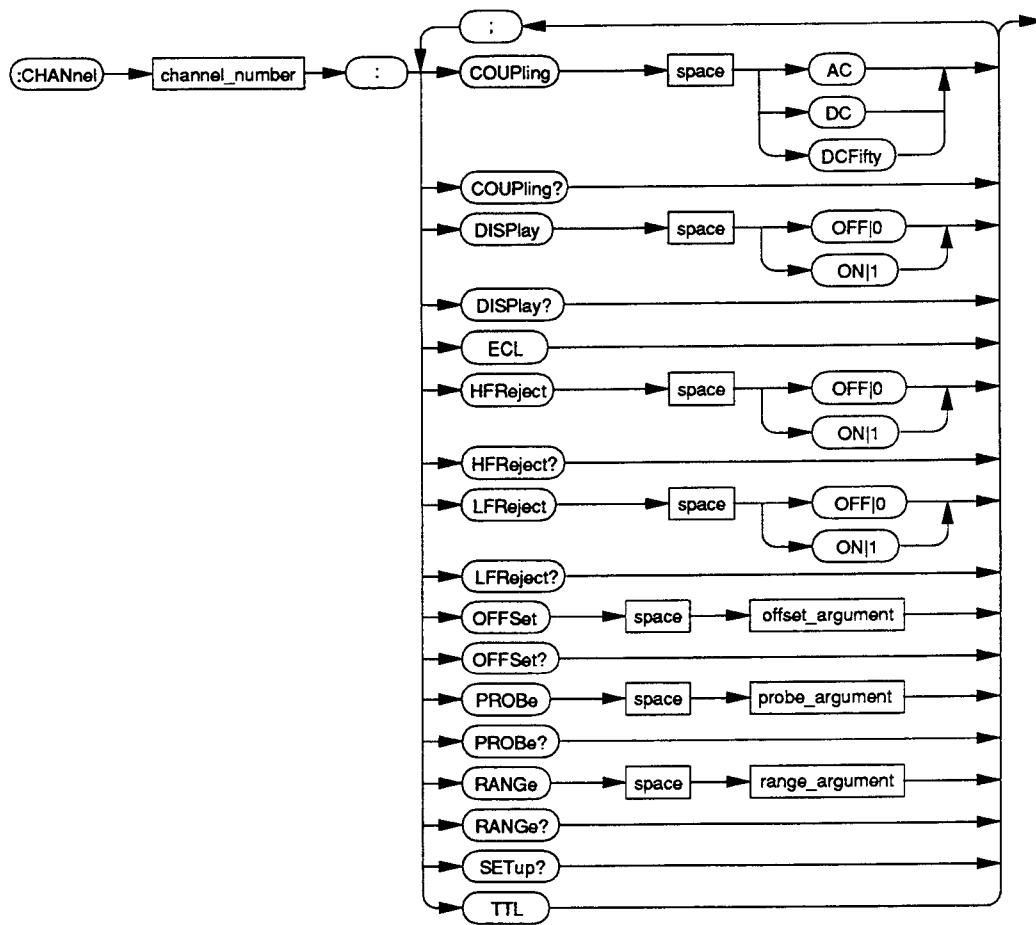


Figure 11-1. Channel Subsystem Commands Syntax Diagram

| | |
|--------------------------|--|
| channel_number = | an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B). |
| offset_argument = | a real number defining the voltage at the center of the display range. |
| probe_argument = | a real number from 0.9 to 1000.0 specifying the probe attenuation with respect to 1. |
| range_argument = | a real number specifying the size of the acquisition window in volts. |

Figure 11-1. Channel Subsystem Commands Syntax Diagram (Continued)

COUPLing

COUPLing

command/query

The `:CHANNEL<N>:COUPLING` command selects the input coupling for the specified channel. The coupling for each channel can be set to AC, DC, or DCFIFTY. DCFIFTY places an internal 50 Ω load on the input.

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

Command Syntax: `:CHANnel<N>:COUPLing {AC | DC | DCFifty}`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

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

Query Syntax: `:CHANnel<N>:COUPLing?`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

Returned Format: `[[:CHANnel<N>:COUPLing] {AC | DC | DCFifty}<NL>`

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

[DISPlay]**command/query**

The **:CHANNEL <N> [:DISPlay]** command controls the individual channel displays. On starts displaying, and off stops displaying the channel selected.

The **DISPlay** query returns the current setting of the command.

Note 

Because **[:DISPlay]** is an implied command, only the parameters need be sent to turn a selected channel on or off. For example, to turn channel 1 display on send either **":CHAN1:DISP ON"** or **":CHAN1 ON"**.

Command Syntax: `:CHANne1<N>[:DISPlay] {{ON | 1} | {OFF | 0}}`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

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

Query Syntax: `:CHANne1<N>[:DISPlay]?`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

Returned Format: `[:CHANne1<N>[:DISPlay]] {1 | 0}<NL>`

Example:
`DIM Disp$[50]`
`OUTPUT 707;":CHAN1?"`
`ENTER 707;Disp$`
`PRINT Disp$`

ECL

ECL

command

The `:CHANNEL <N>:ECL` command sets the vertical range, offset, channel coupling, and trigger level of the selected channel for optimum viewing of ECL signals. The channel offset and trigger level are set to -1.3 volts and the range is set to 1.6 volts full scale. Channel coupling is set to DC.

There is no query form of this command.

Command Syntax: `:CHANne1<N>:ECL`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)

Example: `OUTPUT 707;":CHAN1:ECL"`

HFReject
command/query

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

The HFREJECT query returns the current setting of the command.

Command Syntax: `:CHANne1<N>:HFReject {{ON | 1} | {OFF | 0}}`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

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

Query Syntax: `:CHANne1<N>:HFReject?`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

Returned Format: `[:CHANne1<N>:HFReject] {1 | 0}<NL>`

Example:
`DIM HF$[50]`
`OUTPUT 707;":CHAN1:HFR?"`
`ENTER 707;HF$`
`PRINT HF$`

LFReject

LFReject

command/query

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

The `LFREJECT` query returns the current setting of the command.

Command Syntax: `:CHANne1<N>:LFReject {{ON | 1} | {OFF | 0}}`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

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

Query Syntax: `:CHANne1<N>:LFReject?`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

Returned Format: `[[:CHANne1<N>:LFReject] {1 | 0}<NL>`

Example: `DIM Lf$[50]`
`OUTPUT 707;":CHAN1:LFREJECT?"`
`ENTER 707;Lf$`
`PRINT Lf$`

OFFSet

command/query

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

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

Command Syntax: `:CHANnel<N>:OFFSet <va lue>`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
`<value>` ::= offset value

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

Query Syntax: `:CHANnel<N>:OFFSet?`

Returned Format: `[:CHANnel<N>:OFFSet] <va lue><NL>`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
`<value>` ::= offset value in volts (exponential - NR3 format)

Example: `OUTPUT 707;":SYSTEM:HEADERS OFF"` !turn headers off
`OUTPUT 707;":CHANNEL2:OFFSET?"`
`ENTER 707;Offset`
`PRINT Offset`

PROBe

PROBe

command/query

The `:CHANNEL<N>:PROBE` command specifies the probe attenuation factor for the selected channel. The range of the probe attenuation factor is from 0.9 to 1000.0. This command does not change the actual input sensitivity of the oscilloscope. It changes the reference constants for scaling the display factors, for making automatic measurements, for setting trigger levels, etc.

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

Command Syntax: `:CHANnel<N>:PROBe <attenuation>`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
`<attenuation>` ::= 0.9 to 1000

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

Query Syntax: `:CHANnel<N>:PROBe?`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)

Returned Format: `[:CHANnel<N>:PROBe] <attenuation><NL>`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
`<attenuation>` ::= 0.9 to 1000 (exponential - NR3 format)

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

RANGe

command/query

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

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

Command Syntax: `:CHANne1<N>:RANGe <range>`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
`<range>` ::= full-scale range value

Examples: `:OUTPUT 707;":CHANNEL1:RANGE 0.64"`

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

Query Syntax: `:CHANne1<N>:RANGe?`

Returned Format: `[[:CHANne1<N>:RANGe] <range><NL>`

Where:

`<N>` ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
`<range>` ::= full-scale range value (exponential - NR3 format)

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

SETup

SETup

query

The :CHANnel<N>:SETUP query returns the current settings for the CHANnel Subsystem commands.

Query Syntax: :CHANnel<N>:SETup?

Returned Format: :CHANnel<N>:DISPlay{ON|OFF};RANGe<range>;OFFset<value>;COUP{AC|DC|DCFifty};HFReject{ON|OFF};LFReject{ON|OFF};PROBE<attenuation><NL>

Where:

<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
<range> ::= full-scale range value (exponential - NR3 format)
<value> ::= offset value in volts (exponential - NR3 format)
<attenuation> ::= 0.9 to 1000 (exponential - NR3 format)

Example: DIM Stp\$[300]
OUTPUT 707;" :CHAN1:SET?"
ENTER 707;Stp\$
PRINT Stp\$

TTL**command**

The `:CHANNEL<N>:TTL` command sets the vertical range, offset, channel coupling, and trigger level of the selected channel for optimum viewing of TTL signals. The channel offset is set to 2.5 volts. The trigger level is set to 1.4 volts and the range is set to 8.0 volts full scale. Channel coupling is set to DC.

There is no query form of this command.

Command Syntax: `:CHANne1<N>:TTL`

Where:

`<N> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)`

Example: `OUTPUT 707;":CHAN1:TTL"`

Display Subsystem

Introduction

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

Note



The Display mode can be selected with the :ACQUIRE:TYPE command. The number of averages can be specified with the ACQUIRE:COUNT command.

The Display subsystem contains the following commands:

- COLumn
- CONNect
- DATA
- FORMat
- GRATICule
- INVerse
- LINE
- MASK
- PERSistence
- ROW
- SETup
- SCReen
- SOURce
- STRing
- TEXT
- MARKer | TMARKer | VMARKer

Figure 12-1 lists the syntax diagrams for the Display subsystem commands.

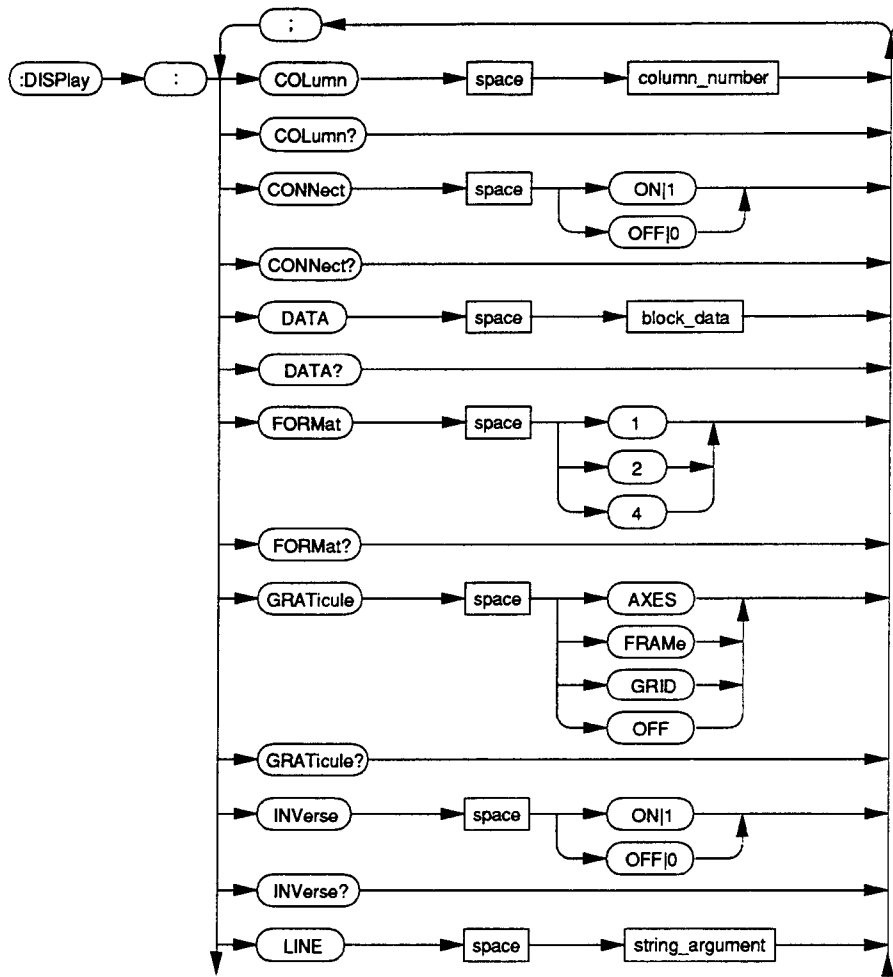


Figure 12-1. Display Subsystem Commands Syntax Diagram

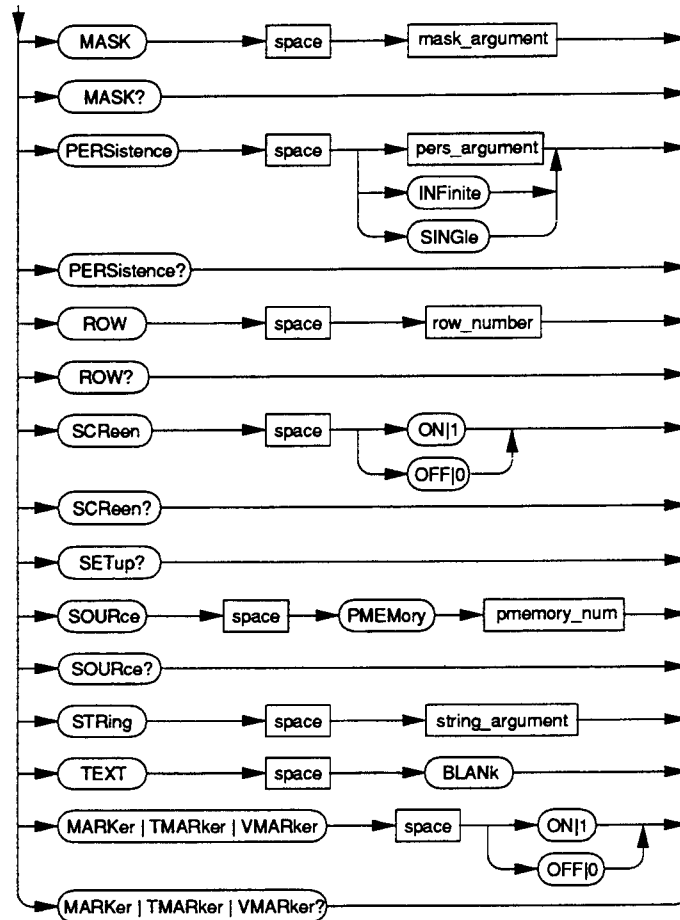


Figure 12-1. Display Subsystem Commands Syntax Diagram (continued)

| | |
|--------------------------|------------------------------------|
| column_number = | an integer, 0 through 72. |
| block_data = | block data in IEEE 488.2 # format. |
| mask_argument = | an integer, 0 through 255. |
| pers_argument = | a real number, 0.1 through 11. |
| pmemory_num = | an integer, 0 through 3. |
| row_number = | an integer, 0 through 24. |
| string_argument = | any quoted string. |

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

COLumn

command/query

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

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

Command Syntax: `:DISPlay:COLumn <number>`

Where:

`<number> ::= 0 through 72`

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

Query Syntax: `:DISPlay:COLumn?`

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

Where:

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

Example:

```
DIM C1mn$[30]
OUTPUT 707;":DISPLAY:COLUMN?"
ENTER 707;C1mn$
PRINT C1mn$
```

CONNECT

CONNECT

command/query

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

The `CONNECT` query returns the current setting of the connect-the-dots function. The returned status is indicated by using a 1 for on and a 0 for off.

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

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

Query Syntax: `:DISPlay:CONNect?`

Returned Format: `[.:DISP lay:CONNect] {1 | 0}<NL>`

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

DATA

command/query

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

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



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

Command Syntax: :DISPlay:DATA <binary block>

Where:

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

DATA

Query Syntax: :DISPlay:DATA?

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

Example:

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

This example transfers data from the active display memory to the controller, then transfers the data back to pixel memory 1 in the oscilloscope.



In program line 70, the space after :DISPLAY:DATA and before the quotation mark is required.

FORMat

command/query

The **:DISPLAY:FORMAT** command sets the number of display areas on the screen. **FORMAT 1** provides one display area and uses eight divisions for the full-scale range. **FORMAT 2** sets the number of screens to 2 and uses four divisions for the full-scale range. **FORMAT 4** sets the number of screens to 4 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: `:DISP lay:FORMat?`

Returned Format: `[:DISP lay:FORMat] {1 | 2 | 4}<NL>`

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

GRATICule

GRATICule

command/query

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

The `GRATICULE` query returns the type of graticule that is currently displayed.

Command Syntax: `:DISP lay:GRAT icule {OFF | GRID | AXES | FRAMe}`

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

Query Syntax: `:DISP lay:GRAT icule?`

Returned Format: `[:DISP lay:GRAT icule] {OFF | GRID | AXES | FRAMe}<NL>`

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

INVerse**command/query**

The `:DISPLAY:INVERSE` command determines whether text sent with the `LINE` or `STRING` command in the `DISPLAY` subsystem is to be written with the `INVERSE` attribute. If the inverse attribute is `ON` the text will be written in inverse video.

The `INVERSE` query returns the current state of this command.

Command Syntax: `:DISP lay:INVerse {{ON | 1} | {OFF | 0}}`

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

Query Syntax: `:DISP lay:INVerse?`

Returned format: `[:DISP lay:INVerse] {1 | 0}<NL>`

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

LINE

LINE

command

The `:DISPLAY:LINE` command writes a text string to the screen. The text is displayed starting at the location of the current row and column. The row and column can be set by the `:DISPLAY:ROW` and `:DISPLAY:COLUMN` commands prior to sending the `:DISPLAY:LINE` command. Text can be written over the entire screen with the `LINE` command.

If the text string is longer than the available space on the current line, the text wraps around to the start of the same line. In any case, the `ROW` value is incremented by one and the `COLUMN` value remains the same. The next `:DISPLAY:LINE` command will write on the next line of the display starting at the same column as the previous text. After writing line 24, the last line in the display area, `ROW` is reset to 0.

Command Syntax: `:DISPlay:LINE <quoted string>`

Where:

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

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

MASK

command/query

The :DISPLAY:MASK command inhibits the instrument from writing to selected areas of the screen. Text sent over the HP-IB with the line and string commands, or the SYSTEM:DSP command, is not affected by this command. The purpose of the command is to allow HP-IB text to be written anywhere on the screen, and to prevent the instrument from overwriting the text through its normal operation.

The mask parameter is an 8-bit integer in which each bit controls writing to an area of the screen. A zero inhibits writing to the area represented by the bit, and a 1 enables writing to that area.

The MASK query returns the current value of the MASK command.

Command Syntax: :DISP lay:MASK <value>

Where:

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

Example: OUTPUT 707;":DISPLAY:MASK 67"

The previous example enables writing to the Menu Area (bit 6), the Status Line (bit 1), and the Advisory Area (bit 0).

Query Syntax: :DISP lay:MASK?

Return Format: [:DISP lay:MASK] <value><NL>

Where:

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

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

MASK

Table 12-1. Display Mask Byte.

| Bit | Weight | Screen Area Affected |
|------------|---------------|-----------------------------|
| 7 | 128 | unused |
| 6 | 64 | Menu Area |
| 5 | 32 | Timebase Information |
| 4 | 16 | Measurement Result Area |
| 3 | 8 | Graticule Area |
| 2 | 4 | unused |
| 1 | 2 | Status Line |
| 0 | 1 | Advisory Area |

PERSistence

command/query

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

In the real-time time base mode, the parameters for this command are the keywords `INFINITE` or `SINGLE`.

In the repetitive time base mode, the parameter for this command is the keyword `INFINITE`, or real numbers from 0.5 through 10.0 representing the persistence in seconds. Any value less than 0.45 sets the `PERSISTENCE` to `MINIMUM`. Any value greater than 10 seconds sets the `PERSISTENCE` to `INFINITE`.

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

In the real-time time base mode the `PERSISTENCE` query returns the actual mnemonic: `SINGLE` or `INFINITE`.

In the repetitive time base mode the `PERSISTENCE` query returns the current persistence value. When `Minimum` is displayed, the value returned is 0. When `Infinite` is displayed, the value returned is 11.

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

Example: `OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE" !select time base mode`
`OUTPUT 707;":DISPLAY:PERSISTENCE 3.0"`

PERSistence

Query Syntax: `:DISP lay:PERSistence?`

Returned Format: `[:DISP lay:PERSistence] <va lue><NL>`

Where:

`<value> ::= {0 | .5 to 10 | 11}` (exponential - NR3 format) in the repetitive mode

Where:

`1.100E+1 = infinite`
`<value> ::= {SINGle | INFinite}` in the real-time mode

Example:

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

ROW**command/query**

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

The `ROW` query returns the current `ROW` number.

Command Syntax: `:DISP lay:ROW <row number>`

Where:

`<row number> ::= 0 through 24`

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

Query Syntax: `:DISPLAY:ROW?`

Returned Format: `[:DISP lay:ROW] <row number><NL>`

Where:

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

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

SCReen

SCReen

command/query

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

The `SCREEN` query returns the current setting of this function. The returned status is indicated by using a 1 for on and a 0 for off.

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

Command Syntax: `:DISP lay:SCReen {{ON | 1} | {OFF | 0}}`

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

Query Syntax: `:DISP lay:SCReen?`

Returned Format: `[:DISP lay:SCReen] {1 | 0}<NL>`

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

SETup

query

The `:DISPLAY:SETUP` query returns the current settings for the Display Subsystem commands.

Query Syntax: `:DISPlay:SETup?`

Returned Format: `:DISPlay:ROW {1 to 24};COL {0 to 72};FORMat {1|2|4};INVerse {ON|OFF};GRATicule {OFF|GRID|AXES|FReMe};PERS {<value>};SCReen {ON|OFF};SOURce {PMEMory{0|1|2|3};MASK <mask>;MARKer {ON|OFF} (<NL>`

Where:

`<value> ::= {0 | .5 to 10 | 11}` (exponential - NR3 format) in the repetitive mode

`1.100E+1 = infinite`
`<value> ::= {SINGle | INFinite}` in the real-time mode

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

Example: `DIM Stp$[300]`
`OUTPUT 707;":DISP:SET?"`
`ENTER 707;Stp$`
`PRINT Stp$`

SOURce

SOURce

command/query

The :DISPLAY:SOURCE command specifies the source or destination for the :DISPLAY:DATA query and command. The SOURCE command has one parameter which can be set to PMEMORY0 through PMEMORY3.

The SOURCE query returns the currently specified SOURCE.

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

Where:

PMemOry0 ::= active display
PMemOry1 ::= pixel memory 1
PMemOry2 ::= pixel memory 2
PMemOry3 ::= half-bright portion of the display (graticule, markers, and displayed memories)

Example:

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

This example transfers data from the active display to the controller, then transfers it back to pixel memory 1 in the oscilloscope.

Query Syntax: :DISPlay:SOURce?

Returned Format: [:DISP lay:SOURce] PMemOry{0 | 1 | 2 | 3}<NL>

Example:

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

STRing**command**

The **:DISPLAY:STRING** command writes a text string to the screen of the oscilloscope. The text is written starting at the current **ROW** and **COLUMN** values. If the column limit is reached (column 72), the excess text is written over the text on the left side of that line. If 90 or more characters are sent, the error -144, "Character data too long" is produced. The **STRING** command does not increment the **ROW** value; however, the **LINE** command does.

Command Syntax: `:DISPlay:STRing <quoted string>`

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

TEXT

TEXT

command

The `:DISPLAY:TEXT` command blanks the user text area on the screen. When this command is sent, all text on the entire screen is blanked. This command has only one parameter: `BLANK`.

There is no query form of this command.

The `:DISPLAY:SCREEN ON` command restores the user text to the screen.

Command Syntax: `:DISP1ay:TEXT BLANK`

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

MARKer | TMARKer | VMARKer

command/query

The :DISPLAY:MARKER | TMARKER | VMARKER command turns the all markers (x1/y1 and x2/y2) on and off.



All three commands perform the identical function, and are included only for compatibility with other oscilloscopes.

The MARKer | TMARKer | VMARKer query returns the current state of the specified marker.

Command Syntax: :DISP lay:MARKer|TMarker|VMARKer {{ON | 1} | {OFF | 0}}

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

Query Syntax: :DISP lay:MARKer|TMarker|VMARKer?

Returned Format: [:DISP lay:MARKer|TMarker|VMARKer] {0 | 1}<NL>

Example:
 DIM Mkr\$[30]
 OUTPUT 707;":DISP:MARK?"
 ENTER 707;Mkr\$
 PRINT Mkr\$



It is a recommended practice to turn the selected marker on before attempting to set them using a :MEASURE:(T/V)START or MEASURE:(T/V) STOP command. Also see MARKer subsystem for more information on using markers.

Function Subsystem

Introduction

The FUNCTION subsystem defines functions using the displayed channels or waveform memories as operands. All available channels and waveform memories are available for functions.

Functions are turned on/off with the VIEW/BLANK commands (see Chapter 7, "Root Level Commands"), or by using the FUNCTION[:DISPlay] command (in this chapter).

Some of the commands apply only when the FFT function is selected. The Function subsystem contains the following commands:

- ADD
- DIFFerentiate
- DISPlay
- FFT (fast Fourier transform)
- FREQuency (FFT function)
- INTegrate
- INVert
- LEVel (FFT function)
- MAGNify (FFT function)
- MULTiply
- OFFSet
- ONLY
- PEAK (FFT function)
- POINts (FFT function)
- RANGe
- SETup
- SPAN (FFT function)
- SUBTract
- VERSus
- WINDow (FFT function)

Figure 13-1 lists the syntax diagrams for the Function subsystem commands.

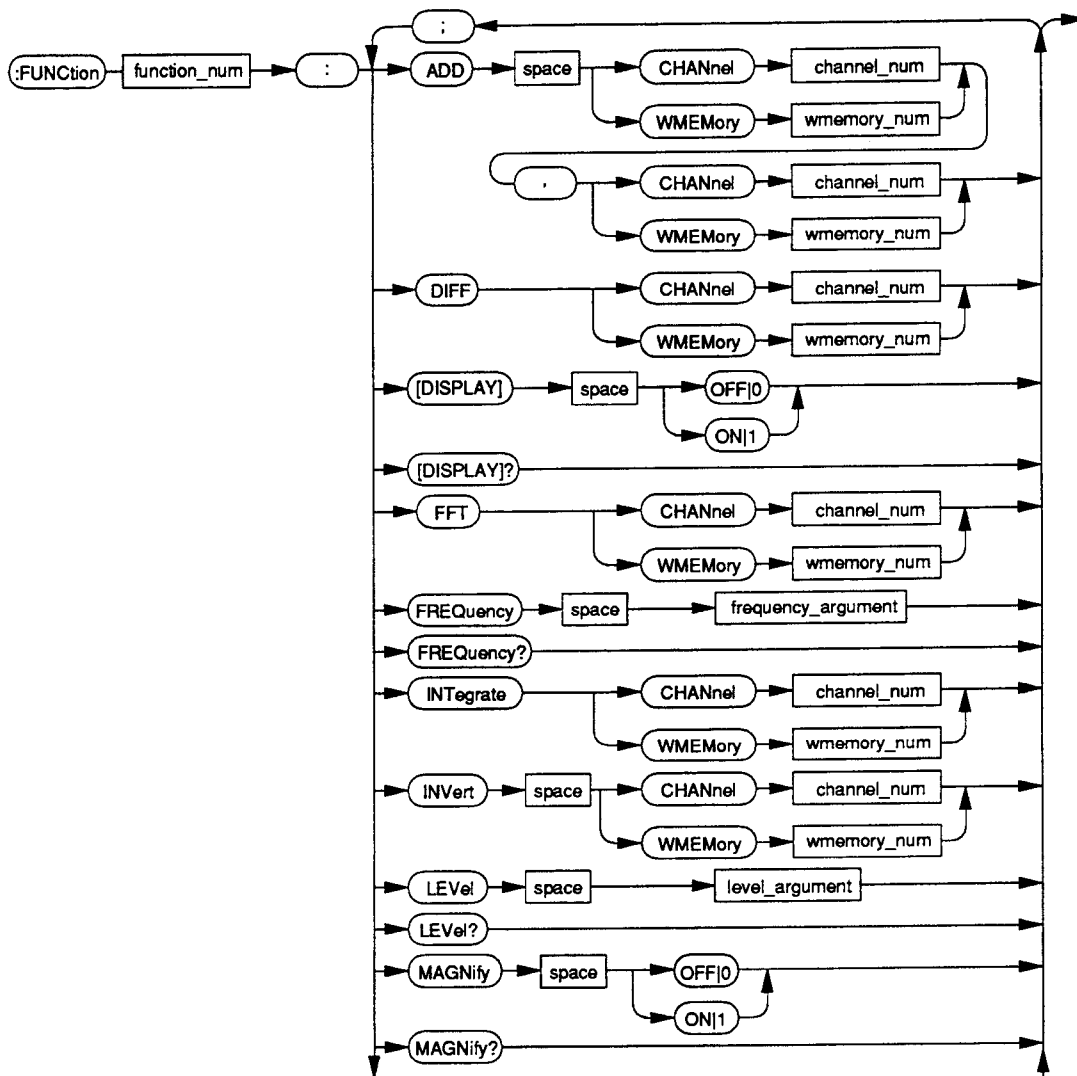


Figure 13-1. Function Subsystem Commands Syntax Diagram

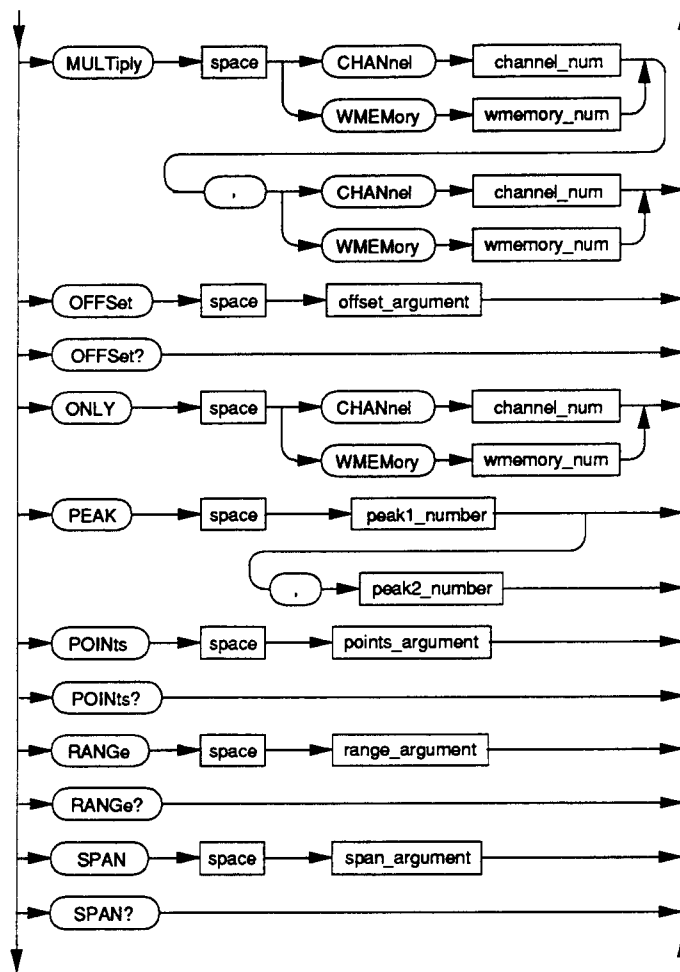
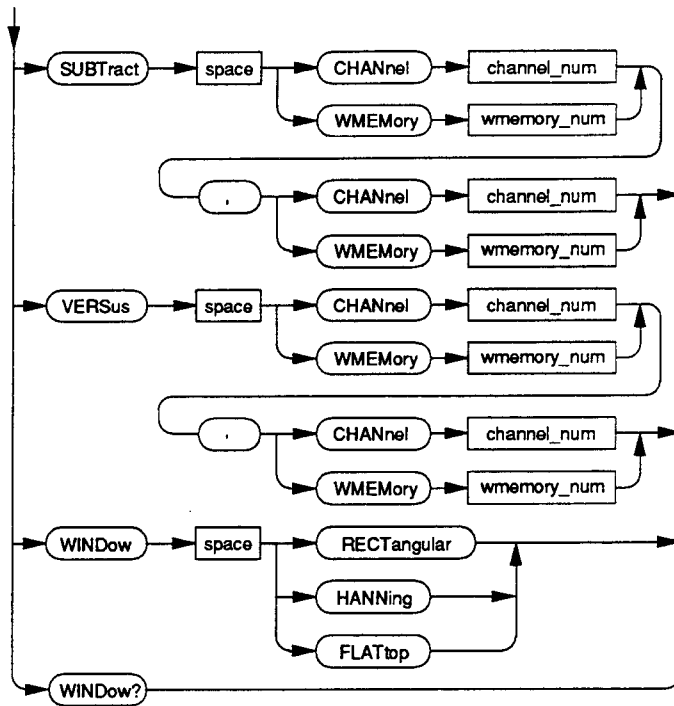


Figure 13-1. Function Subsystem Commands Syntax Diagram (continued)



- channel_num =** an integer, 1 through 4.
- frequency_argument =** center frequency in Hz.
- function_num =** an integer, 1 through 4.
- level_argument =** 0 to \pm 600 dBm.
- offset_argument =** 0 to \pm voltage full scale, or 0 to \pm 200 dBm for FFT functions.
- peak_num =** 1 to 99.
- points_num =** 512, 1024, 2048, 4096, or 8192.
- range_argument =** full screen voltage, or full screen dB for FFT functions.
- span_argument =** frequency span in Hz.
- wmemory_num =** an integer, 1 through 4.

Figure 13-1. Function Subsystem Commands Syntax Diagram (continued)

ADD**command**

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

Command Syntax: **:FUNction<N>:ADD** <operand>, <operand>

Where:

<N> ::= 1 through 4

<operand> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne1 4 | WMEMory1 |
WMEMory2 | WMEMory3 | WMEMory4}

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

Note

<operand>'s CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

DIFF

DIFF

command

The :FUNCTION<N>:DIFF command computes the voltage differences between consecutive points in time divided by the time bucket width, Δt .

$$d_1 = 0$$

$$d_n = \frac{c(n) - c(n-1)}{\Delta t}$$

The equation calculates the differential waveform of the operand, where d represents the differential waveform and c represents the operand voltage.



Note

The differential function will magnify signal noise. This noise will be less in the real time mode.

Command Syntax: :FUNct ion<N>:DIFF <operand>

Where:

<N> ::= 1 through 4

<operand> ::= {CHANnel1 | CHANnel2 | CHANnel3 | CHANnel 4 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}

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



Note

<operand>'s CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

[DISPlay]

command/query

The `:FUNCTION <N> [:DISPLAY]` command controls the individual function displays. On starts displaying, and off stops displaying the selected function.

The `DISPLAY` query returns the current setting of the command.

Note Because `[:DISPlay]` is an implied command, only the parameters need to be sent to turn a selected function on or off. For example, to turn function 1 display on send either `:FUNC1:DISP ON` or `:FUNC1 ON`.

Command Syntax: `:FUNCTION<N>[:DISPlay] {{ON | 1} | {OFF | 0}}`

Where:

`<N> ::= 1 through 4`

Example: `OUTPUT 707;":FUNCTION2:DISP ON"`

Query Syntax: `:FUNCTION1<N>[:DISPlay]?`

Where:

`<N> ::= 1 through 4`

Returned Format: `[:FUNCTION<N>[:DISPlay]] {1 | 0}<NL>`

Example:
`DIM Disp$[50]`
`OUTPUT 707;":FUNC1?"`
`ENTER 707;Disp$`
`PRINT Disp$`

FFT

FFT

command

The `:FUNCTION <N>:FFT` computes the fast Fourier transform function of the specified channel or memory. The FFT takes the digitized time record of the specified channel and transforms it to the frequency domain.

When the function is turned on and the `:FUNCTION<N>:FFT` command is sent to the oscilloscope, the FFT spectrum is plotted on the oscilloscope display as dBm versus frequency.

An example FFT program is in Chapter 5, "Example Programs".

Command Syntax: `:FUNCTION<N>:FFT<operand>`

Where:

`<N> ::= 1 through 4`

`<operand> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory3 | WMEMory4}`

Example: `OUTPUT 707;"FUNCTION1:FFT CHANNEL1"`



`<operand>`'s `CHANnel3` and `CHANnel4` are only available when the oscilloscope is an HP 54506B/54512B.

FREQuency

command/query

This command is used when an FFT function is selected. See FFT command.

The :FUNCTION<N>:FREQ command sets the center frequency when the FFT magnify mode is selected. See MAGNify command.

Command Syntax: :FUNct ion<N>:FREQuency<va lue>

Where:

<N>::=1 through 4

<value>::= frequency from 0 Hz to 1.5X of frequency span

Example: OUTPUT 707;"FUNCTION1:FREQ 25E6"

Query Syntax: :FUNct ion<N>:FREQ?

Returned Format: [FUNct ion<N>:FREQuency] <va lue><NL>

Example: DIM Frq\$[50]
OUTPUT 707;"FUNCTION2:FREQ?"
ENTER 707;Frq\$
PRINT Frq\$

INTEGRATE

INTEGRATE

command

The `:FUNCTION<N>:INTEGRATE` command computes the integral of the specified waveform. The integral is calculated by adding voltage points multiplied by the time bucket width, Δt .

$$I_n = \sum_{i=0}^{n-1} C_i \Delta t$$

The equation calculates the integral of the operand, where I represents the integral and C represents the operand data points.

Command Syntax: `:FUNCTION<N>:INTEGRATE <operand>`

Where:

`<N> ::= 1 through 4`
`<operand> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}`

Example: `OUTPUT 707;":FUNCTION2:INTEGRATE WMEMORY4"`



`<operand>`'s `CHANne13` and `CHANne14` are only available when the oscilloscope is an HP 54506B/54512B.

INVert**command**

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

Command Syntax: `:FUNCTION<N>:INVERT <operand>`

Where:

`<N> ::= 1 through 4`

`<operand> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 |
WMEMory2 | WMEMory3 | WMEMory4}`

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

Note 

`<operand>`'s CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

LEVel

LEVel

command/query

This command is used when an FFT function is selected. See FFT command.

The :FUNCTION<N>:LEVEL command sets the minimum search level for the peak search. The oscilloscope will search for peaks above the level specified with this command. To qualify as a peak, both positive and negative edges of the peak must be at least one vertical display division high (or one-half a display division in split screen), and be above the search level.

Command Syntax: :FUNCTION<N>:LEVEL <level>

Where:

<N> ::= 1 through 4

<level> ::= level value 0 to \pm 600 dBm

Example: OUTPUT 707;"FUNCTION1:LEVEL -25"

Query Syntax: :FUNCTION<N>:LEVEL?

Where:

<N> ::= 1 through 4

Returned Format: [FUNCTION<N>:LEVEL]<level><NL>

Example: DIM Lv1\$[50]
OUTPUT 707;"FUNCTION1:LEVEL?"
ENTER 707;Lv1\$
PRINT Lv1\$

MAGNify

command/query

This command is used when an FFT function is selected. See FFT command.

The :FUNCTION<N>:MAGNify command magnifies a particular frequency range of interest.

In the magnify mode, 500 points of the FFT are displayed. For POINTS = 512, half of the 500 points displayed in the magnify mode are interpolated. When the magnify is off, a compression algorithm is used to compress all the FFT points into 500 points. The compression algorithm accurately displays peaks and most noise. However, low points in the noise may occasionally be missed.

Command Syntax: :FUNCTION<N>:MAGNIFY {{ON|1} | {OFF|0}}

Where:

<N> ::= 1 through 4

Example: OUTPUT 707;"FUNCTION1:MAGNIFY ON"

Query Syntax: :FUNCTION<N>:MAGNIFY?

Where:

<N> ::= 1 through 4

Returned Format: [:FUNCTION<N>:MAGNify] {1|0}<NL>

Example: DIM Mfy\$[50]
OUTPUT 707;"FUNCTION1:MAGNIFY?"
ENTER 707;Mfy\$
PRINT Mfy\$

MULTiPLY

MULTiPLY

command

The :FUNCTION <N>:MULTIPLY command algebraically multiplies the two defined operands.

Command Syntax: :FUNct ion<N>:MULTiPLY <operand>, <operand>

Where:

<N> ::= 1 through 4

<operand> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 |
WMEMory2 | WMEMory3 | WMEMory4}

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

Note



<operand>'s CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

OFFSet

command/query

The `:FUNCTION<N>:OFFSET` command sets the voltage or, if the FFT function is selected, the dBm value represented at the center of the screen for the selected function.

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

Command Syntax: `:FUNCTION<N>:OFFSet <offset>`

Where:

`<N>` ::= 1 through 4
`<offset>` ::= offset value of 0 to \pm voltage full scale, or 0 to \pm 200 dBm for FFT function

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

Query Syntax: `:FUNCTION<N>:OFFSet?`

Where:

`<N>` ::= 1 through 4

Returned Format: `[:FUNCTION<N>:OFFSet] <offset><NL>`

Where:

`<N>` ::= 1 through 4
`<offset>` ::= offset value (see above) (exponential - NR3 format)

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

ONLY

ONLY

command

The `:FUNCTION<N>:ONLY` command produces another copy of the operand and places it in the specified function. The `ONLY` command is useful for scaling channels and memories with the `:FUNCTION<N>:RANGE` and `:FUNCTION<N>:OFFSET` commands.

Command Syntax: `:FUNCTION<N>:ONLY <operand>`

Where:

`<N>` ::= 1 through 4

`<operand>` ::= {`CHANne11` | `CHANne12` | `CHANne13` | `CHANne14` | `WMEMemory1` | `WMEMemory2` | `WMEMemory3` | `WMEMemory4`}

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



`<operand>`'s `CHANne13` and `CHANne14` are only available when the oscilloscope is an HP 54506B/54512B.

PEAK

command

This command is used when an FFT function of the oscilloscope is selected. See FFT command.

The :FUNCTION<N>:PEAK command causes a peak search to be performed. The peak search sets Vmarker1 (y1) and the start marker on the first peak specified and sets Vmarker2 (y2) and the stop marker on the second peak specified. Peaks from 1 to 99 can be specified.

The marker values in frequency and dBm are automatically displayed at the bottom of the oscilloscope screen. The difference in frequency and dBm between the two peaks is also displayed.

The marker values can be read over the bus using the VSTART, VSTOP, VDELTA, TSTART, TSTOP, and TDELTA queries. See Chapter 15, "Measure Subsystem".

If a peak is not found a message is displayed on the screen. The vertical marker is set to the right edge of the screen and the horizontal marker is set to the bottom of the screen.

For maximum frequency accuracy, the FFT horizontal magnify function should be turned on. See MAGNIFY command.

Command Syntax: :FUNCTION<N>:PEAK <peak 1 number>, <peak 2 number>

Where:

<N> ::= 1 through 4

<peak_number> ::= 1 through 99

Example: OUTPUT 707;":FUNCTION1:PEAK 5, 7"

POINTS

POINTS

command/query

This command is used when the FFT function is selected. See FFT command.

The `:FUNCTION<N>:POINTS <points_number>` assigns the number of data points in the record to be used for the FFT computation. The number of points affects the computation speed, frequency resolution, and the noise floor. The frequency resolution of an FFT is $\frac{F_s}{N}$, where: F_s is the sampling frequency and N is the number of points. The maximum signal-to-noise ratio for an FFT is also related to the number of points and the A/D converter bits of resolution.

An FFT performed on a small number of points can be computed faster. An FFT performed on a large number of points has better frequency resolution and a lower noise floor. The resolution affects how accurately frequencies can be measured and how well two frequencies that are close together can be resolved.

If the oscilloscope is in repetitive mode, or the operand is a memory that was stored from the repetitive mode, the number of points defaults to 512 and cannot be changed.

When the operand is a repetitive source, on-screen data is used for the FFT. When the operand is real time, the FFT is performed on data in the 8k record, but not necessarily data that is displayed on the screen. The points used from the source record for the FFT are based on the TIMEBASE reference setting of left, center, or right. See also `:TIMEbase:REFerence` command.

When `:TIMEbase:REFerence LEFT` is sent, data from the beginning of the record is used. When `:TIMEbase:REFerence CENTER` is sent, data from the middle of the record is used. When `:TIMEbase:REFerence RIGHT` is sent, data from the end of the record is used. The memory bar above the graticule indicates the portion of the data record used for the FFT. The memory bar showing the FFT record is displayed whenever the FFT menu with the # of points key is on.

POINTS

When it is necessary to window the data, the LEFT REFERENCE is a good choice since the FFT record then begins on the left side of the screen.

An FFT with an input record of N points becomes a frequency record of N points. Because half of the points are above the Nyquist frequency (sample frequency/2) and provide redundant information, they are not used so the number of frequency points are half the number of input points.

The FFTs are computed on records that are powers of two. If a repetitive source (500 points) or an entire record (8000 points) is selected, the record is zero padded at the ends to achieve 512 and 8192 points.

If the number of FFT points is 512, 256 additional frequency points are created by interpolating between the actual points to generate 512 frequency points. Table 13-1 lists the number of input points and the resultant number of frequency points.

Table 13-1. FFT Points

| INPUT POINTS | FREQUENCY POINTS |
|---------------------|-------------------------|
| 512 (repetitive) | 512 |
| 512 (real time) | 512 |
| 1024 | 512 |
| 2048 | 1024 |
| 4096 | 2048 |
| 8192 | 4096 |

POINTS

Command Syntax: :FUNCTION<N>:POINTS <points_number>

Where:

<N> ::= 1 through 4

<points_number> ::= {512 | 1024 | 2048 | 4096 | 8192}

Example: OUTPUT 707;":FUNCTION1:POINTS 2048"

Query Syntax: :FUNCTION<N>:POINTS?

Where:

<N> ::= 1 through 4

Returned Format: [:FUNCTION<N>:POINTS] <number of points>

Where:

<N> ::= 1 through 4

<number of points> ::= {512 | 1024 | 2048 | 4096 | 8192}

Example: DIM Pts\$[50]
OUTPUT 707;":FUNCTION1:POINTS?"
ENTER 707;Pts\$
PRINT Pts\$

RANGe

command/query

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

The `RANGe` query returns the current range setting for the specified function.

Command Syntax: `:FUNction<N>:RANGe <range>`

Where:

`<N> ::= 1 through 4`

`<range> ::= full scale vertical range in volts, dB, or dBm`

Example: `OUTPUT 707;":FUNCTION2:RANGe 400E-3"`

Query Syntax: `:FUNction<N>:RANGe?`

Where:

`<N> ::= 1 through 4`

Returned Format: `[:FUNction<N>:RANGe] <range><NL>`

Where:

`<N> ::= 1 through 4`

`<range> ::= current range setting (exponential - NR3 format)`

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

SETup

SETup

query

The **:FUNCTION<N>:SETUP** query returns the current settings for the FUNCTION Subsystem commands.

Query Syntax: **:FUNCTION<N>:SETup?**

Returned Format: **:FUNCTION<N>:FREQUENCY<value>;LEVEL<level>;MAGNIFY{ON|OFF};OFFSET<offset>;POINTS<points>;RANGE<range>;SPAN<frequency>;WINDOW{RECT|HANN|FLAT}<NL>**

Where:

<N> ::= 1 through 4
<value> ::= freq in hertz from 0 to 1.5X span (exponential - NR3 format)
<level> ::= 0 +\ -600 dBm
<offset> ::= offset value in volts or +\ -200 dBm for FFT (exp - NR3 format)
<points> ::= 512 | 1024 | 2048 | 4096 | 8192
<range> ::= full-scale range value (exponential - NR3 format)
<frequency> ::= freq in hertz (exponential - NR3 format)

Example:
DIM Stp\$[200]
OUTPUT 707;":FUNC1:SET?"
ENTER 707;Stp\$
PRINT Stp\$

SPAN**command/query**

This command is used when the FFT function is selected . See FFT command.

The :FUNCTION<N>:SPAN command controls the frequency span of the FFT record. Changing the span for an FFT function with a channel source causes the timebase to change. The span is the sample frequency divided by two. Since the sample frequency for memories is fixed once a record is stored, the span is also fixed and cannot be changed for FFT functions with a memory source.

Command Syntax: :FUNction<N>:SPAN<frequency>

Where:

<N> ::= 1 through 4
<frequency> ::= number in Hertz

Example: OUTPUT 707::"FUNCTION1:SPAN 25E6"

Query Syntax: :FUNCTION<N>:SPAN?

Returned Format: [FUNCTION<N>:SPAN]<frequency><NL>

Example: DIM Spn\$[50]
OUTPUT 707::"FUNCTION1:SPAN?"
ENTER 707;Spn\$
PRINT Spn\$

SUBTract

SUBTract

command

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

Command Syntax: `:FUNction<N>:SUBTract <operand1>,<operand2>`

Where:

`<N> ::= 1 through 4`
`<operand1> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}`
`<operand2> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}`

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

In this example, Waveform Memory 2 is algebraically subtracted from Waveform Memory 3.



`<operand>`'s CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

VERSus

command

The `:FUNCTION<N>:VERSUS` command allows X vs Y displays with two operands. The first operand defines the Y-axis and the second defines the X-axis. The Y-axis range and offset is initially equal to that of the first operand, and can be adjusted with the `FUNCTION<N>:RANGE` and `FUNCTION<N>:OFFSET` commands.

The X-axis range and offset is always equal to that of the second operand. They can only be changed by changing the vertical settings of the second operand.

Command Syntax: `:FUNCTION<N>:VERSUS <Y_operand>,<X_operand>`

Where:

`<N> ::= 1 or 2`

`<Y_operand> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}`

`<X_operand> ::= {CHANne11 | CHANne12 | CHANne13 | CHANne14 | WMEMory1 | WMEMory2 | WMEMory3 | WMEMory4}`

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



`<operand>`'s CHANne13 and CHANne14 are only available when the oscilloscope is an HP 54506B/54512B.

This command is used when the FFT function is selected. See FFT command.

The `:FUNCTION <N>:WINDOW` command allows selection of three windows for the FFT function.

The FFT operation assumes that the time record repeats. Unless there is an integer number of cycles of the sampled waveform in the record, a discontinuity is created at the end of the record. This introduces additional frequency components into the spectrum about the actual peaks. This is referred to as leakage. In order to minimize leakage, windows that approach zero smoothly at the beginning and end of the record are employed as filters to the FFTs.

Each window is useful for certain classes of input signals.

The **rectangular** window is essentially no window, all points are multiplied by 1. The **rectangular** window is useful for transient signals and signals where there are an integral number of cycles in the time record. The **hanning** window is useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together or for making frequency measurements. The **flattop** window is best for making accurate amplitude measurements of frequency peaks.

Command Syntax: `:FUNCTION<N>:WINDOW {RECTangular | HANNing | FLATtop}`

Example: `OUTPUT 707;":FUNCTION<N>:WINDOW RECT`

Query Syntax: `:FUNCTION<N>:WINDOW?"`

Example: `DIM Wnd$[50]
OUTPUT 707;":FUNCTION1:WINDOW?"
ENTER 707;Wnd$
PRINT Wnd$`

Hardcopy Subsystem

Introduction

The HARDCOPY subsystem commands set various parameters for printing and plotting waveforms from the HP 54505B, HP 54506B, HP 54510B and HP 54512B.

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

The Hardcopy subsystem contains the following commands:

- LENGth
- MODE
- PAGE
- PLOT:AREa
- PLOT:INITialize
- PLOT:PEN|COLor

Figure 14-1 lists the syntax diagrams for the Hardcopy subsystem commands.

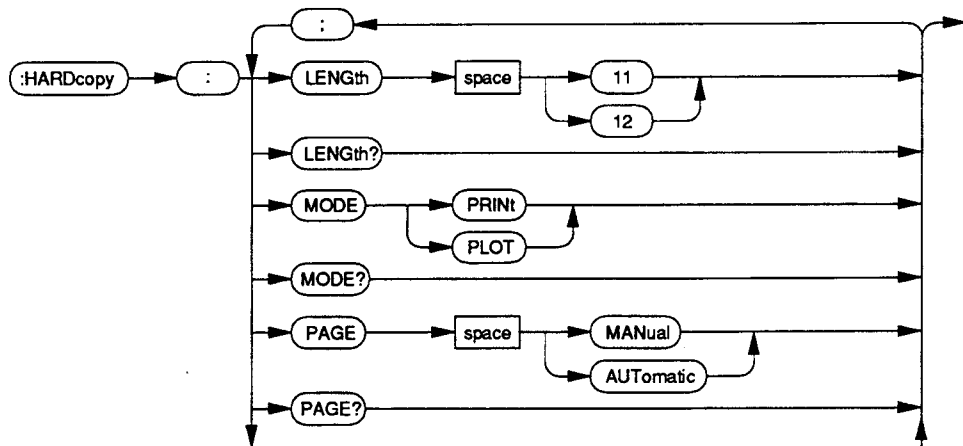


Figure 14-1. Hardcopy Subsystem Commands Syntax Diagram

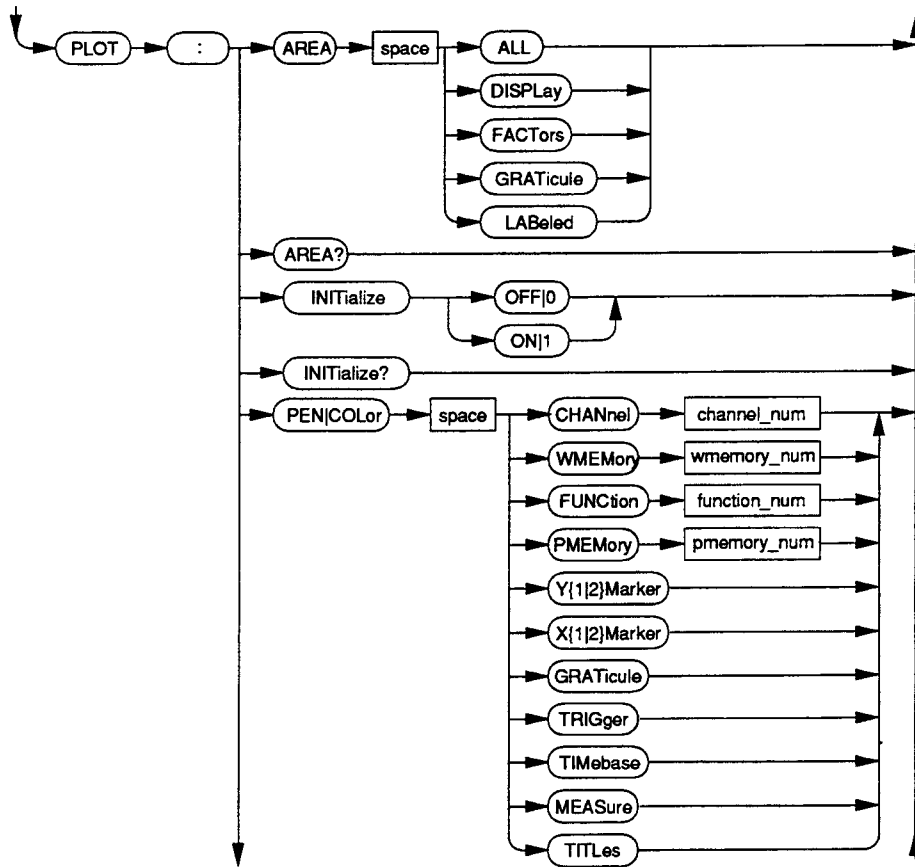
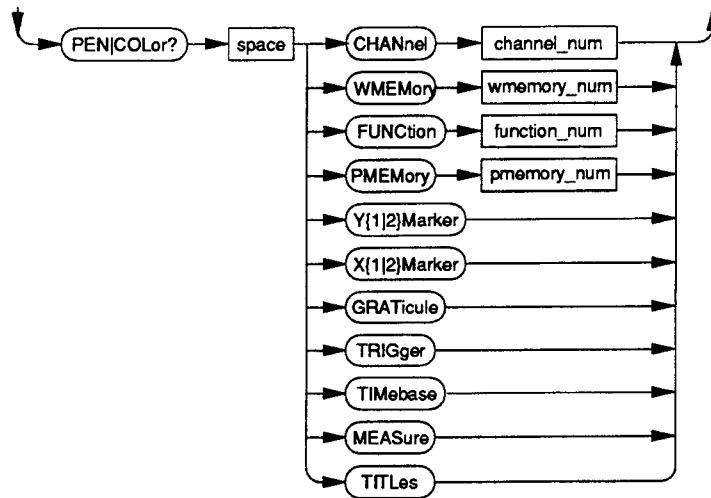


Figure 14-1. Hardcopy Subsystem Commands Syntax Diagram (continued)



- channel_num =** an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B).
- function_num =** an integer, 1 through 4.
- pen_num =** an integer, 0 to 8.
- pmemory_num =** an integer, 1 or 2.
- wmemory_num =** an integer, 1 to 4.

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

LENGth

LENGth

command/query

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

The `LENGTH` query returns the current length setting.

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

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

Query Syntax: `:HARDcopy:LENGth?`

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

Example:
`DIM Lgth$[50]`
`OUTPUT 707;":HARDCOPY:LENGTH?"`
`ENTER 707;Lgth$`
`PRINT Lgth$`

MODE**command/query**

The `:HARDCOPY:MODE` command sets the HP-IB device mode for a printer or plotter output.

The `MODE` query returns the current setting of the `MODE` command.

Command Syntax: `:HARDcopy:MODE {PRINT | PLOT}`

Example: `OUTPUT 707;":HARD:MODE PRINT"`

Query Syntax: `:HARDcopy:MODE?`

Returned Format: `[:HARDcopy:MODE] {PRINT | PLOT}<NL>`

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

PAGE

PAGE

command/query

The :HARDCOPY:PAGE command configures the oscilloscope to send a formfeed to the printer after it outputs a hardcopy.

If the PAGE command is set to AUTOMATIC, a formfeed occurs at the end of the hardcopy; otherwise, the page scrolls up by 4 lines.

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

Command Syntax: :HARDcopy:PAGE {MANua1 | AUTomatic}

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

Query Syntax: :HARDcopy:PAGE?

Returned Format: [:HARDcopy:PAGE] {MANua1 | AUTomatic}<NL>

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

PLOT:AREA**command/query**

The **:HARDCOPY:PLOT:AREA** command selects the area to be plotted.

The **PLOT:AREA** query returns the current selection for the **PLOT:AREA** command.

Command Syntax: **:HARDcopy:PLOT:AREA** {ALL | DISPlay | FACTors | GRATicule | LABeled}

Example: OUTPUT 707;":HARD:PLOT:AREA ALL"

Query Syntax: **:HARDcopy:PLOT:AREA?**

Returned Format: [**:HARDcopy:PLOT:AREA**] {ALL | DISPlay | FACTors | GRATicule | LABeled}<NL>

Example: DIM P1\$[50]
OUTPUT 707;":HARDCOPY:PLOT:AREA?"
ENTER 707;P1\$
PRINT P1\$

PLOT:INITialize

PLOT:INITialize

command/query

The :HARDCOPY:PLOT:INITialize command sets the plotter to a known state. Sending the command :HARDCOPY:PLOT:INITIALIZE ON sends the "IN" command to the plotter which sets P1 and P2 to the default value. Sending the command :HARDCOPY:PLOT:INITIALIZE OFF sends the "DF" command to the plotter and does not set P1 and P2 to the default value.

The PLOT:INITIALIZE query returns the current setting of the command.

Command Syntax: :HARDcopy:PLOT:INITialize {{ON | 1} | {OFF | 0}}

Example: OUTPUT 707;":HARD:PLOT:INITIALIZE ON"

Query Syntax: :HARDcopy:PLOT:INITialize?

Returned Format: [:HARDcopy:PLOT:INITialize] {1 | 0}<NL>

Example: DIM P1\$[50]
OUTPUT 707;":HARDCOPY:PLOT:INIT?"
ENTER 707;P1\$
PRINT P1\$

PLOT:PEN|COLor

command/query

The :HARDCOPY:PLOT:PEN|COLOR command selects the plotter pen to plot individual items on the screen. By selecting the appropriate plotter pen, you can plot different items on the screen in different colors to distinguish them from each other. Pen/color numbers 1 through 8 select the corresponding pens on the plotter. Pen/color number 0 returns the plotter pen to its carousel without plotting the selected item.

The PLOT:PEN|COLOR query returns the current pen/color selection for an individual item on the screen.

Command Syntax: :HARDCOPY:PLOT:PEN|COLor <item>,<pen_number>

Where:

<item> ::= {CHANnel{1 | 2 | 3 | 4} | WMemory{1 | 2 | 3 | 4} | FUNCTION{1 | 2 | 3 | 4} | PMemory{1 | 2} | Y{1|2}Marker | X{1|2}Marker | GRATICule | TRIGger | TIMEbase | MEASure | TITLes}
 <pen_number> ::= 0 through 8

Example: OUTPUT 707;":HARD:PLOT:COLOR CHANNEL1,2"

Query Syntax: :HARDCOPY:PLOT:COLor? <item>

Returned Format: [:HARDCOPY:PLOT:PEN|COLor] <pen_number><NL>

Where:

<item> ::= {CHANnel{1 | 2 | 3 | 4} | WMemory{1 | 2 | 3 | 4} | FUNCTION{1 | 2 | 3 | 4} | PMemory{1 | 2} | Y{1|2}Marker | X{1|2}Marker | GRATICule | TRIGger | TIMEbase | MEASure | TITLes}
 <pen_number> ::= 0 through 8 (integer - NR1)

Example: DIM P1\$[50]
 OUTPUT 707;":HARDCOPY:PLOT:COLOR? CHANNEL1"
 ENTER 707;P1\$
 PRINT P1\$

Note CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

Measure Subsystem

Introduction

The commands in the MEASURE subsystem are used to make parametric measurements on displayed waveforms and to report the settings of the voltage and time markers. Some commands in this subsystem can be used to set the voltage and time markers to specified voltages, times, or events.

The Measure subsystem contains the following commands:

- ALL
- COMPare
- CURSor
- DEFine
- DELay
- DESTination
- DUTycycle
- ESTart
- ESTop
- FALLtime
- FREQuency
- LIMittest
- LOWer
- MODE
- NWIDTH
- OVERshoot
- PERiod
- POSTfailure
- PREShoot
- PWIDTH
- RESults
- RISetime
- SCRatch
- SOURce
- STATistics
- TDELta
- TMAX
- TMIN
- TSTart
- TSTOp
- TVOLt
- UNITs
- UPPer
- VACRms
- VAMPplitude
- VAVerage
- VBASe
- VDCRms
- VDELta
- VFIFty
- VMAX
- VMIN
- VPP
- VRELative
- VRMS
- VSTart
- VSTOp
- VTIMe
- VTOP
- WCOMpare

Figure 15-1 lists the syntax diagrams for the Measure subsystem commands.

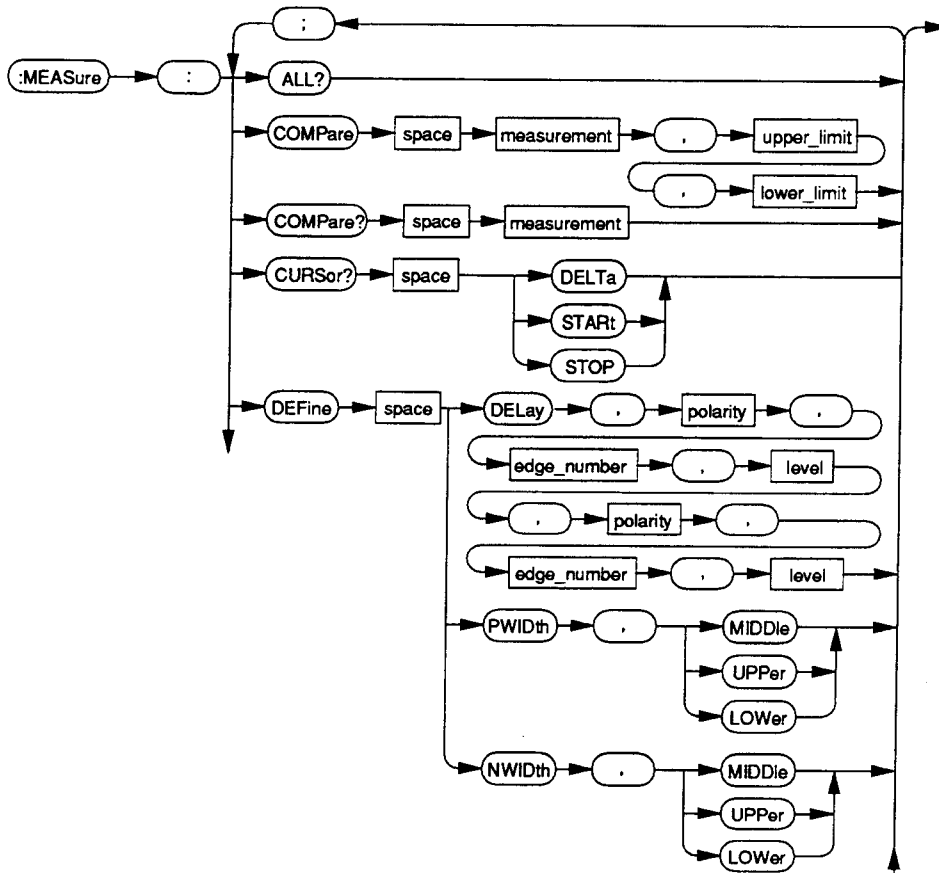


Figure 15-1. Measure Subsystem Commands Syntax Diagram

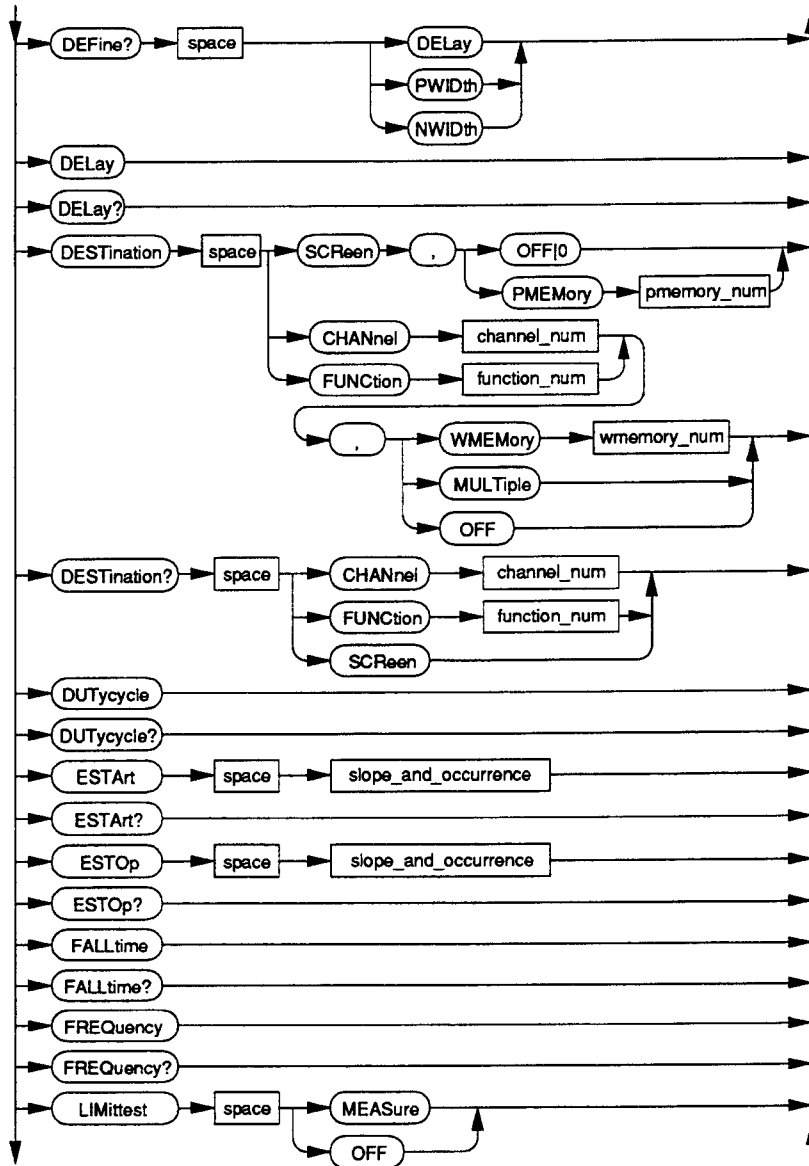


Figure 15-1. Measure Subsystem Commands Syntax Diagram (continued)

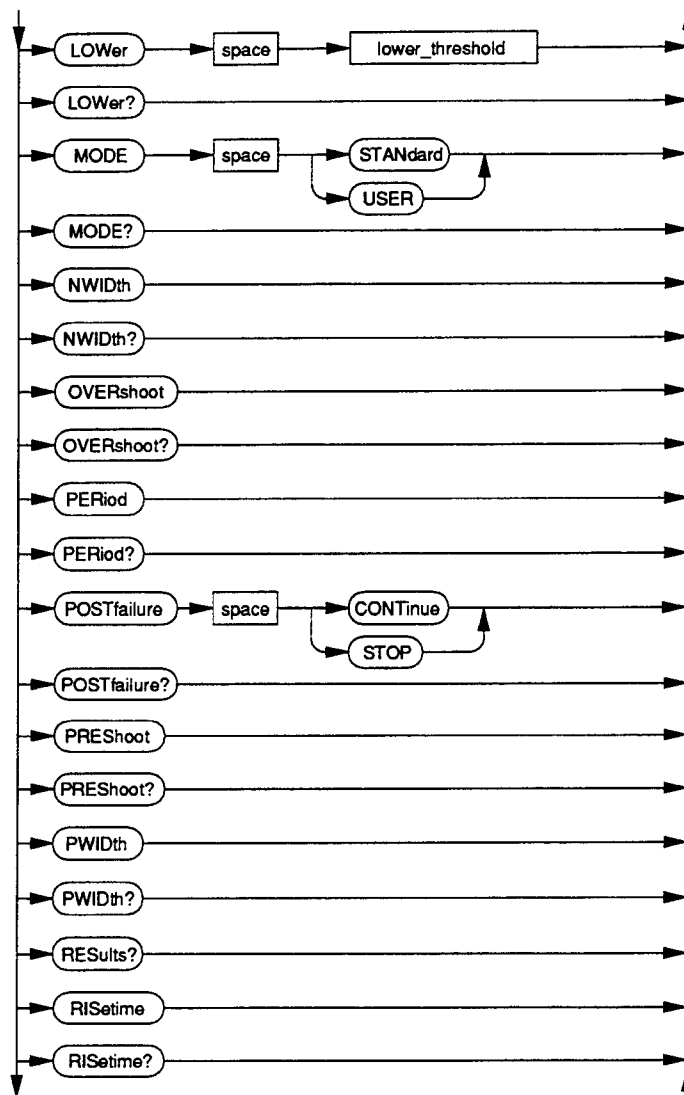


Figure 15-1. Measure Subsystem Commands Syntax Diagram (continued)

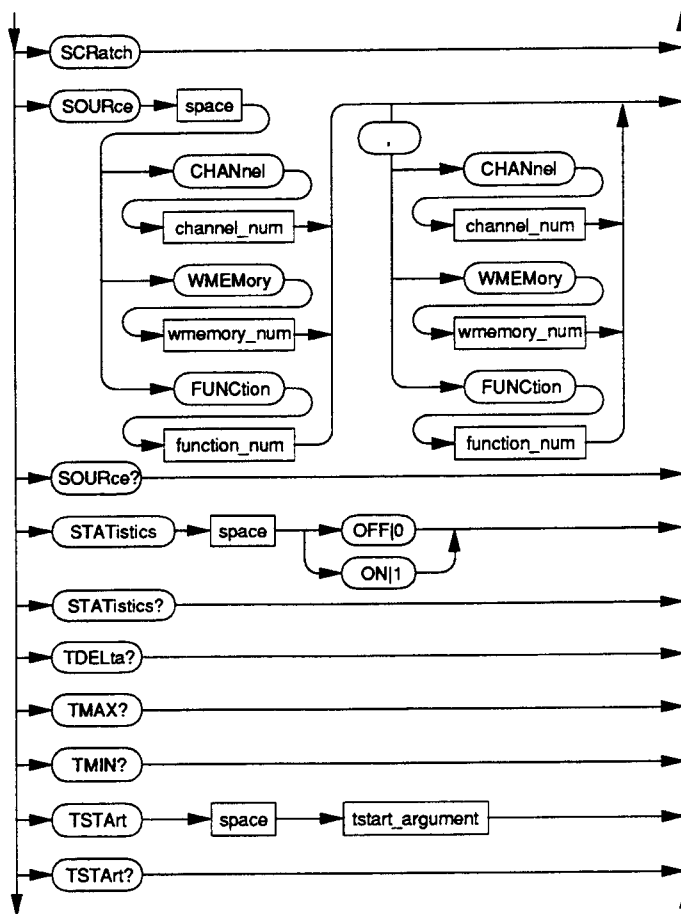


Figure 15-1. Measure Subsystem Commands Syntax Diagram (continued)

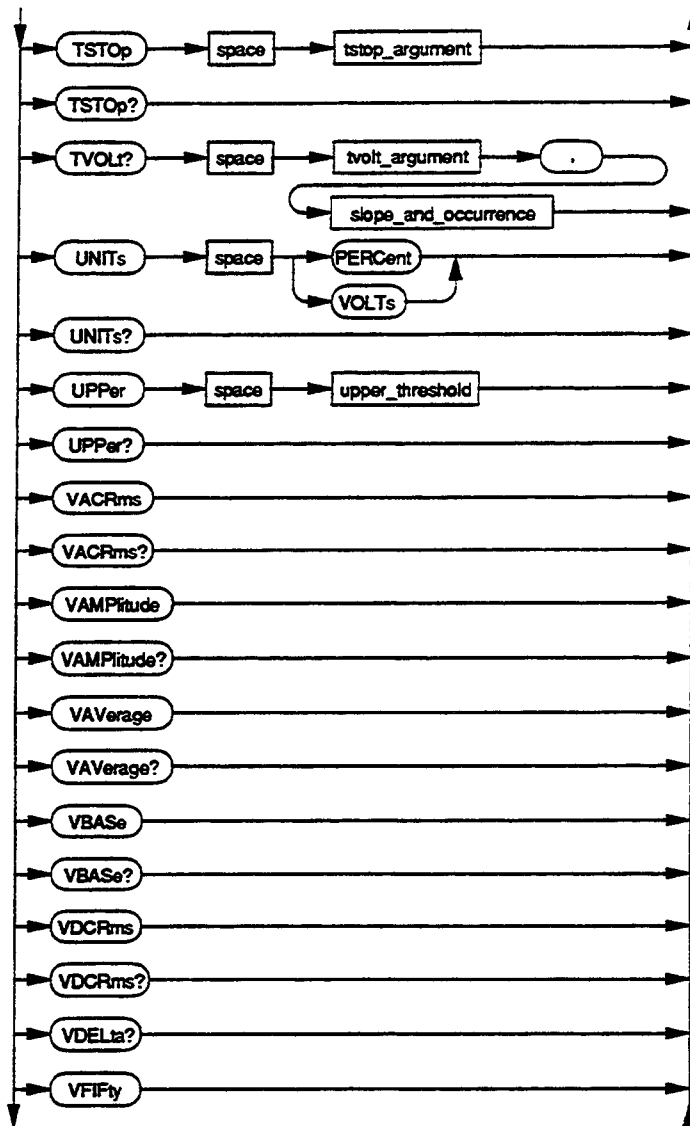


Figure 15-1. Measure Subsystem Commands Syntax Diagram (continued)

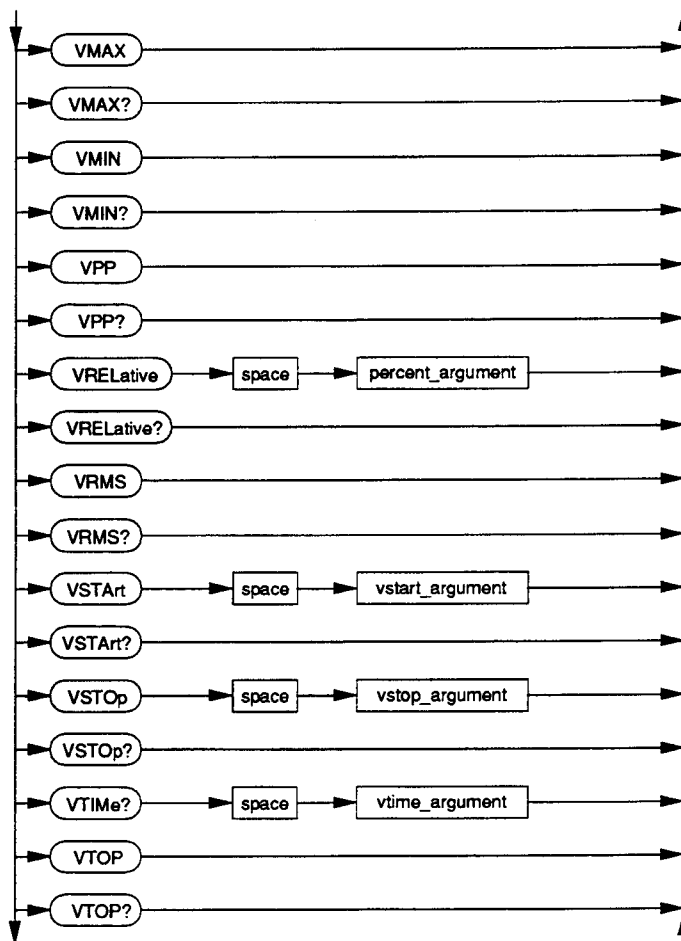


Figure 15-1. Measure Subsystem Commands Syntax Diagram (continued)

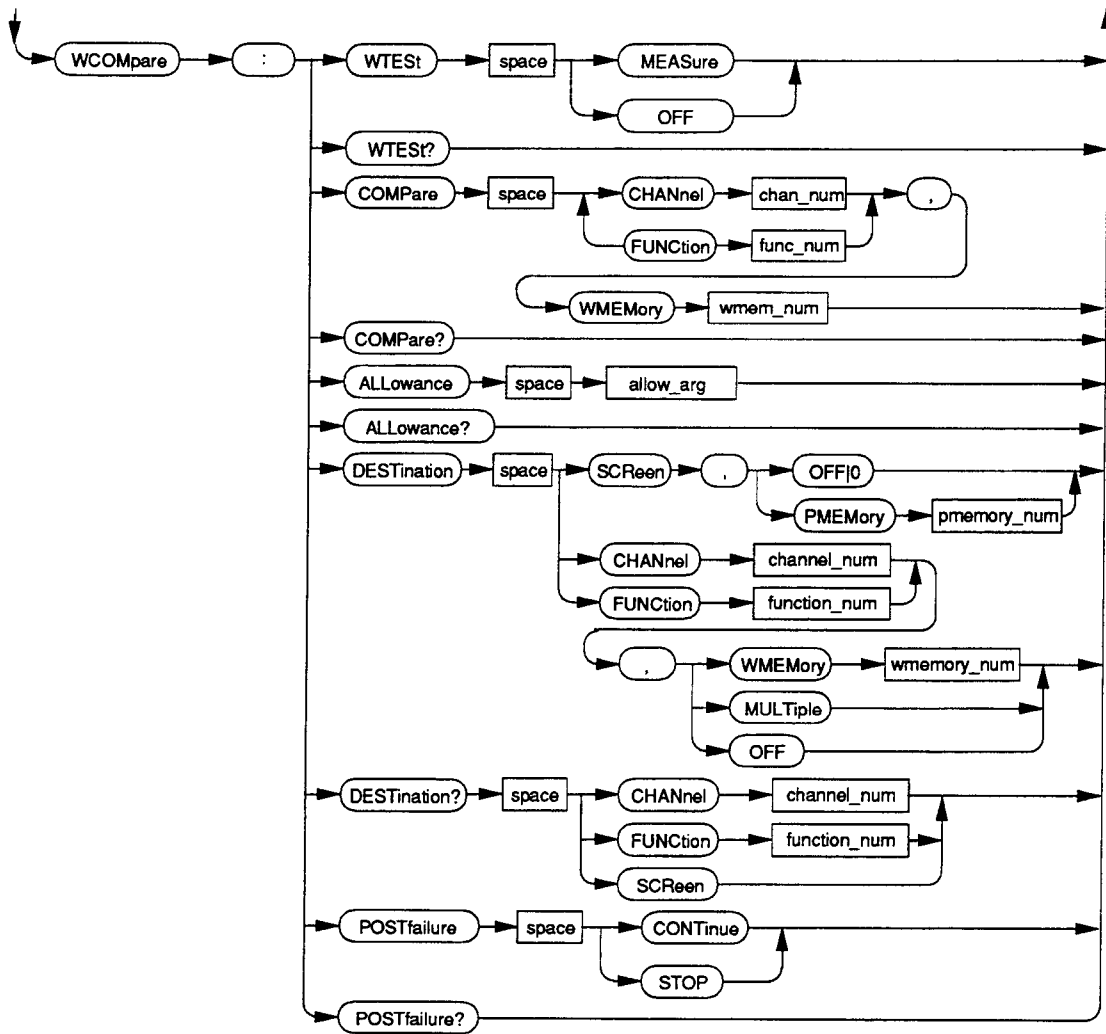


Figure 15-1. Measure Subsystem Commands Syntax Diagram (continued)

| | |
|-------------------------------|--|
| channel_num = | an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B). |
| edge_number = | an integer, 1 to 4000. |
| function_num = | an integer, 1 through 4. |
| level = | MIDDLE, UPPER, or LOWER. |
| lower_limit = | lower limit for compare. |
| lower_threshold = | lower threshold value in percent or volts. |
| measurement = | name of the measurement to be compared. |
| percent_argument = | an integer, 0 to 100. |
| pmemory_num = | an integer, 1 or 2. |
| polarity = | POSITIVE or NEGATIVE. |
| slope_and_occurrence = | an integer, -4000 to 4000 (excluding 0) specifying a displayed edge. |
| tstart_argument = | time in seconds from the trigger. |
| tstop_argument = | time in seconds from the trigger. |
| tvolt_argument = | a real number specifying voltage. |
| upper_limit = | upper limit for compare. |
| upper_threshold = | upper threshold value in percent or volts. |
| vstart_argument = | a real number within the voltage range. |
| vstop_argument = | a real number within the voltage range. |
| vtime_argument = | a real number in the horizontal display window. |
| wmemory_num = | an integer, 1 through 4. |

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

Measurement Setup

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

- For a period or frequency measurement, at least one complete cycle must be displayed.
- For a pulse width measurement, the entire pulse must be displayed.
- For a rise time measurement, the leading (positive-going) edge of the waveform must be displayed.
- For a fall time measurement, the trailing (negative-going) edge of the waveform must be displayed.

User-Defined Measurements

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

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

Measurement Error

If a measurement cannot be made, the value $+9.99999E+37$ is returned for that parameter. This will be returned when no data is available, or if the measurement cannot be performed. For example, when performing a Vpp? measurement on a clipped waveform, or a FREQ? measurement without a full cycle present.

Making Measurements

If more than one waveform, edge, or pulse is displayed, time measurements are made on the first (left-most) portion of the displayed waveform.

When any of the defined measurements are requested, the oscilloscope first determines the top (100%) and base (0%) voltages of the waveform. From this information, it can determine the other important voltage values (10%, 90%, and 50% voltage values) for making measurements.

The 10% and 90% voltage values are used in the rise time and fall time measurements when standard measurements are selected. The 50% voltage value is used for measuring frequency, period, pulse width, and duty cycle with standard measurements selected.

The measurements can also be made using user-defined parameters instead of the standard measurement values.

When the command form of the measurement or front-panel measurements are used, the instrument is placed into the continuous measurement mode. When the query form of these measurements is used, the continuous measurement mode is turned off. Then the measurement is made one time, and the measurement result is returned.

Except for `V AVERAGE`, `V RMS`, `V AC RMS`, and `V DC RMS` measurements, which use only one cycle, voltage measurements are made using the entire display. Therefore, if you want to make a measurement on a particular cycle, display only that cycle on the screen.

All voltage values are returned in volts. Returned voltage values are measured with zero volts as the reference. The value returned for the VDELTA? query is the difference between y1marker and y2marker in volts.

All time values are returned in seconds. Returned time values are measured with the trigger point (time 0) as the reference. The value returned for TDELTA? is the time difference between the x2marker and x1marker.

Measurements are made on the displayed waveforms specified by the SOURCE command. The SOURCE command allows two sources to be specified. When two sources are specified, y1marker is assigned to the first specified source and y2marker is assigned to the second specified source.

Most measurements can only be made on a single source. If one of these measurements is made with two sources specified, the measurement is made on the first source specified.

If the horizontal scaling is questionable, an "error 11" is placed in the error queue. In this case, the value returned is the most accurate value that can be made using the current scaling. You might be able to obtain a more accurate measurement by rescaling the horizontal to obtain more data points on the edge.

For more information about measurement algorithms, refer to chapter 26, "Algorithms."

ALL

query

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

To make a measurement, the portion of the waveform required for the measurement must be displayed. Time measurements are made on the first (left-most) displayed edges of the waveforms. To obtain the most accurate measurement possible, use proper horizontal scaling.

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

Query Syntax: :MEASure:ALL?

Returned Format: [:MEASure:FREQuency] <result>; [PERiod] <result>; [PWIDth] <result>; [NWIDth] <result>; [RISetime] <result>; [FALLtime] <result>; [VAMPlitude] <result>; [VPP] <result>; [PREShoot] <result>; [OVERshoot] <result>; [DUTycycle] <result>; [VRMS] <result>; [VMAX] <result>; [VMIN] <result>; [VTOP] <result>; [VBASe] <result>; [VAverage] <result><NL>

Where:

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

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

Note

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

COMPare

COMPare

command/query

The `:MEASURE:COMPARE` command specifies the measurement and limits to be used for the measurement comparison (limit test). The first limit is the upper limit, the second is the lower limit.

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

The COMPARE query returns the current specification.

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

Where:

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

`<upper_limit>` ::= high limit value.

`<lower_limit>` ::= low limit value.



The suffix "HZ" can be used when setting the limits for FREQUENCY.

Example: `OUTPUT 707;":MEASURE:COMPARE RISETIME,10ns,2ns"`

Query Syntax: :MEASure:COMParE? <measurement>

Where:

<measurement> ::= {RISetime | FALLtime | FREQuency | PERiod | PWIDth |
NWIDth | VAMPLitude | VBASE | VTOP | VPP | VAverage | VMAX | VMIN | VRMS |
DUTycycle | DELAY}

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

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

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

```
OUTPUT 707;":MEASURE:SCRATCH" !clear measurements
OUTPUT 707;":MEASURE:FREQ"!Select measurement
OUTPUT 707;":MEASURE:COMPARE FREQ,1000HZ,10HZ"!Set measurement limits
OUTPUT 707;":MEASURE:LIMITTEST MEASURE"!Start test
OUTPUT 707;":RUN" !start acquisition
```

When a limit test failure occurs, bit 3 of the status byte is set to a 1.

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

- If `DELTA` is specified, the instrument returns the value of delta V (y) and delta T (x).
- If `START` is specified, the positions of the x1marker and y1marker are returned.
- If `STOP` is specified, the positions of the x2marker and y2marker are returned.

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

Query Syntax: `:MEASure:CURSor? {DELTA | START | STOP}`

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

Where:

`<time>` ::= delta time, x1marker time, or x2marker time
`<voltage>` ::= delta voltage, y1marker voltage, or y2marker voltage

Example:

```
OUTPUT 707;":MEAS:SOURCE CHAN1" !select measurement source
OUTPUT 707;":MEAS:CURSOR? START"
ENTER 707;Tme,V1t
PRINT Tme,V1t
```

DEFine

command/query

The :MEASURE:DEFINE command defines the setup for a measurement.

The DEFINE query returns the current setup.

Command Syntax: :MEASure:DEFine <measurement_spec>

Where:

```
<measurement_spec> ::= {DElay, <polarity>, <edge_number>, <level>,
<polarity>, <edge_number>, <level> | PWIDTH, <level> | NWIDTH, <level>}
```

Where:

```
<polarity> ::= {POSitive | NEGative}
<edge_number> ::= an integer, 1 to 4000 specifying a displayed edge
<level> ::= {MIDDLE | UPPer | LOWer}
```

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

This example sets the parameters for a time measurement from the first positive edge at the upper threshold level to the second negative edge at the middle threshold. If one source is specified, both parameters apply to that signal. If two sources are specified, the measurement is from the first positive edge on source 1 to the second negative edge on source 2.

Note 

The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

DEFine

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

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

Where:

<measurement_spec> ::= {DELAy, <polarity>, <edge_number>, <level>,
<polarity>, <edge_number>, <level> | PWIDth, <level> | NWIDth, <level>}

Where:

<polarity> ::= {POSitive | NEGative}
<edge_number> ::= 1 to 4000 (integer - NR1 format)
<level> ::= {MIDDLE | UPPER | LOWER}

Example: DIM Dfn\$[100]
OUTPUT 707;":MEASURE:DEFINE? DELAY"
ENTER 707;Dfn\$
PRINT Dfn\$

DElAy**command/query**

The :MEASURE:DELAY command determines the delay from the first specified edge on one source to the next specified edge on the same source, or to the first specified edge on another source.

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

If user-defined measurement specifications are selected, make sure the defined measurement is displayed.

The DELAY query returns the specified delay value.

Command Syntax: :MEASure:DElAy

Example: OUTPUT 707;":MEAS:DEL"

Query Syntax: :MEASure:DElAy?

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

Where:

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

Example: DIM D1y\$[50]
OUTPUT 707;":MEASURE:DELAY?"
ENTER 707;D1y\$
PRINT D1y\$

DESTination

DESTination

command/query

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

Source choices include the screen, any channel, or any function.

Destination choices are as follows:

- **Waveform memory:** when selected, the memory is overwritten each time a violation is found. Can only be selected when the source is CHANNEL or FUNCTION.
- **Multiple memory:** when selected, up to 665 records may be saved. If multiple failures occur and MEASURE:POSTfailure STOP is selected, 665 records are saved and the test stops. If multiple failures occur and MEASURE:POSTfailure CONTINUE is selected, the data records wrap around. Can only be selected when the source is CHANNEL or FUNCTION.
- **Pixel memory:** when selected, an accumulated save occurs. Can only be selected when the source is SCREEN.
- **Off:** when selected, no save occurs.

The DESTINATION query returns the destination currently selected for the specified source.

Command Syntax: :MEASURE:DESTINATION {{CHANNEL{ 1 | 2 | 3 | 4 } | FUNCTION{ 1 | 2 | 3 | 4 }}, {OFF | WMEMORY{ 1 | 2 | 3 | 4 } | MULTIPLE}} | {{SCREEN}, {OFF | PMEMORY{ 1 | 2 }}}

Example: OUTPUT 707;":MEAS:DEST CHAN1,MULT"



CHANNEL3 and CHANNEL4 are only available when the oscilloscope is an HP 54506B/54512B.

Query Syntax: :MEASure:DESTination {CHANne1{ 1 | 2 | 3 | 4 } | FUNction{ 1 | 2 | 3 | 4 }
| SCReen}?

Returned Format: [:MEASure:DESTination {CHANne1{ 1 | 2 | 3 | 4 } | FUNction{ 1 | 2 | 3 | 4 }
| SCReen}] {WMEMory{ 1 | 2 | 3 | 4 }|PMEMory{ 1 | 2 }|MULTiple| OFF}<NL>

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

DUTycycle

DUTycycle

command/query

The `:MEASURE:DUTYCYCLE` command places the instrument in the continuous measurement mode and starts the duty cycle measurement.

The `DUTYCYCLE` query measures and outputs the duty cycle of the signal specified by the `SOURCE` command. The signal must be displayed for the measurement to be made. The value returned for duty cycle is the ratio of the positive pulse width to the period.

The positive pulse width and the period of the specified signal are measured, then the duty cycle is calculated with the following formula:

$$\text{duty cycle} = \frac{\text{pulse width}}{\text{period}}$$

Command Syntax: `:MEASure:DUTycycle`

Example: `OUTPUT 707;":MEASURE:DUTYCYCLE"`

Query Syntax: `:MEASure:DUTycycle?`

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

Where:

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

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

ESTArt**command/query**

The :MEASURE:ESTART command positions the x1marker on the specified edge and slope of the displayed waveform. All edges must be displayed and are counted from the left edge of the display. The x1marker is positioned where y1marker intersects the waveform. The desired edge is specified by sending an integer value after the command name. If a positive integer is sent, the x1marker is placed on a positive-going waveform edge. If a negative integer is sent, the x1marker is placed on a negative-going waveform edge. If y1marker does not intersect the waveform as specified, the error message "Edges required not found" is displayed.

The ESTART query returns the currently specified edge.

Note 

The short form of this command, ESTA, does not follow the defined short form convention. The normal short form "EST" is the same for ESTART and ESTOP. Sending EST for the ESTART command produces an error.

Command Syntax: :MEASure:ESTArt <edge>

Where:

<edge> ::= -4000 to 4000 excluding 0 (if a positive value is sent the + sign may be omitted or a space may be used)

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

This example places the x1marker at the second displayed positive-going intersection of the waveform and y1marker.

ESArt

Query Syntax: :MEASure:ESArt?

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

Where:

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

Example: OUTPUT 707;":MEAS:ESArt?"
ENTER 707;Value
PRINT Value

ESTOp

command/query

The :MEASURE:ESTOP command positions the x2marker on the specified edge and slope of the displayed waveform. All edges must be displayed and are counted from the left edge of the display. The x2marker is positioned where y2marker intersects the waveform. The desired edge is specified by sending an integer value after the command name. If a positive integer is sent, the oscilloscope places the x2marker on a positive-going waveform edge. If a negative integer is sent, the x2marker is placed on a negative-going waveform edge.

If the y2marker does not intersect the waveform as specified, the error message "Edges required not found" is displayed.

The ESTOP query returns the currently specified edge.

Note

The short form of this command, ESTO, does not follow the defined short form convention. The normal short form "EST" is the same for ESTART and ESTOP. Sending EST for the ESTOP command produces an error.

Command Syntax: :MEASure:ESTOp <edge>

Where:

<edge> ::= -4000 to 4000 excluding 0 (if a positive value is sent the + sign may be omitted or a space may be used)

Example: OUTPUT 707;":MEAS:ESTOP -2"

This example places the x2marker at the second displayed negative-going intersection of the waveform and the y2marker.

ESTOp

Query Syntax: :MEASure:ESTOp?

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

Where:

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

Example: OUTPUT 707;":MEASURE:ESTOP?"
ENTER 707;Value
PRINT Value

FALLtime

command/query

The `:MEASURE:FALLTIME` command places the instrument in the continuous measurement mode and starts a fall time measurement.

The `FALLTIME` query measures and outputs the fall time of the first displayed falling (negative-going) edge. For highest measurement accuracy, set the sweep speed as fast as possible while leaving the falling edge of the waveform on the display. The fall time is determined by measuring the time at the upper threshold of the falling edge, then measuring the time at the lower threshold of the falling edge and calculating the fall time with the following formula:

fall time = time at lower threshold point – time at upper threshold point.

Command Syntax: `:MEASure:FALLtime`

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

Query Syntax: `:MEASure:FALLtime?`

Returned Format: `[[:MEASure:FALLtime] <value><NL>`

Where:

<value> ::= time in seconds between lower threshold and upper threshold voltage points (exponential - NR3 format)

Example:

```
DIM F11$[50]
OUTPUT 707;":MEASURE:FALLTIME?"
ENTER 707;F11$
PRINT F11$
```

FREQuency

FREQuency

command/query

The :MEASURE:FREQUENCY command places the instrument in the continuous measurement mode and starts a frequency measurement.

The FREQUENCY query measures and outputs the frequency of the first complete cycle on the screen. This command uses the 50% levels when Standard measurements are selected and the mid-threshold value when User-Defined measurements are selected.

The algorithm is:

```
if first edge on screen is rising
then
  frequency = 1/(time at second rising edge – time at first rising
  edge)
else
  frequency = 1/(time at second falling edge – time at first falling
  edge)
```

Command Syntax: :MEASure:FREQuency

Example: OUTPUT 707;":MEASURE:FREQ"

Query Syntax: :MEASURE:FREQuency?

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

Where:

<value> ::= frequency in Hertz (exponential - NR3 format)

Example: DIM Frq\$ [50]
OUTPUT 707;":MEASURE:FREQUENCY?"
ENTER 707;Frq\$
PRINT Frq\$

LIMittest
command

The `:MEASURE:LIMITTEST` command is used to start or stop a limit test. To perform a limit test, the following conditions must be satisfied:

- The measurement must be selected.
- The measurement comparison must be specified.
- The instrument must be acquiring data.

If `LIMITTEST` is sent with the `MEASURE` parameter, then the instrument starts the test. If the `OFF` parameter is sent, the test is stopped.

The `LTF` (limit test failure) bit of the status byte is set when a failure is found. This bit can be read with a `*STB?` or `LTER?` query, or by doing an HP-IB Serial Poll. The `*STB?` query does not clear the bit, and the results require some additional evaluation. Where as, the `LTER?` query returns the actual value of the bit and clears the bit so that the next limit test failure can be monitored. For greater speed, the `*SRE` command can be used in conjunction with an HP-IB Serial Poll. For more information, refer to the individual commands, queries, or functions.

Command Syntax: `:MEASure:LIMittest {MEASure | OFF}`

Example:

```

OUTPUT 707;":MEASURE:RISETIME"      !Select measurement
OUTPUT 707;":MEASURE:COMPARE RISETIME,10NS,2NS" !Specify measurement limits
OUTPUT 707;":RUN"                    !Acquire data
OUTPUT 707;":MEAS:LIM MEAS"

```

LOWer

LOWer

command/query

The `:MEASURE:LOWER` command sets the lower measurement threshold. This command sends a value to the instrument in the units selected with the `UNITS` command.



Set the measurement units with the `:MEASURE:UNITS` command prior to sending the lower threshold value.

The `LOWER` query returns the current setting of the lower measurement threshold.

Command Syntax: `:MEASure:LOWer <lower_threshold>`

Where:

`<lower_threshold>` ::= user defined lower threshold in percent or volts

Example:
OUTPUT 707;":MEASURE:MODE USER"
OUTPUT 707;":MEASURE:UNITS PERC"
OUTPUT 707;":MEASURE:LOWER 47"

Query Syntax: `:MEASure:LOWer?`

Returned Format: `[:MEASure:LOWer] <lower_threshold><NL>`

Where:

`<lower_threshold>` ::= user defined lower threshold in percent or volts

Example:
DIM Lwr\$[50]
OUTPUT 707;":MEAS:LOW?"
ENTER 707;Lwr\$
PRINT Lwr\$

MODE**command/query**

The **:MEASURE:MODE** command selects standard or user-defined definitions and thresholds for voltage and time measurements.

The **MODE** query returns the current measurement mode setting.

Command Syntax: `:MEASure:MODE {STANdard | USER}`

Example: `OUTPUT 707;":MEAS:MODE STAN"`

Query Syntax: `:MEASure:MODE?`

Returned Format: `[:MEASure:MODE] {STANdard | USER}<NL>`

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



The **:MEASure:MODE** command must be set to **USER** before **:LOWer**, **:UPPer**, **:UNITs**, or **:DEFine** **:MEASure** commands will be in effect.

NWIDth

NWIDth

command/query

The :MEASURE:NWIDTH command places the instrument in the continuous measurement mode and starts an NWIDTH measurement.

If Standard measurements are selected, the NWIDTH query measures and outputs the width of the first negative pulse on the screen using the 50% levels.

If User Defined measurements are selected, the measurement is made at the threshold value currently selected using the :MEASure:DEFine, :UPPer, and :LOWer commands.

The algorithm is:

```
if the first edge on screen is rising
then
  width = (time at second rising edge – time at first falling edge)
else
  width = (time at first rising edge – time at first falling edge)
```

Command Syntax: :MEASure:NWIDth

Example: OUTPUT 707;":MEAS:NWIDTH"

Query Syntax: :MEASure:NWIDth?

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

Where:

<value> ::= negative pulse width in seconds (exponential - NR3 format)

Example:

```
DIM Nwd$ [50]
OUTPUT 707;":MEASURE:NWIDTH?"
ENTER 707;Nwd$
PRINT Nwd$
```

OVERshoot

command/query

The :MEASURE:OVERSHOOT command places the instrument in the continuous measurement mode and selects the overshoot measurement.

The OVERSHOOT query measures and outputs the overshoot of the first displayed edge of the selected signal. Overshoot is measured with the following algorithm:

```

if the first edge on screen is rising
then
  overshoot = (Vmax – Vtop)/Vamplitude
else
  overshoot = (Vbase – Vmin)/Vamplitude

```

Command Syntax: :MEASure:OVERshoot

Example: OUTPUT 707;":MEAS:OVER"

Query Syntax: :MEASure:OVERshoot?

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

Where:

<value> ::= ratio of overshoot to Vamplitude (exponential - NR3 format)

Example: DIM Ovr\$[50]
 OUTPUT 707;":MEASURE:OVERSHOOT?"
 ENTER 707;Ovr\$
 PRINT Ovr\$

PERiod

PERiod

command/query

The :MEASURE:PERIOD command places the instrument in the continuous measurement mode and selects the period measurement.

The PERIOD query measures and outputs the period of the first complete cycle on the screen. The period is measured at the 50% point when standard measurements are selected and at the mid-threshold voltage level of the waveform when user-defined measurements are selected.

The algorithm for this measurement is:

```
if the first edge on screen is rising
then
    period = (time at second rising edge – time at first rising edge)
else
    period = (time at second falling edge – time at first falling edge)
```

Command Syntax: :MEASure:PERiod

Example: OUTPUT 707;":MEAS:PERIOD"

Query Syntax: :MEASure:PERiod?

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

Where:

<value> ::= waveform period in seconds (exponential - NR3 format)

Example: DIM Prd\$[50]
OUTPUT 707;":MEASURE:PERIOD?"
ENTER 707;Prd\$
PRINT Prd\$

POSTfailure

command/query

The :MEASURE:POSTFAILURE command specifies what the instrument will do after a violation is found by the limit test.

- If STOP is selected, the instrument stops the limit test when the first violation is found. If multiple memory is selected as the destination, then the test will terminate after all available memory space is filled.
- If CONTINUE is selected and a violation is found, the violation is written to the desired location, and the instrument continues to look for another violation. If a waveform memory is selected as the destination, then all subsequent violations will overwrite the previous violation. If multiple memory is selected as the destination, then the data records will wrap around.

The destination can be specified with the :MEASURE:DESTINATION command.

The POSTFAILURE query returns the current selection.

Command Syntax: :MEASure:POSTfailure {CONTInue | STOP}

Example: OUTPUT 707;":MEAS:POST"

Query Syntax: :MEASure:POSTfailure?

Returned Format: [:MEASure:POSTfailure] {CONTInue | STOP}<NL>

Example: DIM Pf\$[50]
OUTPUT 707;":MEASURE:POSTFAILURE?"
ENTER 707;Pf\$
PRINT Pf\$

PREShoot

PREShoot

command/query

The :MEASURE:PRESHOOT command places the instrument in the continuous measurement mode and starts the preshoot measurement.

The PRESHOOT query measures and outputs the preshoot of the first displayed edge of the selected signal. Preshoot is measured with the following algorithm:

```
if the first edge on screen is rising
then
  preshoot = (Vbase - Vmin)/Vamplitude
else
  preshoot = (Vmax - Vtop)/Vamplitude
```

Command Syntax: :MEASure:PREShoot

Example: OUTPUT 707;":MEASURE:PRES"

Query Syntax: :MEASure:PREShoot?

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

Where:

<value> ::= ratio of preshoot to Vamplitude (exponential - NR3 format)

Example: DIM Prs\$[50]
OUTPUT 707;":MEASURE:PRESHOOT?"
ENTER 707;Prs\$
PRINT Prs\$

PWIDth

command/query

The :MEASURE:PWIDTh command places the instrument in the continuous measurement mode and starts the Pwidth measurement.

The PWIDTh query measures and outputs the width of the first displayed positive pulse. Pulse width is measured at the 50% voltage level with standard measurements selected. If User Defined measurements are selected, the measurement is made at the threshold value currently selected using the :MEASure:DEFine, :UPPer, and :LOWer commands.

The algorithm for this measurement is:

```

if the first edge on screen is falling
then
    width = (time at second falling edge – time at first rising edge)
else
    width = (time at first falling edge – time at first rising edge)

```

Command Syntax: :MEASure:PWIDth

Example: OUTPUT 707;":MEAS:PWIDTh"

Query Syntax: :MEASure:PWIDth?

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

Where:

<value> ::= width of positive pulse in seconds (exponential - NR3 format)

Example: DIM Pwd\$ [50]
 OUTPUT 707;":MEASURE:PWIDTh?"
 ENTER 707;PwD\$
 PRINT PwD\$

RESults

RESults

query

The `:MEASURE:RESULTS` query returns the currently active measurements. If statistics are on, the minimum, maximum, and average is returned for each measurement. If the limit test is on and `POSTFAILURE` is set to `CONTINUED`, the pass ratio is returned instead of the average.

If the number of measurements returned is 0, then no measurements are returned.

Query Syntax: `:MEASure:RESults?`

Returned Format: `[[:MEASure:RESults] <No. of Meas>[:<measurement>]...<NL>`

Where:

`<No. of Meas> ::=` number of measurements displayed on the screen, 0 through 8 (integer - NRI format)
`<measurement> ::=` measurement_name measurement_result.

Example:

```
DIM Mr$[100]
OUTPUT 707;":MEASURE:RESULTS?"
ENTER 707;Mr$
PRINT Mr$
```

Note 

measurement_results returned are dependent on the current `:MEASure:STATistics` selection. When `ON`, the current, minimum, maximum, and average (or pass) measurements are returned. When `OFF`, only the current measurement is returned.

RISetime

command/query

The :MEASURE:RISETIME command places the instrument in the continuous measurement mode and starts a rise time measurement.

The RISETIME query measures and outputs the rise time of the first displayed rising (positive-going) edge. For maximum measurement accuracy set the sweep speed as fast as possible while leaving the leading edge of the waveform on the display. The rise time is determined by measuring the time at the lower threshold of the rising edge and the time at the upper threshold of the rising edge, then calculating the rise time with the following formula:

$$\text{rise time} = (\text{time at upper threshold point} - \text{time at lower threshold point})$$

Command Syntax: :MEASure:RISet ime

Example: OUTPUT 707;":MEAS:RIS"

Query Syntax: :MEASure:RISet ime?

Returned Format: [:MEASure:RISet ime] <value><NL>

Where:

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

Example: DIM Rs\$[50]
OUTPUT 707;":MEAS:RIS?"
ENTER 707;Rs\$
PRINT Rs\$

SCRatch

SCRatch

command

The `:MEASURE:SCRATCH` command clears the measurement results from the oscilloscope display.

Command Syntax: `:MEASure:SCRatch`

Example: `OUTPUT 707;":MEASURE:SCRATCH"`

SOURce

command/query

The `:MEASURE:SOURCE` command selects the sources for the measurements. The specified source becomes the source for the MEASURE subsystem commands.

Two sources can be specified with this command. All measurements except DELAY are made on the first specified source. The DELAY measurement uses two sources if two have been specified. If only one source is specified, the DELAY measurement uses that source for both of its parameters.

The SOURCE query returns the current source selection. If the specified sources are different, both are returned. Otherwise, one source is returned.

Command Syntax: `:MEASure:SOURce <source1>[,<source2>]`

Where:

`<source1>` and `<source2>` ::= {CHANnel{1 | 2 | 3 | 4} | FUNCTION{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

Example: `OUTPUT 707;":MEASURE:SOURCE CHANNEL1, WMEMORY1"`

Query Syntax: `:MEASure:SOURce?`

Returned Format: `[:MEASure:SOURce] <source1>[,<source2>]<NL>`

Where:

`<source1>` and `<source2>` ::= {CHANnel{1 | 2 | 3 | 4} | FUNCTION{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

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



CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

The `:MEASURE:STATISTICS` command turns the statistics function on and off. When this mode is on, and the measurements are in the continuous mode, the minimum, maximum, average, and current measurements are shown as the active measurements. If a `RESULTS` query is executed, all of the displayed data is returned to the controller.

The `STATISTICS` query returns the current mode.



"Average" is replaced by "pass ratio" when the "limit test" is selected and "after failure" is set to continue. Pass ratio lists the percentage of times a certain test passed.

Command Syntax: `:MEASure:STATistics {{ON | 1} | {OFF | 0}}`

Example: `OUTPUT 707;":MEASURE:STAT ON"`

Query Syntax: `:MEASure:STATistics?`

Returned Format: `[:MEASure:STATistics] {1 | 0} <NL>`

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

TDELta

query

The :MEASURE:TDELTA query returns the time difference between the x1 and x2 time markers:

$$Tdelta = x1marker - x2marker$$

This is the same value that is displayed as delta x on the front panel screen.

Note

This command is identical to the MARKER:XDELTA command, and is provided for compatibility with other oscilloscopes.

Query Syntax: :MEASure:TDELta?

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

Where:

<value> ::= difference between x1 and x2 markers (exponential - NR3 format)

Example:

```
DIM Td1$[50]
OUTPUT 707;":MEASURE:TDELTA?"
ENTER 707;Td1$
PRINT Td1$
```

TMAX

TMAX

query

The :MEASURE:TMAX query returns the time at which the first maximum voltage occurred.

Query Syntax: :MEASure:TMAX?

Returned Format: [:MEASure:TMAX] <time><NL>

Where:

<time> ::= time at maximum voltage.

Example:

```
DIM Tmx$[50]
OUTPUT 707;" :MEASURE:TMAX?"
ENTER 707;Tmx$
PRINT Tmx$
```

TMIN**query**

The :MEASURE:TMIN query returns the time at which the first minimum voltage occurred.

Query Syntax: :MEASure:TMIN?

Returned Format: [:MEASure:TMIN] <time><NL>

Where:

<time> ::= time at minimum voltage.

Example:

```
DIM Tmn$[50]
OUTPUT 707;":MEAS:TMIN?"
ENTER 707;Tmn$
PRINT Tmn$
```

TSTArt

TSTArt

command/query

The :MEASURE:TSTART command moves the x1marker to the specified time with respect to the trigger time.



This command is identical to the MARKER:X1POSITION command, and is provided for compatibility with other oscilloscopes.

The TSTART query returns the time at the x1marker.



The short form of this command, TSTA, does not follow the defined short form convention. The normal short form "TST" is the same for TSTART and TSTOP. Sending TST for the TSTART command produces an error.

Command Syntax: :MEASure:TSTArt <start marker time>

Where:

<start marker time> ::= time at x1marker in seconds

Example: OUTPUT 707;":MEASURE:TSTART 30 NS"

Query Syntax: :MEASure:TSTArt?

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

Where:

<value> ::= time at x1marker in seconds (exponential - NR3 format)

Example:
DIM Tst\$
OUTPUT 707;":MEASURE:TSTART?"
ENTER 707;Tst\$
PRINT Tst\$

TSTOp

command/query

The :MEASURE:TSTOP command moves the x2marker to the specified time with respect to the trigger time.



This command is identical to the MARKER:X2POSITION command, and is provided for compatibility with other oscilloscopes.

The TSTOP query returns the time at the x2marker.



The short form of this command, TSTO, does not follow the defined short form convention. The normal short form "TST" is the same for TSTART and TSTOP. Sending TST for the TSTOP command produces an error.

Command Syntax: :MEASure:TSTOp <stop marker time>

Where:

<stop marker time> ::= time at x2marker in seconds

Example: OUTPUT 707;":MEAS:TSTOP 40 NS"

Query Syntax: :MEASure:TSTOp?

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

Where:

<value> ::= time at x2marker in seconds (exponential - NR3 format)

Example: DIM Tst\$[50]
OUTPUT 707;":MEASURE:TSTOP?"
ENTER 707;Tst\$
PRINT Tst\$

When the :MEASURE:TVOLT query is sent, the displayed signal is searched for the defined voltage level and transition. The time interval between the trigger event and this defined occurrence is returned as the response to this query.

The voltage can be specified as a negative or positive voltage. To specify a negative voltage, use a minus (-) sign. The sign of the slope selects a rising (+) or falling (-) edge.

The magnitude of occurrence defines the occurrence to be reported. For example, +3 returns the time for the third time the waveform crosses the specified voltage level in the positive direction. Once this voltage crossing is found, the oscilloscope outputs the time at that crossing in seconds, with the trigger point (time zero) as the reference.

If the specified crossing cannot be found, the oscilloscope outputs +9.99999E+37. This value is returned if the waveform does not cross the specified voltage, or if the waveform does not cross the specified voltage for the specified number of times in the specified direction.

Query Syntax: :MEASure:TVOLT? <voltage>, <slope><occurrence>

Where:

<voltage> ::= positive or negative voltage level that the waveform must cross.

<slope> ::= direction of the waveform when <voltage> is crossed - rising (sp or +) or falling (-)

<occurrence> ::= number of crossings to be reported (if one - the first crossing is reported, if two - the second crossing is reported, etc.)

Returned Format: [:MEASure:TVOLT] <time><NL>

Where:

<time> ::= time in seconds of specified voltage crossing (exponential - NR3 format)

Example: DIM Tv1t\$[50]
OUTPUT 707;":MEASURE:TVOLT? -.250,+3"
ENTER 707;Tv1t\$
PRINT Tv1t\$

UNITS

UNITS

command/query

The :MEASURE:UNITS command sets the measurement threshold units when the user defined measurement mode is selected. The UNITS can be set to PERCENT or VOLTS.

The UNITS query returns the currently selected units.

Command Syntax: :MEASure:UNITs {PERCent | VOLTs}

Example: OUTPUT 707;":MEASURE:MODE USER"
OUTPUT 707;":MEASURE:UNITS PERCENT"

Query Syntax: :MEASure:UNITs?

Returned Format: [:MEASure:UNITs] {PERCent | VOLTs}<NL>

Example: DIM Unt\$[50]
OUTPUT 707;":MEASURE:UNITS?"
ENTER 707;Unt\$
PRINT Unt\$



The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

UPPer

command/query

The :MEASURE:UPPER command sets the upper measurement threshold.



Set the measurement units with the :MEASURE:UNITS command prior to sending the upper threshold value.

The UPPER query returns the value of the upper measurement threshold.

Command Syntax: :MEASure:UPPer <upper_threshold>

Where:

<upper_threshold> ::= upper threshold value in percent or volts

Example: OUTPUT 707;":MEAS:UPPER 90"

Query Syntax: :MEASure:UPPer?

Returned Format: [:MEASure:UPPer] <upper_threshold><NL>

Where:

<upper_threshold> ::= upper threshold value in percent or volts (exponential - NR3 format)

Example:
DIM Up\$[50]
OUTPUT 707;":MEAS:UPP?"
ENTER 707;Up\$
PRINT Up\$



The :MEASure:MODE command must be set to USER before :LOWer, :UPPer, :UNITs, or :DEFine :MEASure commands will be in effect.

VACRms

VACRms

command/query

The :MEASURE:VACRMS command and query perform the same functions as the VRMS command and query.

The :MEASURE:VACRMS command places the instrument in the continuous measurement mode and starts an ac rms voltage measurement.

The VACRMS query measures and outputs the AC RMS voltage of the selected waveform. The AC RMS voltage is measured on the first cycle of the displayed signal. If a complete cycle is not present, the oscilloscope computes the RMS value on all displayed data points.

Note

The VACRMS measurement is an AC RMS measurement. This means that the average value of the waveform is subtracted from each data point before the RMS voltage is computed.

Command Syntax: :MEASure:VACRms

Example: OUTPUT 707;":MEAS:VACRMS"

Query Syntax: :MEASure:VACRms?

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

Where:

<value> ::= calculated ac rms voltage (exponential - NR3 format)

Example:
DIM Vrm\$[50]
OUTPUT 707;":MEASURE:VACRMS?"
ENTER 707;Vrm\$
PRINT Vrm\$

VAMPlitude

command/query

The :MEASURE:VAMPLITUDE command places the instrument in the continuous measurement mode and starts a Vamplitude measurement.

The VAMPLITUDE query returns the difference between the top and base voltage of the displayed signal. The VAMPLITUDE value is not normally the same as the Vp-p value when the input signal is a pulse.

The Vamplitude value is calculated with the formula:

$$\text{Vamplitude} = V_{\text{top}} - V_{\text{base}}$$

Command Syntax: :MEASure:VAMPitude

Example: OUTPUT 707;":MEAS:VAMP"

Query Syntax: :MEASure:VAMPitude?

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

Where:

<value> ::= difference between top and base voltages (exponential - NR3 format)

Example: DIM Vmp\$[50]
OUTPUT 707;":MEASURE:VAMPLITUDE?"
ENTER 707;Vmp\$
PRINT Vmp\$

VAVerage

VAVerage

command/query

The `:MEASURE:VAVERAGE` command places the instrument in the continuous measurement mode and starts a Vaverage measurement.

The `VAVERAGE` query measures the average voltage of the first cycle of the displayed signal. If a complete cycle is not present, the oscilloscope averages all data points.

Command Syntax: `:MEASure:VAVerage`

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

Query Syntax: `:MEASure:VAVerage?`

Returned Format: `[[:MEASure:VAVerage] <average_voltage><NL>`

Where:

`<average_voltage>` ::= calculated average voltage (exponential - NR3 format)

Example:

```
DIM Vv$ [50]
OUTPUT 707;":MEAS:VAV?"
ENTER 707;Vv$
PRINT Vv$
```

VBASe

command/query

The `:MEASURE:VBASE` command places the instrument in the continuous measurement mode and starts a Vbase measurement.

The VBASE query measures and outputs the voltage value at the base of the waveform. The base voltage of a pulse is normally not the same as the minimum value.

Command Syntax: `:MEASure:VBASe`

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

Query Syntax: `:MEASure:VBASe?`

Returned Format: `[[:MEASure:VBASe] <base_voltage><NL>`

Where:

<base_voltage> ::= voltage at base of selected waveform (exponential - NR3 format)

Example:

```
DIM Vbs$[50]
OUTPUT 707;":MEASURE:VBASe?"
ENTER 707;Vbs$
PRINT Vbs$
```

VDCRms

VDCRms

command/query

The `:MEASURE:VDCRMS` command places the instrument in the continuous measurement mode and starts a dc rms voltage measurement.

The `VDCRMS` query measures and outputs the RMS voltage of the selected waveform. The RMS voltage is measured on the first cycle of the displayed signal. If a complete cycle is not present, the oscilloscope computes the RMS value on all displayed data points.



The `VDCRMS` measurement is a true RMS measurement.

Command Syntax: `:MEASure:VDCRms`

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

Query Syntax: `:MEASure:VDCRms?`

Returned Format: `[[:MEASure:VDCRms] <value><NL>`

Where:

`<value>` ::= calculated dc rms voltage (exponential - NR3 format)

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

VDELta

query

The :MEASURE:VDELTA query outputs the voltage difference between the y1marker and y2marker. No measurement is made when the VDELTA query is received by the oscilloscope. The delta voltage value that is output is the current value. This is the same value as the front panel delta y value.

$$\text{VDELTA} = \text{y2marker} - \text{y1marker 1}$$

Note

This command is identical to the MARKER:YDELTA command, and is provided for compatibility with other oscilloscopes.

Query Syntax: :MEASure:VDELta?

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

Where:

<value> ::= delta V value in volts (exponential - NR3 format)

Example:

```
DIM Vd1$[50]
OUTPUT 707;":MEAS:VDELTA?"
ENTER 707;Vd1$
PRINT Vd1$
```

VFIFty

VFIFty

command

The :MEASURE:VFIFTY command finds the top and base values of the specified waveforms, then places the y1 and y2 markers at the 50% voltage point on the specified sources.

If only one source has been specified with the source command, the VFIFty command sets both y1 and y2 markers to the 50% voltage level on that source.

If two sources are specified with the source command, the y1 marker is set to the 50% level of the first specified source and the y2 marker is set to the 50% level of the second specified source.

There is no query form of this command.

Command Syntax: :MEASure:VFIFTy

Example: OUTPUT 707;":MEASURE:VFIFTY"

VMAX**command/query**

The `:MEASURE:VMAX` command places the instrument in the continuous measurement mode and starts a Vmax measurement.

The `VMAX` query measures and outputs the absolute maximum voltage present on the selected waveform.

Command Syntax: `:MEASure:VMAX`

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

Query Syntax: `:MEASure:VMAX?`

Returned Format: `[[:MEASure:VMAX] <value><NL>`

Where:

`<value>` ::= maximum voltage of selected waveform (exponential - NR3 format)

Example:

```
DIM Vmx$[50]
OUTPUT 707;":MEASURE:VMAX?"
ENTER 707;Vmx$
PRINT Vmx$
```

VMIN

VMIN

command/query

The `:MEASURE:VMIN` command places the instrument in the continuous measurement mode and starts a VMIN measurement.

The VMIN query measures and outputs the absolute minimum voltage present on the selected waveform.

Command Syntax: `:MEASure:VMIN`

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

Query Syntax: `:MEASure:VMIN?`

Returned Format: `[:MEASure:VMIN] <value><NL>`

Where:

`<value>` ::= minimum voltage value of the selected waveform
(exponential - NR3 format)

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

VPP

command/query

The :MEASURE:VPP command places the instrument in the continuous measurement mode and starts a VPP measurement.

The VPP query measures the maximum and minimum voltages for the selected source, then calculates the peak-to-peak voltage and outputs that value. The peak-to-peak voltage (Vpp) is calculated with the following formula:

$$V_{pp} = V_{max} - V_{min}$$

Vmax and Vmin are the maximum and minimum voltages present on the selected source.

Command Syntax: :MEASure:VPP

Example: OUTPUT 707;":MEAS:VPP"

Query Syntax: :MEASure:VPP?

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

Where:

<value> ::= voltage peak to peak (exponential - NR3 format)

Example: DIM Vp\$[50]
OUTPUT 707;":MEAS:VPP?"
ENTER 707;Vp\$
PRINT Vp\$

The `:MEASURE:VRELATIVE` command moves the `y1` and `y2` markers to the specified percentage points of their last established position. The last established position is not necessarily on the currently displayed waveform.

For example, after a `:MEASURE:VAMPLITUDE?` query is sent, the `y1` marker is located at the base (0%) of the signal and the `y2` marker is at the top (100%) of the signal. If the `VRELATIVE 10` command is executed, the `y1` marker is moved to the 10% level and the `y2` marker to the 90% level of the signal.

Any value between 0% and 100% can be used. If `VRELATIVE 0` is sent, the `y` markers are not moved because the command indicates 0% movement from the current position.

As an example, when the following values are sent, the markers are moved to the following percentage values of their current position:

- 10 moves `y1` marker to 10% and `y2` marker to 90%
- 20 moves `y1` marker to 20% and `y2` marker to 80%
- 50 moves both markers to 50%
- 80 moves `y1` marker to 20% and `y2` marker to 80%
- 90 moves `y1` marker to 10% and `y2` marker to 90%

The starting position of the markers must be known for this command to be meaningful. The markers can be set to a known position on the selected waveform using the `:MEASURE:VAMPLITUDE?` query.

The `VRELATIVE` query returns the current relative position of `VMarker2`, which is always in the range of 50% through 100%.



The `VRELATIVE` command does not affect the upper and lower thresholds selected by the `UPPER` and `LOWER` commands.

Command Syntax: :MEASure:VRELative <percent>

Where:

<percent> ::= {0 through 100}

Example: OUTPUT 707;":MEASURE:VRELATIVE 20"

Query Syntax: :MEASure:VRELative?

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

Where:

<value> ::= Vmarker 2 relative position in percent {50 through 100}
(integer - NR1 format)

Example: DIM Vr1\$[50]
OUTPUT 707;":MEAS:VREL?"
ENTER 707;Vr1\$
PRINT Vr1\$

VRMS

VRMS

command/query

The `:MEASURE:VRMS` command and query perform the same functions as the `VACRMS` command and query.

The `:MEASURE:VRMS` command places the instrument in the continuous measurement mode and starts an ac rms voltage measurement.

The VRMS query measures and outputs the RMS voltage of the selected waveform. The RMS voltage is measured on the first cycle of the displayed signal. If a complete cycle is not present, the oscilloscope computes the RMS value on all displayed data points.



The VRMS measurement is an AC RMS measurement. This means that the average value of the waveform is subtracted from each data point before the RMS voltage is computed.

Command Syntax: `:MEASure:VRMS`

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

Query Syntax: `:MEASure:VRMS?`

Returned Format: `[[:MEASure:VRMS] <value><NL>`

Where:

`<value>` ::= calculated ac rms voltage (exponential - NR3 format)

Example:

```
DIM Vrm$ [50]
OUTPUT 707;":MEASURE:VRMS?"
ENTER 707;Vrm$
PRINT Vrm$
```

VSTArt

command/query

The :MEASURE:VSTART command moves the y1marker to the specified voltage. The values are limited to the currently defined channel, function, or memory range.



This command is identical to the MARKER:Y1POSITION command, and is provided for compatibility with other oscilloscopes.

The VSTART query returns the current voltage level of the y1marker.



The short form of this command, VSTA, does not follow the defined short form convention. The normal short form "VST" is the same for VSTART and VSTOP. Sending VST for the VSTART command produces an error.

Command Syntax: :MEASure:VSTArt <voltage>

Where:

<voltage> ::= voltage value for y1marker

Example: OUTPUT 707;":MEAS:VSTA -10MV"

Query Syntax: :MEASure:VSTArt?

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

Where:

<value> ::= voltage at y1marker (exponential - NR3 format)

Example:
 DIM Vst\$[50]
 OUTPUT 707;":MEASURE:VSTART?"
 ENTER 707;Vst\$
 PRINT Vst\$

VSTOp

VSTOp

command/query

The `:MEASURE:VSTOP` command move sthe y2marker to the specified voltage.



This command is identical to the `MARKER:Y2POSITION` command, and is provided for compatibility with other oscilloscopes.

The `VSTOP` query returns the current voltage level of the y2marker.



The short form of this command, `VSTO`, does not follow the defined short form convention. The normal short form "VST" is the same for `VSTART` and `VSTOP`. Sending `VST` for the `VSTOP` command produces an error.

Command Syntax: `:MEASure:VSTOp <voltage>`

Where:

`<voltage>` ::= voltage value for y2marker

Example: `OUTPUT 707;":MEAS:VSTO -100MV"`

Query Syntax: `:MEASure:VSTOp?`

Returned Format: `[[:MEASure:VSTOp] <value><NL>`

Where:

`<value>` ::= voltage at y2marker (exponential - NR3 format)

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

VTIMe**query**

The :MEASURE:VTIME query returns the voltage at a specified time. The specified time must be on screen and is referenced to the trigger event.

Query Syntax: :MEASure:VTIME? <time>

Where:

<time> ::= displayed time from trigger in seconds

Returned Format: [:MEASure:VTIME] <voltage><NL>

Where:

<voltage> ::= voltage at specified time (exponential - NR3 format)

Example:

```
DIM Vtm$[50]
OUTPUT 707;":MEASURE:VTIME? .001"
ENTER 707;Vtm$
PRINT Vtm$
```

VTOP

VTOP

command/query

The `:MEASURE:VTOP` command places the instrument in the continuous measurement mode and starts a Vtop measurement.

The VTOP query returns the voltage at the top of a waveform.

Command Syntax: `:MEASure:VTOP`

Example: `OUTPUT 707;":MEASURE:VTOP"`

Query Syntax: `:MEASure:VTOP?`

Returned Format: `[[:MEASure:VTOP] <value><NL>`

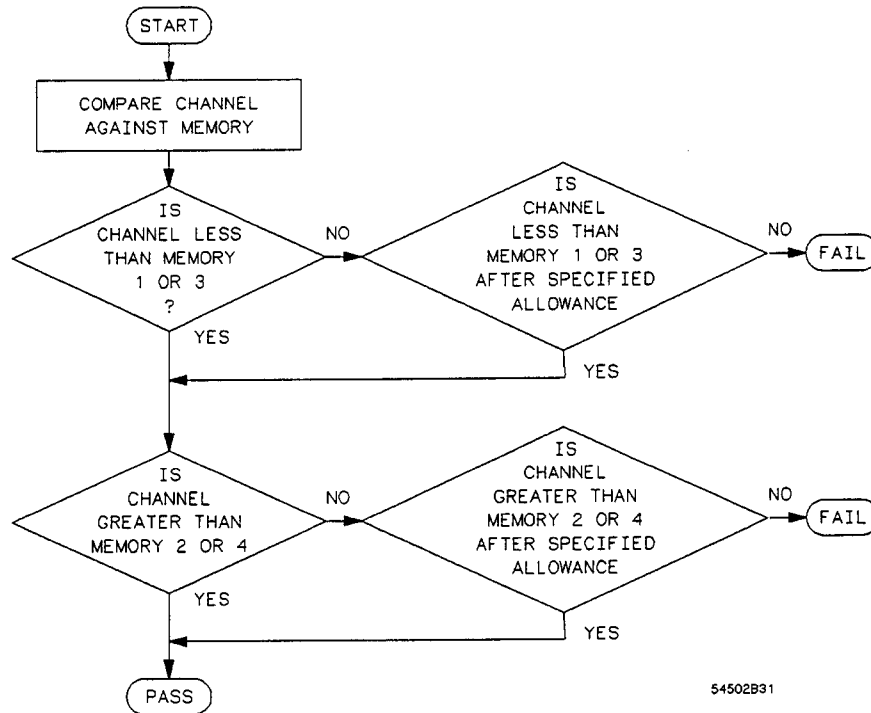
Where:

`<value>` ::= voltage at the top of the waveform (exponential - NR3 format)

Example:

```
DIM Vtp$[50]
OUTPUT 707;":MEASURE:VTOP?"
ENTER 707;Vtp$
PRINT Vtp$
```

When a waveform comparison test fails, a second test is performed on the waveform data and memory against a specified tolerance or allowance, called a Waveform Comparison Test. The chart in figure 15-2 lists the flow for the waveform comparison test.



54502B91

15-2. Waveform Compare Flowchart

WCOMpare:ALLOWance

WCOMpare:ALLOWance

command/query

The `:MEASURE:WCOMPARE:ALLOWANCE` command allows you to specify an allowance representing the allowable number of divisions that the waveform comparison test can deviate and still pass. The allowance value that you specify over the bus is automatically rounded to the nearest 0.025 divisions. The chart in figure 14-3 lists the flow for the waveform comparison test.

Command Syntax: `:MEASURE:WCOMPARE:ALLOWANCE <allow_value>`

Where:

`<allow_value>` ::= real number representing number of divisions of allowance (0.0 to 8.0)

Example: `OUTPUT 707;":MEAS:WCOM:ALLOWANCE 0.0"`

Query Syntax: `:MEASURE:WCOMPARE:ALLOWANCE?`

Returned Format: `[[:MEASURE:WCOMPARE:ALLOWANCE] {<allow_value>}<NL>`

Where:

`<allow_value>` ::= real number representing number of divisions of allowance (0.0 to 8.0)

Example:
`DIM Wc1$[50]
OUTPUT 707:":MEAS:WCOM:ALL?"
ENTER 707;Wc1$
PRINT Wc1?`

WCOMpare:COMParE

command/query

The :MEASURE:WCOMPARE:COMPARE command selects the channel and memory pair to be compared by the waveform comparison test. Specifying wmemory1 or wmemory2 selects both wmemory1 (m1) and wmemory2 (m2) as the mask pair that the channel will be compared to. Specifying wmemory3 or wmemory4 selects both wmemory3 (m3) and wmemory4 (m4) as the mask pair that the channel will be compared to.

Command Syntax: :MEASure:WCOMpare:COMParE CHANnel<chan_num> | FUNCtion<func_num>, WMEMory<wmem_num>

Where:

<chan_num> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
<func_num> ::= 1 through 4
<wmem_num> ::= 1 through 4

Example: OUTPUT 707;":MEAS:WCOM:COMP CHAN1, WMEM1"

Query Syntax: :MEASure:WCOMpare:COMParE?

Returned Format: [:MEASure:WCOMpare] {<chan_num> | <func_num>, <wmem_num>} <NL>

Where:

<chan_num> ::= 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B)
<func_num> ::= 1 through 4
<wmem_num> ::= 1 through 4

Example: DIM Wcc\$[50]
OUTPUT 707;":MEAS:WCOM:COMP? "
ENTER 707;Wcc\$
PRINT Wcc\$

WCOMpare:DESTination

WCOMpare:DESTination

command/query

The :MEASURE:WCOMPARE:DESTINATION command specifies the source and destination to be used when a waveform comparison violation is found. Data may be stored to one of the waveform memories, but not to the pair of memories used for the mask in the comparison test. Data can also be stored to one of the pixel memories, the multiple memories, or discarded by specifying OFF.

Source choices include the screen, any channel, or any function.

Destination choices are as follows:

- **Waveform memory:** when selected, the memory is overwritten each time a violation is found. Can only be selected when the source is CHANnel or FUNCtion.
- **Multiple memory:** when selected, up to 665 records may be saved. If multiple failures occur and MEASure:POSTfailure STOP is selected, records are saved until available memory is filled and the test stops. If multiple failures occur and MEASure:POSTfailure CONTInue is selected, the data records wrap around. Can only be selected when the source is CHANnel or FUNCtion.
- **Pixel memory:** when selected, an accumulated save occurs. Can only be selected when the source is SCReen.
- **Off:** when selected, no save occurs.

Note

If the waveform memory selected for the destination is the same as the waveform memories used for the mask, the destination is set to OFF. Otherwise, failure information would be written into the mask.

The DESTINATION query returns the destination currently selected for the specified source.


Command Syntax: :MEASure:WCOMpare:DESTination {{CHANnel{ 1 | 2 | 3 | 4 } | FUNction{ 1 | 2 | 3 | 4 }},{OFF | WMEMory{ 1 | 2 | 3 | 4 } | MULTiple}} | {{SCReen},{OFF|PMEMory{ 1 | 2 }}}}

Example: OUTPUT 707;":MEAS:WCOM:DEST CHAN1,MULT"

Query Syntax: :MEASure:WCOMpare:DESTination? {CHANnel{ 1 | 2 | 3 | 4 } | FUNction{ 1 | 2 | 3 | 4 } | SCReen}

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

Example: DIM Dst\$[50]
OUTPUT 707;":MEAS:WCOM:DEST? CHAN1"
ENTER 707;Dst\$
PRINT Dst\$

Note  CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

WCOMpare:POSTfailure

WCOMpare:POSTfailure

command/query

The `:MEASURE:WCOMPARE:POSTFAILURE` command specifies what will occur after a violation has been found by the waveform comparison test.

- If `STOP` is selected, the instrument stops the waveform comparison test and acquisition when the first violation is found. If multiple memory is selected as the destination, then the test will terminate after all available memory space is filled.
- If `CONTINUE` is selected and a violation is found, the violation is written to the desired location, and the instrument continues to look for another violation. If a waveform memory is selected as the destination, then all subsequent violations will overwrite the previous violation. If multiple memory is selected as the destination, then the data records will wrap around.

The destination can be specified with the `:MEASURE:WCOMPARE:DESTINATION` command.

Command Syntax: `:MEASure:WCOMpare:POSTfailure {CONTInue | STOP}`

Example: `OUTPUT 707;":MEASURE:WCOMPARE:POST CONT"`

Query Syntax: `:MEASure:WCOMpare:POSTfailure?`

Returned Format: `[[:MEASure:WCOMpare:POSTfailure] {CONTInue | STOP}<NL>`

Example:
`DIM Pst$`
`OUTPUT 707;":MEAS:WCOM:POST?"`
`ENTER 707; Pst$`
`PRINT Pst$`

WCOMpare:WTEST

command/query

The :MEASURE:WCOMPARE:WTEST command turns the waveform compare test on and off. When the waveform compare test is on, the channel data is compared with either memories 1 and 2, or memories 3 and 4. To pass the test, the channel data must be less than the first memory data (memory 1 or 3) and greater than the second memory data (memory 2 or 4) plus or minus the value set by the :WCOMPARE:ALLOWANCE command. The chart in figure 14-2 lists the flow for the waveform comparison test.

The WCOMPARE:WTEST runs only on 500 or 501 point records. The WCOMPARE:WTEST will not run in the expand mode (2000 points) or on digitized data if the count is not at 500. If the COMPARE:WTEST is on and the record sizes are not 500 or 501, the error message "Er 2" will be displayed in the upper-left hand corner of the screen.

Note 

LIMITTEST and WCOMPARE:WTEST cannot run simultaneously. Only one test may be on at one time. If you attempt to run these tests simultaneously, the error message "legal command but safety conflict" (error 211) appears on the screen.

Command Syntax: :MEASure:WCOMpare:WTEST {MEASure | OFF}

Example: OUTPUT 707; ":MEASURE:WCOM:WTEST OFF"

Query Syntax: :MEASure:WCOMpare:WTEST?

Returned Format: [MEASure:WCOMpare:WTEST] {MEASure | OFF}<NL>

Example:
DIM Wte\$
OUTPUT 707; ":MEAS:WCOM:WTE?"
ENTER 707;Wte\$
PRINT Wte\$

Note 

The LTER bit is set when a failure occurs in the :WCOMPARE:WTEST. This bit will remain set after a failure until it is read by the :LTER query.

Timebase Subsystem

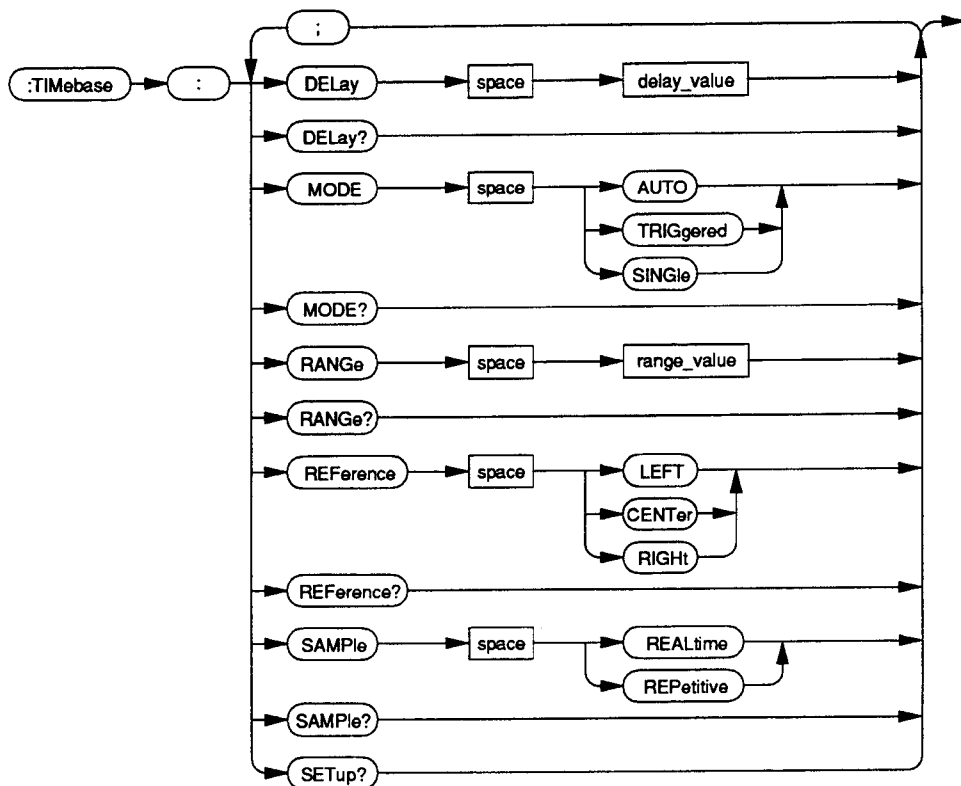
Introduction

The TIMEBASE subsystem commands control the horizontal or X-axis oscilloscope functions.

The Timebase subsystem contains the following commands:

- DELay
- MODE
- RANGe
- REFerence
- SAMPlE
- SETup

Figure 16-1 lists the syntax diagrams for the Timebase subsystem commands.



delay_value = a real number, the maximum value depends on the sweep range.

range_value = a real number, 10 ns through 50 s (in a 1, 2, 5 sequence).

Figure 16-1. Timebase Subsystem Commands Syntax Diagram

DElAy**command/query**

The :TIMEBASE:DELAY command sets the time base delay. This delay is the internal time between the trigger event and the on-screen delay reference point. The delay reference point is the left edge of the display, the right edge of the display, or the center of the display, and is set with the :TIMEBASE:REFERENCE command.

The DELAY query returns the current delay value.

Command Syntax: :TIMebase:DElAy <delay>

Where:

<delay> ::= time in seconds from trigger to the on screen delay reference point. The maximum value depends on the time/division setting.

Example: OUTPUT 707;":TIM:DEL 2MS"

Query Syntax: :TIMebase:DElAy?

Returned Format: [:TIMebase:DElAy] <delay><NL>

Where:

<delay> ::= time from trigger to display reference in seconds. Display reference is left, center, or right (exponential - NR3 format)

Example: DIM D1\$[50]
OUTPUT 707;":TIMEBASE:DELAY?"
ENTER 707;D1\$
PRINT D1\$

MODE

MODE

command/query

The :TIMEBASE:MODE command selects the time base mode. This function is the same as the Auto/Trig'd key in the trigger menu and the SINGLE key on the front panel.

If the AUTO mode is selected, a baseline is provided on the display in the absence of a signal. If a signal is present but the oscilloscope is not triggered, the unsynchronized signal is displayed instead of a baseline.

If the TRIGGERED mode is selected and no trigger is present, the instrument does not sweep, and the data acquired on the previous trigger remains on the screen.

If the SINGLE mode is selected, the screen is cleared and the instrument is stopped. The RUN command arms the trigger, and data is acquired when the trigger is found. The RUN command must be sent to make a single acquisition.

The MODE query returns the current mode.

Command Syntax: :TIMebase:MODE {AUTO | TRIGgered | SINGle}

Example: OUTPUT 707;":TIM:MODE TRIGGERED"

Query Syntax: :TIMebase:MODE?

Returned Format: [:TIMebase:MODE] {AUTO | TRIGgered | SINGle}<NL>

Example: DIM Mode\$[30]
OUTPUT 707;":TIMEBASE:MODE?"
ENTER 707;Mode\$
PRINT Mode\$

RANGe**command/query**

The :TIMEBASE:RANGE command sets the full-scale horizontal time in seconds. The RANGE value is ten times the time per division.

The RANGE query returns the current full-scale range value.

Command Syntax: :TIMebase:RANGe <range>

Where:

<range> ::= 10 ns to 50 s in a 1,2,5 sequence"

Example: OUTPUT 707;":TIM:RANG 100 MS"

Query Syntax: :TIMebase:RANGe?

Returned Format: [:TIMebase:RANGe] <range><NL>

Where:

<range> ::= 10 ns to 50 s (exponential - NR3 format)

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

REference

REference

command/query

The `:TIMEBASE:REFERENCE` command sets the display reference to the left side of the screen, to the right side of the screen, or to the center of the screen.

The `REFERENCE` query returns the current display reference.

Command Syntax: `:TIMebase:REFeRence {LEFT | CENTer | RIGHT}`

Example: `OUTPUT 707;":TIMEBASE:REFERENCE LEFT"`

Query Syntax: `:TIMebase:REFeRence?`

Returned Format: `[[:TIMebase:REFeRence] {LEFT | CENTer | RIGHT}<NL>`

Example:
`DIM Rf$[30]`
`OUTPUT 707;":TIMEBASE:REFERENCE?"`
`ENTER 707;Rf$`
`PRINT Rf$`

SAMPLE**command/query**

The :TIMEBASE:SAMPLE command controls the sampling mode of the instrument. Two sample modes are available: REALTIME and REPETITIVE. Realtime implies "Single-Shot" capture of data, which means that a complete data record is collected on one trigger event. Repetitive implies that the instrument is in a repetitive or equivalent time mode. In this mode the data record is collected over multiple trigger events.

The SAMPLE query returns the current sampling mode.

Command Syntax: :TIMEbase:SAMPle {REALtime | REPetitive}

Example: OUTPUT 707;":TIMEBASE:SAMPLE REAL"

Query Syntax: :TIMEbase:SAMPle?

Returned Format: [:TIMEbase:SAMPle] {REALtime | REPetitive}<NL>

Example: DIM Smp\$ [30]
OUTPUT 707;":TIMEBASE:SAMPLE?"
ENTER 707;Smp\$
PRINT Smp\$

SETup

SETup

query

The :TIMEBASE:SETUP query returns the current settings for all the Timebase Subsystem commands.

Query Syntax: :TIMebase:SETup?

Returned Format: :TIMebase:MODE {AUTO|TRIG|SING};RANGe <range>;DELay <delay>;REF {LEFT|CENT|RIGH};SAMPLe {REAL|REP}<NL>

Where:

<range> ::= 10ns to 50s (exponential - NR3 format)

<delay> ::= time from trigger to display reference in seconds (exponential - NR3 format)

Example:
DIM Stp\$[300]
OUTPUT 707;":TIM:SET?"
ENTER 707;Stp\$
PRINT Stp\$

Trigger Subsystem

Introduction

The commands in the TRIGGER subsystem are used to define the conditions for a trigger. Many of the commands in the Trigger subsystem are valid in more than one of the trigger modes. If the command is a valid command for a trigger mode, that setting is accepted. If the command is not valid for a trigger mode, an error is generated. This subsystem contains the following commands:

- CENTERed
- CONDition
- COUPling*
- DELay
- DELay:SLOPe
- DELay:SOURce
- FIELd
- GLITch
- GLITch:CENTERed
- GLITch:HOLDoff
- GLITch:LEVel
- GLITch:SOURce
- GLITch:WIDth
- HOLDoff
- LEVel
- LINE
- LOGic
- MODE
- OCCurrence
- OCCurrence:SLOPe
- OCCurrence:SOURce
- PATH
- POLarity
- PROBe*

* Available only when the oscilloscope is an HP 54505B/54510B

- QUALify
- SENSitivity
- SETup
- SLOPe
- SOURce
- STANdard

Figure 17-1 lists the syntax diagrams for the Trigger subsystem commands.

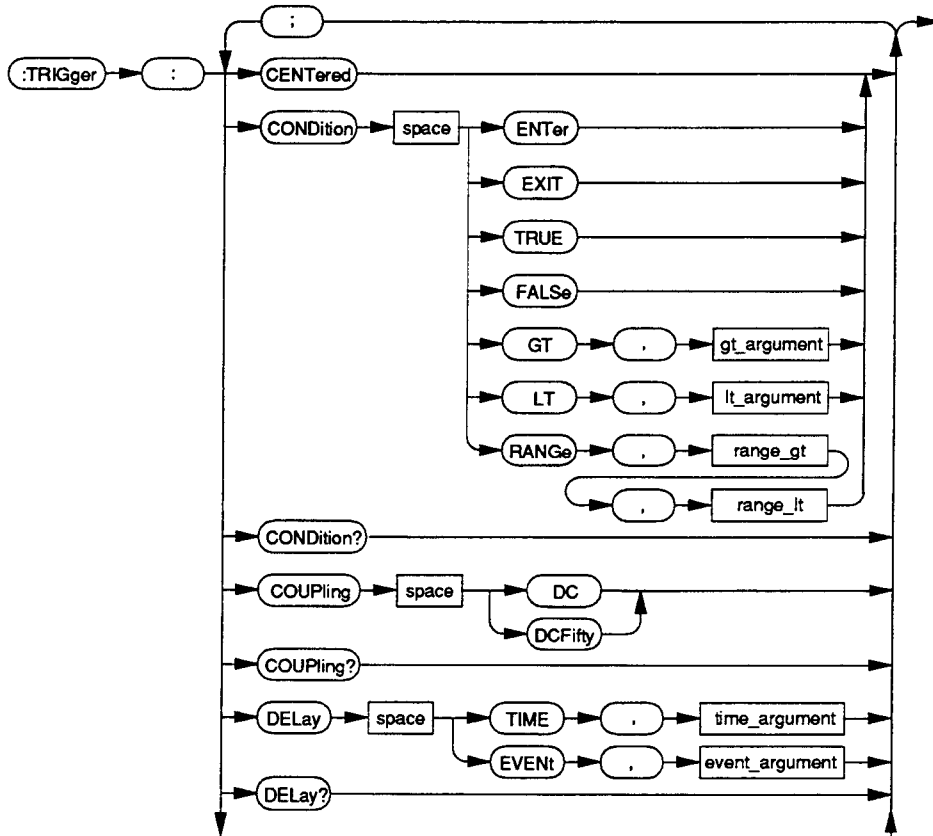


Figure 17-1. Trigger Subsystem Commands Syntax Diagram

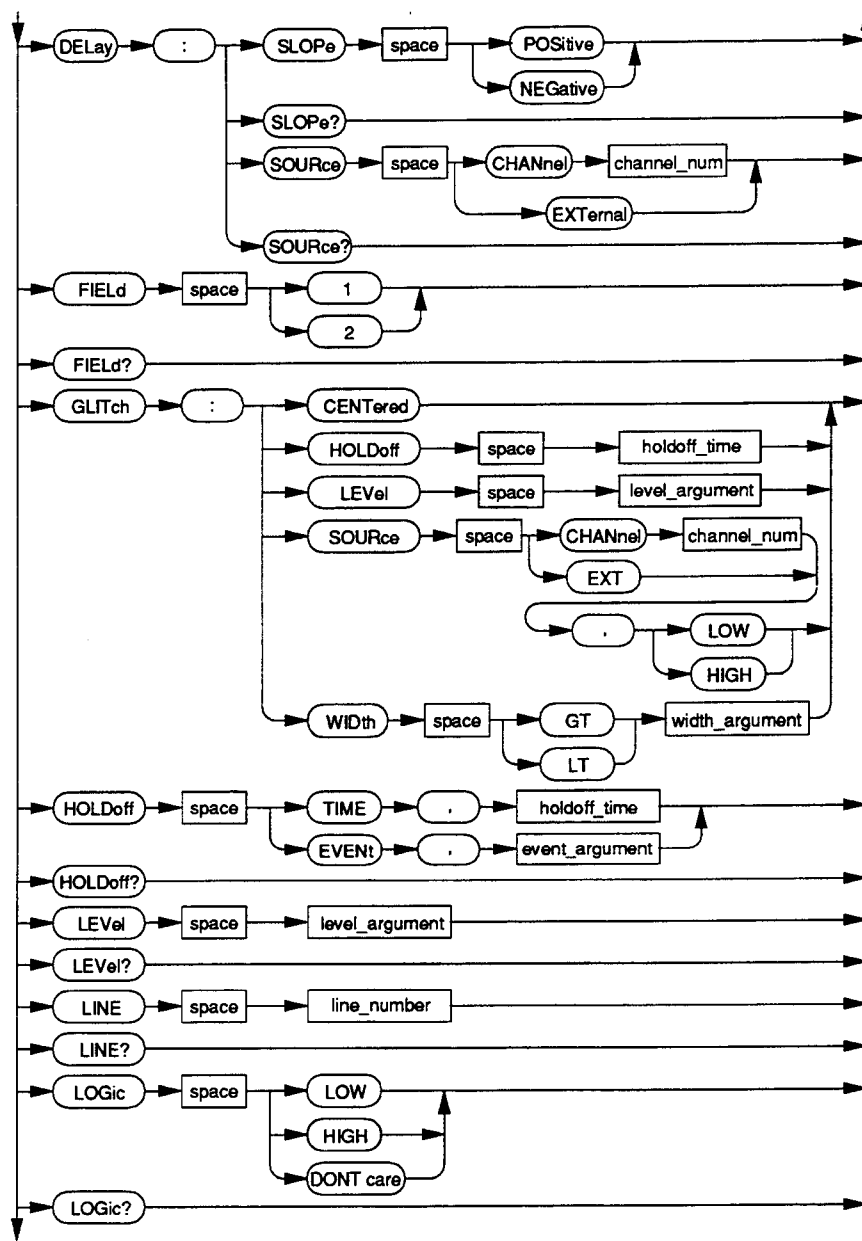


Figure 17-1. Trigger Subsystem Commands Syntax Diagram (continued)

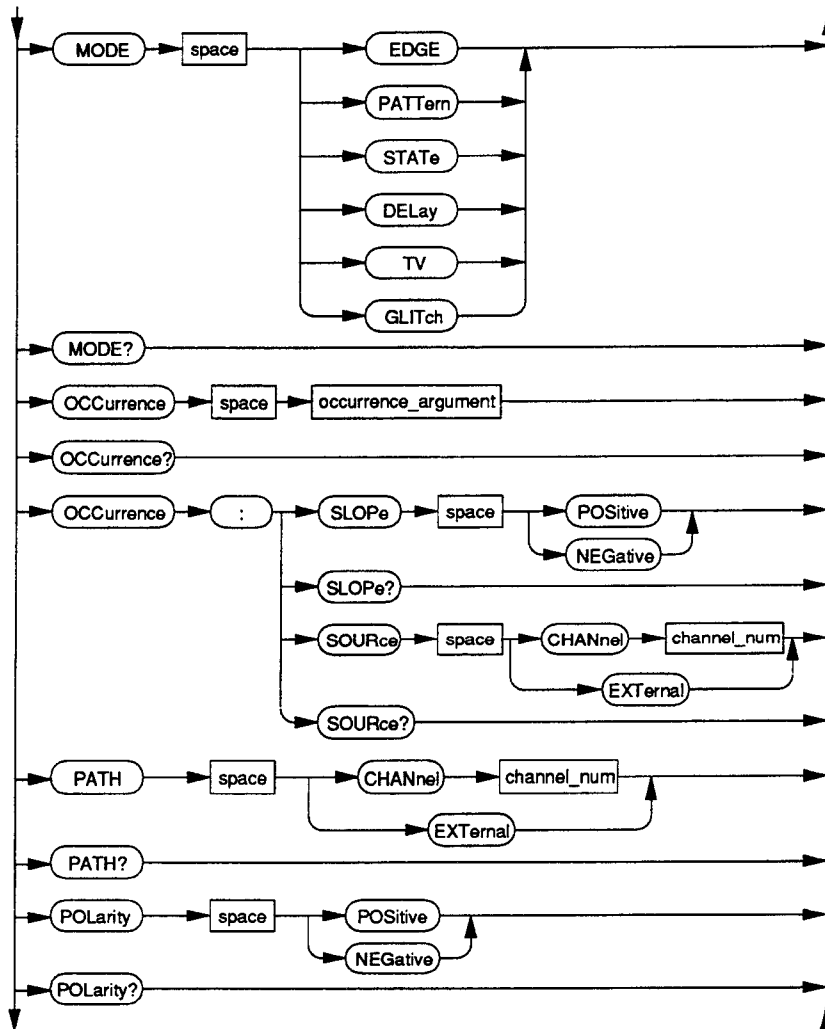


Figure 17-1. Trigger Subsystem Commands Syntax Diagram (continued)

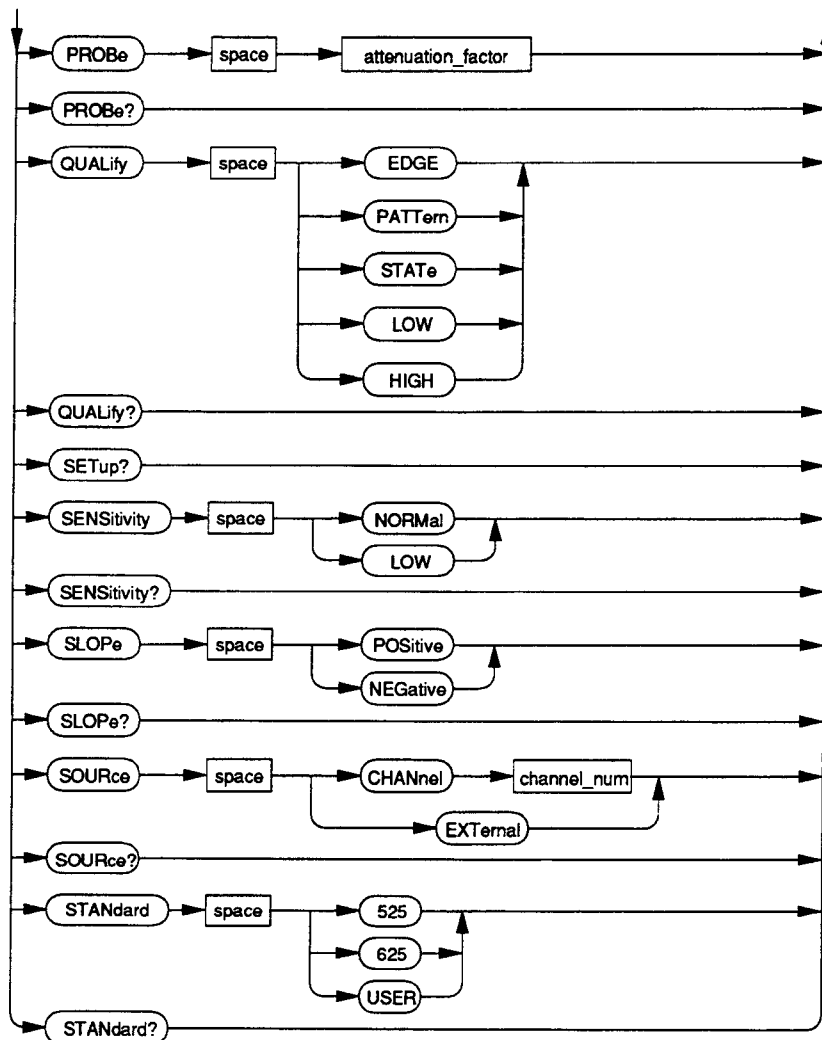


Figure 17-1. Trigger Subsystem Commands Syntax Diagram (continued)

| | |
|------------------------------|---|
| attenuation_factor = | a real number, 0.9 to 1000. |
| channel_num = | an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B). |
| event_argument = | an integer, 1 to 16000000. |
| gt_argument = | a time value, 20 ns to 160 ms. |
| holdoff_time = | a time value, 40 ns to 320 ms. |
| level_argument = | a real number specifying the trigger level in volts. |
| line_number = | an integer, 1 to 625 (depending on the video STANDARD selected). |
| lt_argument = | a time value, 20 ns to 160 ms. |
| range_gt = | a time value, 20 ns to 159.999 ms (the value must be less than range_lt). |
| range_lt = | a time value, 30 ns to 160 ms (the value must be greater than range_gt). |
| time_argument = | a time value, 30 ns to 160 ms. |
| occurrence_argument = | an integer, 1 to 16000000. |
| width_argument = | a time value, 5 ns to 160 ms. |

Figure 17-1. Trigger Subsystem Commands Syntax Diagram (continued)

Trigger Mode

Make sure the instrument is in the proper trigger mode for the command you want to send. The instrument can be placed in the trigger mode from the front panel or over the HP-IB. One method of insuring the instrument is in the proper trigger mode is to send the :TRIGGER:MODE command in the same program message as the parameter to be set. For example,

```
":TRIGGER:MODE TV;LEVEL 200 MV"
```

places the instrument in the TV Trigger Mode and sets the trigger level to 200 mV. This process is necessary because the LEVEL command is also valid for some of the other trigger modes.

The trigger modes are described on the next few pages prior to the descriptions of the individual commands. Table 17-1 lists the different TRIGGER subsystem commands that are available for each trigger mode.



Note Auto or triggered mode is selected with the TIMEBASE:MODE command. For more information, refer to chapter 16, "Timebase Subsystem."

Table 17-1. Valid Commands for Specific Trigger Modes

| EDGE | PATTERN | STATE | DELAY | TV | GLITCH |
|---|--|---|---|---|--|
| CENTERed COUPLing* HOLDoff LEVel PROBe* SENSitivity SLOPe SOURce | CENTERed CONDition COUPLing* HOLDoff LEVel LOGic PATH PROBe* SENSitivity | CENTERed CONDition COUPLing* HOLDoff LEVel LOGic PATH PROBe* SENSitivity SLOPe SOURce | CENTERed CONDition COUPLing DELay DELay:SLOPe DELay:SOURce LEVel LOGic OCCurrence OCCurrence:SLOPe OCCurrence:SOURce PATH PROBe* QUALify SENSitivity SLOPe SOURce | CENTERed CONDition COUPLing* FIELd HOLDoff LEVel LINE OCCurrence OCCurrence:SLOPe POLarity PROBe* QUALify SENSitivity SOURce STANdard | COUPLing* GLITCh:CENTERed GLITCh:HOLDoff GLITCh:LEVel GLITCh:SOURce GLITCh:WIDth PROBe* SENSitivity |
| * = Available only when oscilloscope is an HP 54505B/54510B. | | | | | |

The EDGE Trigger Mode

The Edge Trigger Mode is the easiest trigger mode to understand and use from the front panel or over the HP-IB, because it has the least number of parameters to be set. This explanation of the trigger mode commands follow the front-panel keys very closely. Refer to the *HP 54505B/54506B and HP 54510B/54512B Front-Panel Reference* for further explanations of the trigger operation.

In the Edge Trigger Mode you must set the trigger source using the `:TRIGGER:SOURCE` command. This selects the source that the oscilloscope triggers on. The argument for the `:TRIGGER:SOURCE` command is `channel1`, `channel2`, or `external` (HP 54505B/54510B); or `channel1` through `channel4` (HP 54506B/54512B).

After setting the trigger source, you need to set the trigger level. This value is set using the `:TRIGGER:LEVEL` command and can be set for each trigger source. The trigger level values that are set in the Edge Trigger Mode are used for all modes except the TV and Glitch Trigger Mode. Conversely, the levels set in the `PATTERN`, `STATE`, or `DELAY` modes set the levels for the EDGE mode.

The actual edge that creates the trigger is set with the `:TRIGGER:SLOPE` command. This command can be set to `POSITIVE` or `NEGATIVE` for each of the sources.

The last setting in the Edge Trigger Mode is the Trigger Holdoff value. This value is used only for the EDGE mode.

The Pattern Trigger Mode

This description of the Pattern Trigger Mode follows the front-panel keys very closely. There are additional parameters in this mode that are not used in the Edge Trigger Mode and one parameter that is carried over from the Edge Trigger Mode. The one parameter that carries over is LEVEL. If the level command is used in this mode it also changes the level value for that source in the Edge Trigger Mode. In this trigger mode, the :TRIGGER:LEVEL command defines the voltage that determines if the input voltage is a logic high or logic low for the logic triggers.

The logic pattern for the Pattern Trigger Mode is set using the PATH and LOGIC commands. The PATH command specifies which of the two inputs is selected for the logic pattern. Once the path is selected, the pattern can be set using the LOGIC command. The LOGIC command uses the arguments HIGH, LOW, and DONTCARE to set the "trigger on" bit pattern.

The CONDITION command sets the "when" field on the front panel. This command is used in several of the trigger modes; therefore, it has parameters that are not valid in this mode. The valid parameters for the CONDITION command in the Pattern Trigger Mode are ENTER, EXIT, GT (Greater Than), LT (Less Than), and RANGE.

When the command :TRIGGER:CONDITION ENTER or :TRIGGER:CONDITION EXIT is sent, the Entered or Exited parameter is set on the front panel. When the GT or LT option is used, a time value must be sent to define the limit. When the RANGE option is used, two time values must be sent to define the lower and upper limit. The correct syntax for the RANGE option is :TRIGGER:CONDITION RANGE, <range_low>, <range_high>.

The holdoff time is set in the Pattern Trigger Mode with the :TRIGGER:HOLDOFF command.

The State Trigger Mode

When the State Trigger Mode is selected, the `:TRIGGER:SOURCE` command is used to select the clock source. The syntax for selecting channel 2 as the clock source is `:TRIGGER:SOURCE CHANNEL2`.

After the clock source is selected, the correct edge for the clock can be selected using the `:TRIGGER:SLOPE` command, which can be set to `NEGATIVE` or `POSITIVE`.

The `:TRIGGER:PATH` command can be used with the `:TRIGGER:LOGIC` command to set the three bit logic pattern to qualify the clock trigger. These commands can be sent using the syntax `:TRIGGER:PATH CHAN2;;TRIGGER:LOGIC LOW`, or `:TRIGGER:PATH CHAN2; LOGIC LOW`.

In this trigger mode, the `:TRIGGER:LEVEL` command defines the voltage that determines if the input voltage is a logic high or a logic low for the logic triggers.

The `:TRIGGER:CONDITION` command sets the "is/is not present" state using the parameters `TRUE` for "is present" and `FALSE` for "is not present."

In the State Trigger Mode, a holdoff value can be set as in all other modes.

The Delay Trigger Mode

In the Delay Trigger Mode, the TRIGGER:QUALIFY command can be used to select the EDGE, PATTERN, or STATE mode as a qualifier. When the EDGE qualifier is selected, all Edge parameters and commands can be used to set the Source and Slope. When the PATTERN qualifier is selected, all Pattern commands can be used to set the pattern mode parameters. When the STATE qualifier is selected, all State commands can be used to set the state mode parameters.

The next settings (in front panel order) are the delay settings. The DELAY command is used to set the Time or Count parameter and the amount of delay. To set the delay to time use the command :TRIGGER:DELAY TIME, < time > . To set the delay to count use the command :TRIGGER:DELAY EVENT < number_events > .

If the trigger delay is set to Event (count) you can then set the delay source and slope. To set the delay source, use the command :TRIGGER:DELAY:SOURCE CHANNEL2. To set the delay slope, use the command TRIGGER:DELAY:SLOPE POSITIVE.

The values within the front-panel "trigger on" field are set with the OCCURRENCE command. To set the number of occurrences to 3333, use the command syntax TRIGGER:OCCURRENCE 3333. To set the source for the number of occurrences to channel 2, use the command syntax :TRIGGER:OCCURRENCE:SOURCE CHANNEL2. To set the slope of the trigger occurrence to negative, use the command syntax :TRIGGER:OCCURRENCE:SLOPE NEGATIVE.

The TV Trigger Mode

The TV Trigger Mode is used for triggering on clamped television signals. This mode allows you to select one of the TV signal frames and one of the lines within that frame.

Once the TV Trigger Mode is selected, the Television Signal Standard can be selected using the `:TRIGGER:STANDARD` command. The three parameters for this command are 525, 625, and USER. Any of these modes allow you to select the trigger level and the source of the trigger signal.

The trigger level is set by sending the command `:TRIGGER:LEVEL <value>`.

The trigger source is set by sending the `:TRIGGER:SOURCE` command. This command allows you to select channel 1 or channel 2 as the trigger source.

With the standard set to 525 or 625, the commands that can be used are POLARITY, FIELD, and LINE. The POLARITY command sets the edge for the trigger. This edge can be set to NEGATIVE or POSITIVE. The FIELD command selects the first or second field of the television signal. The LINE command specifies the horizontal line in which the instrument will trigger on. Refer to the LINE command to determine the correct values.

The HOLDOFF value can also be set in the TV trigger mode, as in all modes.

When the USER (user defined) standard is selected, the source and level are set in the same manner as before.

The QUALIFY command is used to set the "qualify on" field. This command can be set to HIGH or LOW.

The edge defined by the QUALIFY command must occur within the range of time values that are displayed in the front panel field. The TRIGGER:CONDITION RANGE command sets the greater than and less than time values. In order to actually generate a trigger, the qualified conditions must be met within the specified time. To set the time values, send the command :TRIGGER:CONDITION RANGE, <greater_than_value>, <less_than_value>.

The field, "trigger on," is set with the OCCURRENCE command and OCCURRENCE:SLOPE command. To set the number of occurrences, send the command :TRIGGER:OCCURRENCE <number>. To set the slope for the occurrences, send the command :TRIGGER:OCCURRENCE:SLOPE <slope>. The slope can be set to POSITIVE or NEGATIVE.

The description for each command tells you in which modes that command is valid.

The Glitch Trigger Mode

When the Glitch Trigger Mode is selected, the `:TRIGGER:GLITCH:SOURCE` command is used to select both the trigger source and the pattern state. The syntax for selecting channel 2 as the trigger source, and high as the state, is `:TRIGGER:GLITCH:SOURCE CHANNEL2,HIGH`.

The trigger level is set by sending the command `:TRIGGER:GLITCH:LEVEL <value>`.

The condition under which a trigger is generated is set using the `TRIGGER:GLITCH:WIDTH` command. This command is used to specify that the trigger will be generated if the glitch width is less than (LT) or greater than (GT) a specified time value. The correct syntax for the WIDTH option is `:TRIGGER:GLITCH:WIDTH {GT|LT},<width>`.

The holdoff time is set in the Glitch Trigger Mode with the `:TRIGGER:GLITCH:HOLDOFF` command.

CENTERed**command**

The `:TRIGGER:CENTERED` command sets the trigger level to the centered mode (at the center of the graph) when the internal source is selected. This command is not valid when the external source is selected.

Note

To return to the ADJUST mode, specify a level with the `TRIGGER:LEVEL` command.

Command Syntax: `:TRIGger:CENTerEd`

Example: `OUTPUT 707;":TRIGGER:SOURCE CHANNEL1" !select trigger source`
`OUTPUT 707;":TRIGGER:CENTERED"`

CONDition

CONDition

command/query

The `:TRIGGER:CONDITION` command is valid in the `PATTERN`, `STATE`, `DELAY`, and `TV` trigger modes. The time values entered using this command are rounded to the nearest 10 ns.

In the **Pattern Trigger Mode**, the valid arguments for the `CONDITION` command are `ENTER`, `EXIT`, `GT`, `LT`, `RANGE`.

In the **Pattern Trigger Mode**, the `CONDITION` command is used to specify whether the trigger is generated upon entering (`ENTER`) or leaving (`EXIT`) the specified logic pattern. Also, the `CONDITION` command specifies whether the pattern must be present for a specified amount of time. The time in the pattern trigger mode can be greater than a value (`GT`), less than a value (`LT`), or between two values (`RANGE`). These settings can also be specified from the front panel using the "when" key in the **Pattern Trigger Mode**.

In the **State Trigger Mode**, the valid parameters for the `CONDITION` command are `TRUE` (is present) and `FALSE` (is not present).

In the **Delay Trigger Mode**, the `CONDITION` command is valid when `PATTERN` or `STATE` is selected as the qualifier. All arguments for this command that are valid in the **Pattern** or **State Trigger Modes** are valid here.

In the **TV Trigger Mode**, the `CONDITION` command is used to set the range of time values on which the trigger occurs. This command is only valid in the "user defined" mode.

The `CONDITION` query returns the currently selected condition for the currently selected mode.

Query Syntax: :TRIGger:CONDition?

Returned Format: [:TRIGger:CONDition] <argument><NL>

Where in PATTERN or DELAY PATTERN mode:

<argument> ::= {ENTer | EXIT | GT,<value> | LT,<value> |
RANGE,<range_gt>,<range_lt>}

Where in STATE or DELAY STATE mode:

<argument> ::= {TRUE | FALSE}

Where in TV mode:

<argument> ::= RANGE,<range_gt>,<range_lt>

Where:

<value> ::= 20 ns to 160 ms

<range_gt> ::= 20 ns to 159.999 ms (must be less than range_lt)

<range_lt> ::= 30 ns to 160 ms (must be greater than range_gt)

Example: DIM Con\$ [50]
OUTPUT 707;":TRIGGER:CONDITION?"
ENTER 707;Con\$
PRINT Con\$

COUPLing

COUPLing

command/query

The `:TRIGGER:COUPLING` command (HP 54505B/54510B only) selects the input coupling for the selected external trigger source. The coupling can be set to DC or DCFifty. The DCFifty parameter places a 50 Ω load on the trigger input.

The `COUPLING` query returns the current selection.

Command Syntax: `:TRIGger:COUPling {DC | DCFifty}`

Example: `OUTPUT 707;":TRIGGER:COUPLING DC"`

Query Syntax: `:TRIGger:COUPling?`

Returned Format: `[[:TRIGger:COUPling] {DC | DCFifty}<NL>`

Example:
`DIM Mode$[50]
OUTPUT 707;":TRIG:COUP?"
ENTER 707;Mode$
PRINT Mode$`

DElay

command/query

The **:TRIGGER:DELAY** command is valid only in the **Delay Trigger Mode**. This command sets a delay value in either time or number of events. In the time delay mode, this command specifies the delay value in seconds. In the events delay mode, this command specifies the delay value in number of trigger events.

The **DELAY** query returns the current delay value.

Command Syntax: `:TRIGger:DElay {TIME,<time_value> | EVENT,<event_value>}`

Where:

`<time_value>` ::= amount of delay from 30 ns to 160 ms
`<event_value>` ::= number of events from 1 to 16000000

Example: `OUTPUT 707;":TRIGGER:MODE DELAY" !select trigger mode`
`OUTPUT 707;":TRIGGER:DELAY TIME,1.23E-01"`

Query Syntax: `:TRIGger:DElay?`

Returned Format: `[[:TRIGger:DElay] {TIME,<time_value> | EVENT,<event_value>}<NL>`

Where:

`<time_value>` ::= amount of delay from 30 ns to 160 ms
`<event_value>` ::= number of events from 1 to 16000000

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

DElay:SLOPe

DElay:SLOPe

command/query

The `:TRIGGER:DELAY:SLOPE` command selects the edge that will be counted by the delay command. The parameters for this command are `NEGATIVE` or `POSITIVE`. This command is only valid in the **Delay Trigger Mode**.

The `DELAY:SLOPE` query returns the current delay slope.

Command Syntax: `:TRIGger:DElay:SLOPe {POSitive | NEGative}`

Example: `OUTPUT 707;":TRIGGER:MODE DELAY" !select trigger mode`
`OUTPUT 707;":TRIG:DEL:SLOP POS"`

Query Syntax: `:TRIGger:DElay:SLOPe?`

Returned Format: `[[:TRIGger:DElay:SLOPe] {POSitive | NEGative}<NL>`

Example: `DIM Tos$[50]`
`OUTPUT 707;":TRIGGER:DELAY:SLOP?"`
`ENTER 707;Tos$`
`PRINT Tos$`

DElay:SOURce**command/query**

The **:TRIGGER:DELAY:SOURCE** command selects the source that is counted by the delay command. The parameters for this command are **CHANNEL1**, **CHANNEL2**, or **EXTERNAL** (Hp 54505B/54510B); or **CHANNEL1** through **CHANNEL4** (HP 54506B/54512B). This command is only valid in the **Delay Trigger Mode**.

The **DELAY:SOURCE** query returns the source of the delay in the Delay Trigger Mode.

Command Syntax: `:TRIGger:DElay:SOURce {CHANne11 | CHANne12 | EXTERna1} (HP 54505B/54510B)`
`:TRIGger:DElay:SOURce {CHANne11 | CHANne12 | CHANne13 | CHANne14}`
 (HP 54506B/54512B)

Example: `OUTPUT 707;":TRIGGER:MODE DELAY" !select trigger mode`
`OUTPUT 707;":TRIG:DEL:SOURCE CHANNEL2"`

Query Syntax: `:TRIGger:DElay:SOURce?`

Returned Format: `[:TRIGger:DElay:SOURce] {CHANne11 | CHANne12 | EXTERna1}<NL>`
 (HP 54505B/54510B)
`[:TRIGger:DElay:SOURce] {CHANne11 | CHANne12 | CHANne13 | CHANne14}<NL>`
 (HP 54506B/54512B)

Example: `DIM Tos$[50]`
`OUTPUT 707;":TRIGGER:DELAY:SOUR?"`
`ENTER 707;Tos$`
`PRINT Tos$`

FIELD

FIELD

command/query

The `:TRIGGER:FIELD` command is only valid in the **TV Trigger Mode**. This command selects the field of the TV signal when the **STANDARD** is set to 525 or 625. The valid parameters for this command are 1 or 2.

If the `:TRIGGER:FIELD` command is set in any other trigger mode, error -221, "Settings conflict," is placed in the error queue.

The `FIELD` query returns the current setting of the `FIELD` command.

Command Syntax: `:TRIGger:FIELD { 1 | 2 }`

Example:
`OUTPUT 707;":TRIGGER:MODE TV" !select trigger mode`
`OUTPUT 707;":TRIGGER:FIEL 2"`

Query Syntax: `:TRIGger:FIELD?`

Returned Format: `[:TRIGger:FIELD] { 1 | 2 }<NL>`

Example:
`DIM F$[50]`
`OUTPUT 707;":TRIG:FIELD?"`
`ENTER 707;F$`
`PRINT F$`

GLITCh**subsystem**

The `:TRIGGER:GLITCh` commands are used to define the conditions for a trigger in the **Glitch Trigger Mode**. All of the commands in the Glitch Trigger subsystem are not used in any of the other trigger modes. This subsystem contains the following commands:

- `GLITCh:CENTERed`
- `GLITCh:HOLDoff`
- `GLITCh:LEVel`
- `GLITCh:SOURce`
- `GLITCh:WIDth`

GLITCh:CENTERed

GLITCh:CENTERed

command

The :TRIGGER:GLITCH:CENTERED command sets the trigger level to the centered mode (at the center of the graph). This command is not valid when the external source is selected (HP 54505B/54510B only). This command is only valid in the **Glitch Trigger Mode**.



To return to the ADJUST mode, specify a level with the TRIGGER:GLITCH:LEVEL command.

Command Syntax: :TRIGger:GLITCh:CENTERed

Example: OUTPUT 707;":TRIGGER:GLITCH:SOURCE CHANNEL1,HIGH" !select trigger source
and state
OUTPUT 707;":TRIGGER:GLITCH:CENTERED"

GLITch:HOLDoff

command/query

The `:TRIGGER:GLITch:HOLDOFF` command is used to enter a the amount of time the trigger circuit is disabled after the trigger event has occurred. This command is only valid in the **Glitch Trigger Mode**.

The `GLITCH:HOLDOFF` query returns the value of the holdoff for the **Glitch Trigger Mode**.

Command Syntax: `:TRIGger:GLITch:HOLDoff <value>`

Where:

`<value>` ::= 40 ns to 320 ms rounded to nearest 20 ns increment.

Example: `OUTPUT 707;":TRIGGER:GLITCH:HOLDOFF 216 US"`

Query Syntax: `:TRIGger:GLITch:HOLDoff?`

Returned Format: `[:TRIGger:GLITch:HOLDoff] <value><NL>`

Where:

`<value>` ::= 40 ns to 320 ms (exponential - NR3 format)

Example: `DIM Ho$[50]
OUTPUT 707;":TRIGGER:GLITCH:HOLD?"
ENTER 707;Ho$
PRINT Ho$`

GLITCh:LEVeL

GLITCh:LEVeL

command/query

The `:TRIGGER:GLITCH:LEVEL` command sets the trigger level voltage for the glitch trigger mode. This command is only valid in the **Glitch Trigger Mode**.

The `GLITCH:LEVEL` query returns the trigger level of the **Glitch Trigger Mode**.

Command Syntax: `:TRIGger:GLITCh:LEVeL <level>`

Where:

`<level>` ::= for internal triggers, $\pm 1.5 \times$ full-scale voltage from center screen.
for external triggers (HP 54510B only), ± 2 volts with probe attenuation at 1:1.

Examples: `OUTPUT 707;":TRIGGER:GLITCH:LEVEL .30"`

`OUTPUT 707;":TRIGGER:GLIT:LEV 300MV"`

`OUTPUT 707;":TRIG:GLIT:LEV 3E-1"`

Refer to chapter 27, "Message Communication and System Functions" for the syntax of using values with multipliers.

Query Syntax: `:TRIGger:GLITCh:LEVeL?`

Returned Format: `[[:TRIGger:GLITCh:LEVeL] <level><NL>`

Where:

`<level>` ::= trigger level in volts (exponential - NR3 format)

Example:
`DIM Tlevel$[30]`
`OUTPUT 707;":TRIGGER:GLITCH:LEVEL?"`
`ENTER 707;Tlevel$`
`PRINT Tlevel$`

GLITCh:SOURce

command/query

The `:TRIGGER:GLITCH:SOURCE` command selects the channel and pattern state that will actually produce the trigger. This command is only valid in the **Glitch Trigger Mode**.

- The source parameters for this command are CHANNEL1, CHANNEL2, or EXTERNAL (HP 54505B/54510B); or CHANNEL1 through CHANNEL4 (HP 54506B/54512B).
- The state for the source selected can be specified as either HIGH or LOW, where HIGH is higher than the current GLITCH:LEVEL value, and LOW is lower than the current GLITCH:LEVEL value.

The `GLITCH:SOURCE` query returns the current source for the **Glitch Trigger Mode**.

Command Syntax: `:TRIGger:GLITCh:SOURce {CHANne11|CHANne12|EXTERna1},{HIGH|LOW}`
(HP 54505B/54510B)

`:TRIGger:GLITCh:SOURce {CHANne11|CHANne12|CHANne13|CHANne14},{HIGH|LOW}`
(HP 54506B/54512B)

Example: `OUTPUT 707;":TRIGGER:GLITCH:SOURCE CHAN2,LOW"`

Query Syntax: `:TRIGger:GLITCh:SOURce?`

Returned Format: `[[:TRIGger:GLITCh:SOURce] {CHANne11|CHANne12|EXTERna1},{HIGH|LOW}`
(HP 54505B/54510B)

`[[:TRIGger:GLITCh:SOURce] {CHANne11|CHANne12|CHANne13|CHANne14},{HIGH|LOW}`
(HP 54506B/54512B)

Example: `DIM Src$[30]`
`OUTPUT 707;":TRIGGER:GLITCH:SOURCE?"`
`ENTER 707;Src$`
`PRINT Src$`

GLITCh:WIDth

GLITCh:WIDth

command/query

The `:TRIGGER:GLITCH:WIDTH` command is used to specify whether the trigger is generated when the pattern is present for greater than (GT), or less than (LT) a specified amount of time.

The `GLITCH:WIDTH` query returns the current condition and time value for the **Glitch Trigger Mode**.

Command Syntax: `:TRIGger:GLITCh:WIDth {GT|LT},<width>`

Where:

`<width>` ::= 5 ns to 160 ms

Example: `OUTPUT 707;":TRIG:GLIT:WIDT GT,33ms"`

Query Syntax: `:TRIGger:GLITCh:WIDth?`

Returned Format: `[:TRIGger:GLITCh:WIDth] {GT|LT},<width><NL>`

Where:

`<width>` ::= 5 ns to 160 ms (exponential - NR3 format)

Example:
`DIM Con$[50]`
`OUTPUT 707;":TRIGGER:GLITCH:WIDTH?"`
`ENTER 707;Con$`
`PRINT Con$`

HOLDoff

command/query

The **:TRIGGER:HOLDOFF** command is valid in the **Edge, Pattern, State, and TV Trigger Modes**. The **HOLDOFF** command enters a holdoff value in terms of time or events.

The **HOLDOFF** query returns the value of the holdoff for the current mode.

Command Syntax: `:TRIGger:HOLDoff {TIME,<holdoff_value> | EVENT,<event_argument>}`

Where:

`<holdoff_value>` ::= 40 ns to 320 ms rounded to nearest 20 ns increment.
`<event_argument>` ::= 1 to 16000000

Examples: `OUTPUT 707;":TRIGGER:HOLDOFF TIME,216 US"`

`OUTPUT 707;":TRIGGER:HOLDOFF EVENT,10"`

Query Syntax: `:TRIGger:HOLDoff?`

Returned Format: `[[:TRIGger:HOLDoff] {TIME,<holdoff_value> | EVENT,<event_argument>}<NL>`

Where:

`<holdoff_value>` ::= 40 ns to 320 ms (exponential - NR3 format)
`<event_argument>` ::= 1 to 16000000 (integer - NR1 format)

Example:
`DIM Ho$[50]`
`OUTPUT 707;":TRIGGER:HOLD?"`
`ENTER 707;Ho$`
`PRINT Ho$`

LEVeL

LEVeL

command/query

The `:TRIGGER:LEVEL` command sets the trigger level voltage of the active trigger. This command can be sent in any mode; however, only two separate levels are stored. One value is kept for the TV Trigger Mode and another value is kept for all other modes (except Glitch). If you are in the Pattern Trigger Mode and set a trigger level value, that level is also used for the Edge, State, and Delay Trigger Modes.

The `LEVEL` query returns the trigger level of the current trigger mode.

Command Syntax: `:TRIGger:LEVeL <level>`

Where:

`<level>` ::= for internal triggers, $\pm 1.5 \times$ full-scale voltage from center screen.
for external triggers, ± 2 volts with probe attenuation at 1:1.

Examples: `OUTPUT 707;":TRIGGER:LEVEL .30"`

`OUTPUT 707;":TRIGGER:LEV 300MV"`

`OUTPUT 707;":TRIG:LEV 3E-1"`

Refer to chapter 27, "Message Communication and System Functions" for the syntax of using values with multipliers.

Query Syntax: `:TRIGger:LEVeL?`

Returned Format: `[:TRIGger:LEVeL] <level><NL>`

Where:

`<level>` ::= trigger level in volts (exponential - NR3 format)

Example:
`DIM Tlevel$[30]`
`OUTPUT 707;":TRIGGER:LEVEL?"`
`ENTER 707;Tlevel$`
`PRINT Tlevel$`

LINE

command/query

The `:TRIGGER:LINE` command specifies the horizontal line in which the instrument will trigger on. The `LINE` command is valid in the **TV Trigger Mode** when the `STANDARD` selected is 525 or 625. If one of these standards is selected when the `TV Trigger Mode` is entered, the line value is set in that standard and selected field.

The `LINE` query returns the current line of the selected standard.

Command Syntax: `:TRIGger:LINE <line_number>`

Where:

`<line_number> ::= 1 to 625 (depends on STANDARD and FIELD selection).`

Example:

```
OUTPUT 707;":TRIGGER:MODE TV"      !select trigger mode
OUTPUT 707;":TRIGGER:STANDARD 525" !select TV signal standard
OUTPUT 707;":TRIG:LINE 22"
```

Query Syntax: `:TRIGger:LINE?`

Returned Format: `[:TRIGger:LINE] <line_number><NL>`

Where:

`<line_number> ::= 1 to 625 (depends on STANDARD and FIELD selection).
(integer - NR1 format)`

Example:

```
DIM Ln$[50]
OUTPUT 707;":TRIGGER:LINE?"
ENTER 707;Ln$
PRINT Ln$
```

LOGic

LOGic

command/query

The `:TRIGGER:LOGIC` command is valid in the **Pattern and State Trigger Modes**, as well as the **DELAY Trigger Mode** when qualifying by **PATTERN** or **STATE**. The **LOGIC** command specifies the relationship between the signal and the defined voltage level that must exist before that part of the pattern is considered valid. If the signal on a selected path is greater than the trigger level, that signal is considered **HIGH**. If the signal is less than the trigger level, it is considered **LOW**.

The **LOGIC** query returns the last specified logic level of the currently enabled path.

Command Syntax: `:TRIGger:LOGic {HIGH | LOW | DONTcare}`

Example:

```
OUTPUT 707;":TRIGGER:MODE DELAY"           !select trigger mode
OUTPUT 707;":TRIGGER:QUALIFY PATTERN"      !qualify trigger
OUTPUT 707;":TRIG:LOGIC DONT"
```

Query Syntax: `:TRIGger:LOGic?`

Returned Format: `[:TRIGger:LOGic] {HIGH | LOW | DONTcare}<NL>`

Example:

```
DIM L$[50]
OUTPUT 707;":TRIGGER:LOGIC?"
ENTER 707;L$
PRINT L$
```

MODE**command/query**

The `:TRIGGER:MODE` command selects the trigger mode. This command can be sent from any trigger mode.

The `MODE` query returns the currently selected trigger mode.

Command Syntax: `:TRIGger:MODE {EDGE | PATtern | STATE | DELay | TV | GLITCh}`

Example: `OUTPUT 707;":TRIGGER:MODE PATT"`

Query Syntax: `:TRIGger:MODE?`

Returned Format: `[[:TRIGger:MODE] {EDGE | PATtern | STATE | DELay | TV | GLITCh}<NL>`

Example:
`DIM Mode$[50]
OUTPUT 707;":TRIGGER:MODE?"
ENTER 707;Mode$
PRINT Mode$`

OCCurrence

OCCurrence

command/query

The `:TRIGGER:OCCURRENCE` command sets the number of trigger events that must occur before the oscilloscope is actually triggered. This command is valid in the **Delay Trigger Mode** and in the **TV Trigger Mode**.

The `OCCURRENCE` query returns the current value of the `OCCURRENCE` command when the oscilloscope is in the **Delay Trigger Mode**, or in the **TV Trigger Mode** with **USER DEFINED** selected.

Command Syntax: `:TRIGger:OCCurrence <occ_number>`

Where:

`<occ_number> ::= 1 to 16000000`

Example: `OUTPUT 707;":TRIGGER:MODE DELAY" !select trigger mode`
`OUTPUT 707;":TRIGGER:OCC 14"`

Query Syntax: `:TRIGger:OCCurrence?`

Returned Format: `[:TRIGger:OCCurrence] <occ_number><NL>`

Where:

`<occ_number> ::= 1 to 16000000 (integer - NR1 format)`

Example: `DIM 0c$[50]`
`OUTPUT 707;":TRIGGER:OCCURRENCE?"`
`ENTER 707;0c$`
`PRINT 0c$`

OCCurrence:SLOPe**command/query**

The `:TRIGGER:OCCURRENCE:SLOPE` command selects the edge that will be counted by the occurrence command. The parameters for this command are `NEGATIVE` or `POSITIVE`. This command is valid in the **Delay Trigger Mode** and the **TV Trigger Mode**.

The `OCCURRENCE:SLOPE` query returns the slope of the current mode.

Command Syntax: `:TRIGger:OCCurrence:SLOPe {POSitive | NEGative}`

Example: `OUTPUT 707;":TRIGGER:MODE DELAY" !select trigger mode`
`OUTPUT 707;":TRIG:OCC:SLOP POS"`

Query Syntax: `:TRIGger:OCCurrence:SLOPe?`

Returned Format: `[[:TRIGger:OCCurrence:SLOPe] {POSitive | NEGative}<NL>`

Example: `DIM Tos$[50]`
`OUTPUT 707;":TRIGGER:OCCURRENCE:SLOP?"`
`ENTER 707;Tos$`
`PRINT Tos$`

OCCurrence:SOURce

OCCurrence:SOURce

command/query

The `:TRIGGER:OCCURRENCE:SOURCE` command selects the source that will be counted by the occurrence command. The parameters for this command are CHANNEL1, CHANNEL2, or EXTERNAL (HP 54505B/54510B); or CHANNEL1 through CHANNEL4 (HP 54506B/54512B). This command is valid only in the **Delay Trigger Mode**.

The `OCCURRENCE:SOURCE` query returns the source of the occurrence in the Delay Trigger Mode.

Command Syntax: `:TRIGger:OCCurrence:SOURce {CHANne11|CHANne12|EXterna1} (HP 54505B/54510B)`
`:TRIGger:OCCurrence:SOURce {CHANne11|CHANne12|CHANne13|CHANne14}`
(HP 54506B/54512B)

Example: `OUTPUT 707;":TRIGGER:MODE DELAY" !select trigger mode`
`OUTPUT 707;":TRIG:OCC:SOURCE CHANNEL2"`

Query Syntax: `:TRIGger:OCCurrence:SOURce?`

Returned Format: `[[:TRIGger:OCCurrence:SOURce] {CHANne11|CHANne12|EXterna1}<NL>`
(HP 54505B/54510B)
`[[:TRIGger:OCCurrence:SOURce] {CHANne11|CHANne12|CHANne13|CHANne14}<NL>`
(HP 54506B/54512B)

Example: `DIM Tos$[50]`
`OUTPUT 707;":TRIGGER:OCCURRENCE:SOUR?"`
`ENTER 707;Tos$`
`PRINT Tos$`

PATH

command/query

The `:TRIGGER:PATH` command is valid in the **Pattern Trigger Mode, State Trigger Mode, and Delay Trigger Mode** when "qualify on" pattern or state is selected. This command selects a pattern bit as the source for future logic commands.

The `PATH` query returns the current trigger source of the present mode.

Command Syntax: `:TRIGger:PATH {CHANne11|CHANne12|EXTErna1}` (HP 54505B/54510B)
`:TRIGger:PATH {CHANne11|CHANne12|CHANne13|CHANne14}` (HP 54506B/54512B)

Example: `OUTPUT 707;":TRIGGER:MODE PATTERN"` !select trigger mode
`OUTPUT 707;":TRIGGER:PATH CHANNEL2"`
`OUTPUT 707;":TRIGGER:LOGIC HIGH"` !select logic level

Query Syntax: `:TRIGger:PATH?`

Returned Format: `[[:TRIGger:PATH] {CHANne11|CHANne12|EXTErna1}<NL> (HP 54505B/54510B)`
`[[:TRIGger:PATH] {CHANne11|CHANne12|CHANne13|CHANne14}<NL> (HP 54506B/54512B)`

Example: `DIM Tp$[50]`
`OUTPUT 707;":TRIG:PATH?"`
`ENTER 707;Tp$`
`PRINT Tp$`

POLarity

POLarity

command/query

The `:TRIGGER:POLARITY` command is valid only in the **TV Trigger Mode**. This command sets the polarity for the trigger when the **STANDARD** is set to 525 or 625. The valid parameters for this command are **POSITIVE** and **NEGATIVE**.

The **POLARITY** query returns the current polarity setting.

Command Syntax: `:TRIGger:POLarity {POSitive | NEGative}`

Example:
`OUTPUT 707;":TRIGGER:MODE TV" !select trigger mode`
`OUTPUT 707;":TRIGGER:POL NEGATIVE"`

Query Syntax: `:TRIGger:POLarity?`

Returned Format: `[[:TRIGger:POLarity] {POSitive | NEGative}<NL>`

Example:
`DIM Tp$ [50]`
`OUTPUT 707;":TRIG:POL?"`
`ENTER 707;Tp$`
`PRINT Tp$`

PROBe
command/query

The `:TRIGGER:PROBE` command (HP 54505B/54510B only) is valid only for external triggers. It specifies the attenuation factor for the external trigger probe.

The `PROBE` query returns the current setting.

Command Syntax: `:TRIGger:PROBe <attenuation_factor>`

Where:

`<attenuation_factor>` ::= .9 to 1000 (exponential - NR3 format)

Example: `OUTPUT 707;":TRIGGER:SOURCE EXTERNAL" !select trigger source`
`OUTPUT 707;":TRIGGER:PROBE 10"`

Query Syntax: `:TRIGger:PROBe?`

Returned Format: `[[:TRIGger:PROBe] <attenuation_factor><NL>`

Where:

`<attenuation_factor>` ::= .9 to 1000 (exponential - NR3 format)

Example: `DIM Af$[50]`
`OUTPUT 707;":TRIG:PROB?"`
`ENTER 707;Af$`
`PRINT Af$`

QUALify

QUALify

command/query

The **:TRIGGER:QUALIFY** command is valid in the **Delay** and **TV Trigger Mode**. When you are in the Delay Trigger Mode, the parameters for this command are:

- EDGE
- PATTERN
- STATE

In the TV Trigger Mode, the parameters for this command are:

- LOW
- HIGH

The **QUALIFY** query returns the current setting of the **QUALIFY** command in the currently selected mode.

Command Syntax: `:TRIGger:QUALify <qualify_parameter>`

Where in Delay Trigger Mode:

`<qualify_parameter> ::= {EDGE | PATTERN | STATE}`

Where in TV Trigger Mode:

`<qualify_parameter> ::= {LOW | HIGH}`

Example: `OUTPUT 707;":TRIGGER:MODE DELAY" !select trigger mode
OUTPUT 707;":TRIGGER:QUALIFY PATT"`

Query Syntax: `:TRIGger:QUALify?`

Returned Format: `[[:TRIGger:QUALify] {EDGE | PATTERN | STATE | LOW | HIGH}<NL>`

Example: `DIM Tq$[50]
OUTPUT 707;":TRIG:QUALIFY?"
ENTER 707;Tq$
PRINT Tq$`

SENSitivity

command/query

The `:TRIGGER:SENSITIVITY` command sets the trigger sensitivity for the selected source. `NORMAL` corresponds to noise reject off and `LOW` corresponds to noise reject on.

The `SENSITIVITY` query returns the current sensitivity for the selected source.

Note 

Trigger sensitivity cannot be set for the `EXTERNAL` trigger source (HP 54505B/54510B only).

Command Syntax: `:TRIGger:SENSitivity {NORMa1 | LOW}`

Example: `OUTPUT 707;":TRIGGER:SOURCE CHANNEL1" !select trigger source`
`OUTPUT 707;":TRIGGER:SENSITIVITY LOW"`

Query Syntax: `:TRIGger:SENSitivity?`

Returned Format: `[[:TRIGger:SENSitivity] {NORMa1 | LOW}<NL>`

Example: `DIM Sens$[50]`
`OUTPUT 707;":TRIG:SENS?"`
`ENTER 707;Sens$`
`PRINT Sens$`

The :TRIGGER:SETUP query returns the current settings for the Trigger Subsystem commands.

Query Syntax: :TRIGger:SETup?

Returned Format: :TRIGger:COND <argument>;COUP {DC|DCF};DEL {TIME, <time_value>|EVEN, <event_value>};DEL:SLOP {POS|NEG};DEL:SOUR <source>;FIEL {1|2};GLIT:HOLD <holdoff_value>;GLIT:LEV <level>;GLIT:SOUR <source>;GLIT:WIDT {GT|LT}, <width>;HOLD {TIME, <holdoff_value>|EVEN, <event_argument>};LEV <level>;LINE <line_number>;LOG {HIGH|LOW|DONT};MODE {EDGE|PATT|STAT|DEL|TV|GLIT};OCC <occ_value>;OCC:SLOP {POS|NEG};OCC:SOUR <source>;PATH <source>;POL {POS|NEG};PROBE <attenuation_factor>;QUAL <qualify_parameter>;SENS {NORM|LOW};SLOP {NEG|POS};SOUR <source>;STAN {525|625|USER}<NL>

Where:

<argument> ::= dependent on trigger mode (refer to TRIGger:CONDition command)
 <time_value> ::= amount of delay from 30ns to 160ms (exp - NR3 format)
 <event_value> ::= number of events from 1 to 16000000 (int - NR1 format)
 <source> ::= CHAN1, CHAN2, EXT (HP 54505B/54510B)
 <source> ::= CHAN1|CHAN2|CHAN3|CHAN4 (HP 54506B/54512B)
 <holdoff_value> ::= 40 ns to 320 ms rounded to nearest 20 ns increment (exponential - NR3 format)
 <level> ::= trigger level in volts (exponential - NR3 format)
 <width> ::= 5 ns to 160 ms (exponential - NR3 format)
 <event_argument> ::= 1 to 16000000 (int - NR1 format)
 <attenuation_factor> ::= .9 to 1000 (exponential - NR3 format)<line_number> ::= 1 to 625 (depends on STANDARD and FIELD selection) (int - NR1 format)
 <occ_value> ::= 1 to 16000000 (integer - NR1 format)
 <qualify_parameter> ::= dependent on trigger mode (refer to TRIGger:QUALify command)

Example: DIM Stp\$[500]
 OUTPUT 707;":TRIG:SET?"
 ENTER 707;Stp\$
 PRINT Stp\$

Note



COUpling and PROBE are returned only if the oscilloscope is an HP 54505B/54510B.

SLOPe

command/query

The `:TRIGGER:SLOPE` command specifies the slope of the edge for the trigger. The SLOPE command is valid in the **Edge Trigger Mode**, **State Trigger Mode**, and **Delay Trigger Mode** when **EDGE** or **STATE** is selected as the qualifier.

The SLOPE query returns the current slope for the currently selected trigger mode.

Command Syntax: `:TRIGger:SLOPe {NEGative | POSitive}`

Example:

```
OUTPUT 707;":TRIGGER:MODE DELAY"      !select trigger mode
OUTPUT 707;":TRIGGER:QUALIFY EDGE"    !qualify trigger
OUTPUT 707;":TRIGGER:SLOPE POSITIVE"
```

Query Syntax: `:TRIGger:SLOPe?`

Returned Format: `[[:TRIGger:SLOPe] {POSitive | NEGative}<NL>`

Example:

```
DIM Ts$[50]
OUTPUT 707;":TRIG:SLOP?"
ENTER 707;Ts$
PRINT Ts$
```

SOURce

SOURce

command/query

The **:TRIGGER:SOURCE** command selects the channel that actually produces the trigger. The **SOURCE** command is valid in the **Edge Trigger Mode**, **State Trigger Mode**, **Delay Trigger Mode**, and **TV Trigger Mode**. In the **Delay Trigger Mode** this command is valid when **EDGE** or **STATE** is selected as the qualifier.

The **SOURCE** query returns the current source for the selected trigger mode.

Command Syntax: `:TRIGger:SOURce {CHANne11|CHANne12|EXTeRNA1|LINE} (HP 54505B/54510B)`
`:TRIGger:SOURce {CHANne11|CHANne12|CHANne13|CHANne14|LINE}`
`(HP 54506B/54512B)`

Example: `OUTPUT 707;":TRIGGER:SOURCE CHAN2"`

Query Syntax: `:TRIGger:SOURce?`

Returned Format: `[[:TRIGger:SOURce] {CHANne11|CHANne12|EXTeRNA1|LINE}<NL> (HP 54505B/54510B)`
`[[:TRIGger:SOURce] {CHANne11|CHANne12|CHANne13|CHANne14|LINE}<NL>`
`(HP 54506B/54512B)`

Example: `DIM Src$[30]`
`OUTPUT 707;":TRIGGER:SOURCE?"`
`ENTER 707;Src$`
`PRINT Src$`

STANDARD**command/query**

The **:TRIGGER:STANDARD** command selects the television signal standard to be used in the **TV Trigger Mode**. The valid parameters for this command are **525**, **625**, and **USER** (user defined).

The **STANDARD** query returns the currently selected standard.

Command Syntax: `:TRIGger:STANdard {525 | 625 | USER}`

Example: `OUTPUT 707;":TRIGGER:MODE TV" !select trigger mode
OUTPUT 707;":TRIGGER:STAN USER"`

Query Syntax: `:TRIGger:STANdard?`

Returned Format: `[[:TRIGger:STANdard] {525 | 625 | USER}<NL>`

Example: `DIM Ts$[50]
OUTPUT 707;":TRIG:STANDARD?"
ENTER 707;Ts$
PRINT Ts$`

Waveform Subsystem

Introduction

The WAVEFORM subsystem is used to transfer waveform data between a controller and the oscilloscope's waveform memories. This subsystem contains the following commands:

- DATA
- FORMat
- POINTs
- PREamble
- SOURce
- TYPE
- XINcrement
- XORigin
- XREFerence
- YINcrement
- YORigin
- YREFerence

Figure 18-1 lists the syntax diagrams for the Waveform subsystem commands.

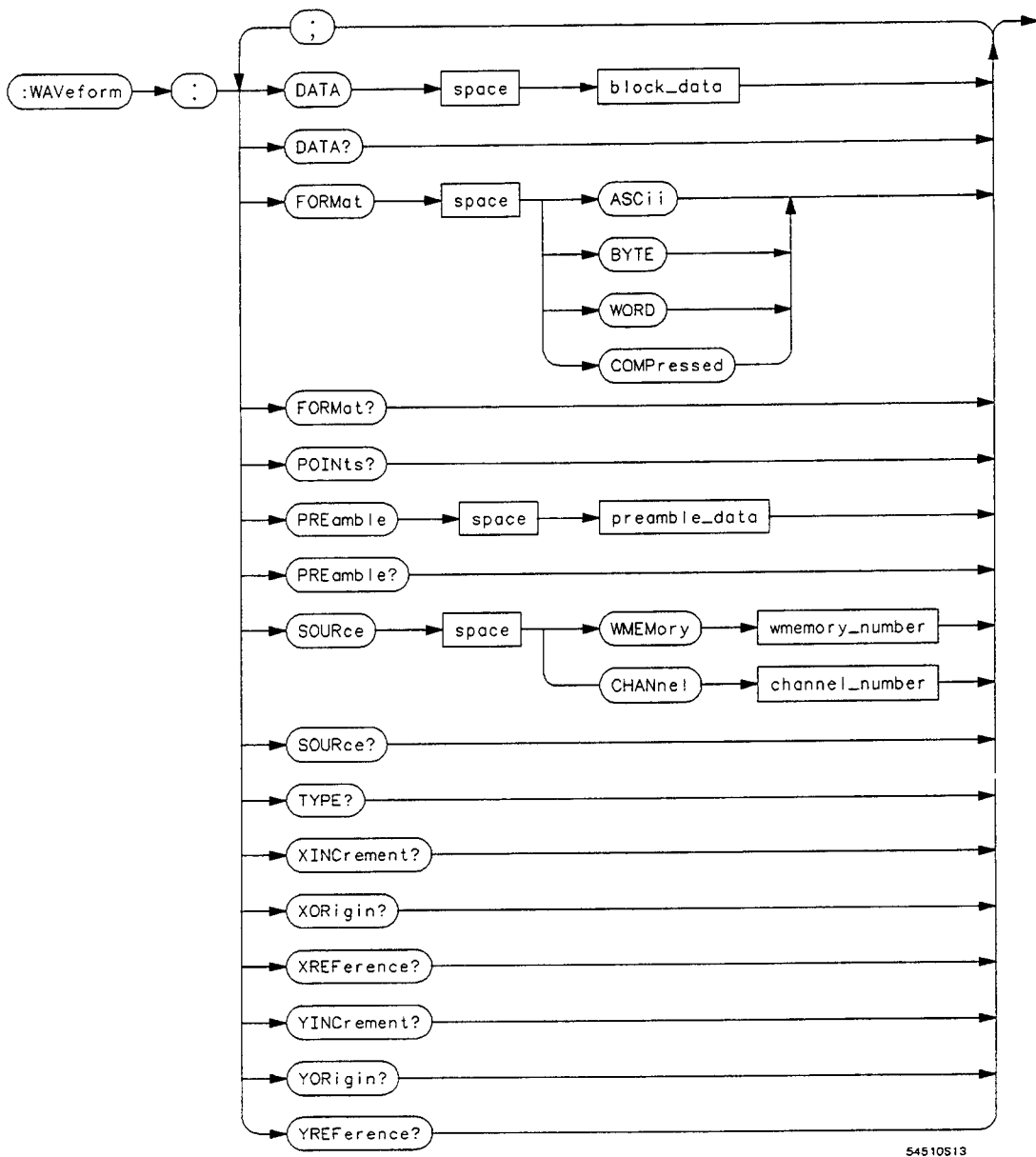


Figure 18-1. Waveform Subsystem Commands Syntax Diagram

| | |
|-------------------------|--|
| channel_number = | an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B). |
| block_data = | block data in IEEE 488.2 # format. |
| preamble_data = | refer to the PREAMBLE command. |
| wmemory_number = | an integer, 1 through 4. |

Figure 18-1. Waveform Subsystem Commands Syntax Diagram (continued)

Waveform Data and Preamble

The waveform record is actually contained in two portions: the waveform data and the preamble. The waveform data is the actual data acquired for each point in the specified source. The preamble contains the information for interpreting the waveform data, which includes the number of points acquired, format of acquired data, and type of acquired data. The preamble also contains the X and Y increments, origins, and references for the acquired data, so that the raw data can be translated to time and voltage values.

The values set in the preamble are determined when the :DIGITIZE command is executed or when the front-panel STORE key is pressed. The preamble values are based on the settings of variables in the ACQUIRE subsystem, or they are based on the front-panel setup when the STORE key is pressed.

Although the preamble values can be changed with a controller, the way the data is acquired cannot be changed. Changing the preamble values cannot change the type of data that was actually acquired, the number of points actually acquired, etc. Therefore, you must use extreme caution when changing any waveform preamble values to ensure the data is still useful. For example, setting POINTS in the preamble to a value different from the actual number of points in the waveform results in inaccurate data.

The waveform data and preamble must be read by the controller or sent to the oscilloscope with two separate commands, DATA and PREAMBLE.

Sending consecutive :DIGITIZE commands may improve the data throughput. Refer to the Root Level Command :DIGITIZE in chapter 7 for more information.

Data Acquisition Types

There are four types of waveform acquisition that can be selected with the :ACQUIRE:TYPE command: NORMAL, AVERAGE, ENVELOPE, and RAWDATA. When the data is acquired using the DIGITIZE command, the data is placed in the channel buffer of the specified source. However, during a RAWDATA acquisition, data is placed in a special rawdata buffer.

After a DIGITIZE command, the instrument is stopped. If the instrument is restarted, over the HP-IB or the front panel, the data acquired with the DIGITIZE command is overwritten.

With the exception of the rawdata mode, waveforms in the oscilloscope are either 500 or 8000 points long. The 8000 point record is available only in the real-time mode. The 500 point record is available in the repetitive or real-time mode.

Realtime Mode 8000 Points

A realtime waveform consists of 8000 sample points. If the following conditions are met, all 8000 points are transferred over the bus:

- :ACQUIRE:POINTS is set to 8000.
- :WAVEFORM:DATA is requested.
- The specific channel or waveform memory contains realtime data.

During normal operation, only 500 or less of these 8000 points are displayed on screen. To get the points on screen, send the command :ACQUIRE:POINTS 500.

500 Points

When `:ACQUIRE:POINTS` is set to 500, the on-screen time is divided into 500 time buckets as in the repetitive mode. If less than 500 points of the realtime waveform fall into the on-screen time, an interpolation filter is applied to create exactly 500 points. Pan and zoom, as described in the front-panel reference, can be used to display more than 500 points on screen. In this case, the 500 points are the last points placed in each time bucket.

Repetitive Modes

In the repetitive mode, the on-screen time is divided into 500 time buckets and all waveforms contain 500 points.

Normal

Normal data consists of the last data point (hit) in each time bucket. This data is transmitted over 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. Only the magnitude values of each data point are transmitted. The first voltage value corresponds to the first time bucket on the left of the screen and the last value corresponds to the next to last time bucket on the right side of the screen. Time buckets that do not have data in them return -1 .

The time values for each data point correspond to the position of the data point in the data array. These values are not transmitted.

Average

Average data consists of the average of the first n hits in a time bucket, where n is the value returned by the `ACQUIRE:COUNT` query. Time buckets that have fewer than n hits return the average of what data they do have. If the `:ACQUIRE:COMPLETE` parameter is set to 100%, then each time bucket must contain the number of data hits specified with the `:ACQUIRE:COUNT` command. If a time bucket does not have any data in it, it returns -1 . This data is transmitted over the 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 corresponds to one point away from the right side of the screen.

Envelope

Envelope data consists of two arrays of data: one containing the minimum hits in each time bucket, and the other containing the maximum hits in each time bucket. If a time bucket does not have any hits in it, then -1 is returned for both the minimum and maximum values. The two arrays are transmitted one at a time over the HP-IB linearly, starting with time bucket 0 (on the left side of the screen) and proceeding through time bucket $m-1$, where m is the value returned by the WAVEFORM:POINTS query. The array with the minimum values is sent first. The first value of each array corresponds to the data point on the left of the screen. The last value is one data point away from the right side of the screen.

The data is transferred from the channel buffer to a controller using the WAVEFORM:DATA query. Data is transferred into the instrument from a controller using the WAVEFORM:DATA command. Envelope data can be transferred into Waveform Memories 1 and 2, if WMEMORY 1 is specified as the source. If WMEMORY 3 is specified as the source, the envelope data can be transferred into Waveform Memories 3 and 4.

The data is transferred as two arrays. If Waveform Memory 1 is specified as the source, the first array is transferred into Waveform Memory 1 and the second array is transferred into Waveform Memory 2. If Waveform Memory 3 is specified as the source, the first array is transferred into Waveform Memory 3 and the second array is transferred into Waveform Memory 4. The data type is then changed to normal for each of the waveform memories.

Rawdata Rawdata is acquired with special routines that are optimized for quick fiso unloading. These routines do not filter the data.

While acquiring data in the Rawdata mode, data is stored as uncalibrated 8-bit Gray code data in a large buffer. When all acquisitions are complete, the buffer is translated into unfiltered, calibrated 16-bit binary data and sent over the bus in the WORD format. The :WAVEFORM:FORMAT command has no effect in the Rawdata mode.

The RAWDATA command has three parameters: length, acquisitions, and mode. Length specifies the number of points of each acquisition. Acquisitions specify the number of acquisitions to be taken in a single digitize operation. Mode specifies how the data is processed. The data is transferred from the oscilloscope in a single IEEE 488.2 data block. The number of bytes in the block can be determined with the following equation:

$$\text{Number of Bytes} = \text{Acquisitions} \times (\text{Length} \times 2 + 8)$$

The data block consists of two arrays. The first array consists of double precision 64-bit floating point numbers. This array contains the xorigin values of the waveform records to follow. It can read directly into a double precision real array in a controller allocated by the BASIC command `ALLOCATE REAL Xorigins(1:Acquisitions)`. The xorigin values are also available in ASCII format using the `:WAVEFORM:XORIGIN` query. These xorigin values are accurate to within the timing resolution of the oscilloscope. A less accurate single xorigin value to within one sample time is supplied in the preamble for the rawdata record. All xorigin values in the array will be greater than or equal to the single xorigin value in the preamble.

The second array consists of 16-bit integer numbers. This data is transmitted in a linear fashion over HP-IB. It starts with sample zero of the first acquisition and continues through sample length-1 of the this acquisition. Then it continues in a similar fashion with sample zero through sample length-1 of the each following acquisition through the last acquisition. This array can be read directly into a two dimensional integer array allocated by the BASIC command `ALLOCATE INTEGER Waveforms(1:Acquisitions, 1:Points)`.

Data Conversion

Data sent from the oscilloscope must be scaled for useful interpretation. The values used to interpret the data are the X and Y references, X and Y origins, and X and Y increments. These values are read from the waveform preamble.

Conversion from Data Value to Voltage

The following formula converts a data value from waveform memories 1 through 4, or channels 1 or 2, to a voltage value:

$$\text{voltage} = [(\text{data value} - \text{yreference}) * \text{yincrement}] + \text{yorigin}.$$

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 fourth data point sent with XORIGIN = 16 ns, XREFERENCE = 0, and XINCREMENT = 2 ns can be calculated using the following formula:

$$\text{time} = [(\text{data point number} - \text{xreference}) * \text{xincrement}] + \text{xorigin},$$

This would result in the following calculation:

$$\text{time} = [(3 - 0) * 2 \text{ ns}] + 16 \text{ ns} = 22 \text{ ns}.$$

Data Format for HP-IB Transfer

There are four formats for transferring waveform data over the HP-IB: WORD, BYTE, COMPRESSED, and ASCII.

WORD, BYTE, and COMPRESSED formatted waveform records are transmitted using the arbitrary block program data format specified in IEEE 488.2. ASCII format block data does not use a block header.

When you use the block data format, the ASCII character string "#8<DD...D>" is sent prior to sending the actual data. The 8 indicates how many D's follow. The D's are ASCII numbers which indicate how many data bytes follow.

For example, if 500 points were acquired and the BYTE format was specified, the block header "#800000500" would be sent. The 8 indicates that eight length bytes follow, and 500 indicates that 500 binary data bytes follow.

WORD Format

In the WORD format, the number of data bytes is twice the number of words (data points). The number of data points is the value returned by the :WAVEFORM:POINTS? query. The number of data bytes is followed by a sequence of bytes representing the data points, with the most significant byte of each word transmitted first. In this format the data is shifted so that the most significant bit after the sign bit contains the most significant bit of the data. If there is a hole in the data, the hole is represented by the 16-bit value of -1. The range of data in the WORD format is from 0 to 32640.

WORD format is useful in applications where the information is read directly into an integer array in a controller.

WORD and ASCII formatted data returns the most accurate data values.

BYTE Format The BYTE format allows only seven bits to be used to represent the voltage values, with the first bit being the sign bit. If there is a hole in the data, the hole is represented by a value of -1 .

The BYTE-formatted data transfers over the HP-IB faster than WORD-formatted data, since one byte per point is transferred in BYTE format and two bytes per point are transferred in WORD format. The BYTE-formatted data has less resolution than WORD-formatted data.

COMPRESSED Format The number of bytes transmitted when the format is COMPRESSED is the same as the value returned by the WAVEFORM:POINTS? query.

Eight bits of resolution are retained in the COMPRESSED format. So that a hole in the data may be represented, a data value of 255 is mapped to 254, and 255 is used to represent a hole. This mode gives greater vertical precision than BYTE-formatted data, with faster transfer times than WORD-formatted data. On the other hand, the COMPRESSED mode probably requires more time after the transfer for data to be unpacked.

ASCII Format Waveform records in the ASCII format are transmitted one value at a time, separated by a comma. The data values transmitted are the same as the values sent in the WORD FORMAT except that they are converted to an integer ASCII format (six or less characters) before being sent over the HP-IB.

ASCII values cannot be used to transfer data into a controller.

DATA

command/query

The `:WAVEFORM:DATA` command transfers a waveform data record to the instrument over the HP-IB and stores it in the previously specified waveform memory. The waveform memory is specified with the `:WAVEFORM:SOURCE` command. Only waveform memories can have waveform data sent to them.



The oscilloscope only accepts 500 or 8000 point records. If you try to send variable length records (RAWDATA acquisition type) back into the oscilloscope, an error message appears on the screen.

The format of the data being sent must match the format previously specified by the waveforms preamble for the destination memory. The `:WAVEFORM:DATA` command does not accept ASCII data. However, the ASCII format can be specified for data output.

The DATA query outputs a waveform record to the controller over the HP-IB that is stored in a waveform memory or channel buffer previously specified with the `:WAVEFORM:SOURCE` command.

Command Syntax: `:WAVEform:DATA <binary block data in # format>`

Query Syntax: `:WAVEform:DATA?`

Returned Format: `[:WAVEform:DATA] <binary block length bytes><binary block><NL>`

DATA

Example: The following program moves data from the oscilloscope to the controller and then back to the oscilloscope with the :WAVEFORM:DATA query and command.

```
5   *RST                                ! Reset the oscilloscope
10  CLEAR 707                          ! Initialize instrument interface
20  OUTPUT 707;":TIMEBASE:SAMPLE REPETITIVE" ! Select sample mode
30  ! Setup the Acquire subsystem
40  OUTPUT 707;":ACQUIRE:TYPE NORMAL"    ! Display mode to NORMAL
50  OUTPUT 707;":ACQUIRE:COUNT 1"
60  OUTPUT 707;":ACQUIRE:POINTS 500"
70  OUTPUT 707;":DIGITIZE CHANNEL1"      ! Acquire data
80  OUTPUT 707;":SYSTEM:HEADER OFF"      ! Headers off
90  OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1" ! Channel 1 as waveform source
100 OUTPUT 707;":WAVEFORM:FORMAT WORD"    ! WORD format
110 OUTPUT 707;":WAVEFORM:DATA?"
120 ENTER 707 USING "#,2A,8D";Headers$,Bytes ! Read length byte
130 Length=Bytes
140 Length=Length/2
150 ALLOCATE INTEGER Waveform(1:Length)
160 ENTER 707 USING "#,W";Waveform(*)     ! Enter waveform data to integer array
170 ENTER 707 USING "-K,B";End$
180 DIM Preamble$[200]                   ! Dimension variable
190 OUTPUT 707;":WAVEFORM:PREAMBLE?"     ! Output waveform preamble to controller
200 ENTER 707 USING "-K";Preamble$
210 OUTPUT 707;":WAVEFORM:SOURCE WMEMORY4" ! Waveform memory 4 as source
220 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Preamble$! Send preamble
    ! from controller to WMEMORY 4
230 OUTPUT 707 USING "#,K";":WAVEFORM:DATA #800001000"! Send header
240 OUTPUT 707 USING "W";Waveform(*)     ! Send waveform data to WMEMORY 4
250 OUTPUT 707;":BLANK CHANNEL1"        ! Turn channel 1 off
260 OUTPUT 707;":VIEW WMEMORY4"         ! Turn WMEMORY 4 on
270  END
```

Note



In program line 220, the space after :WAVEFORM:PREAMBLE and before the quotation mark is required.

FORMat

command/query

The `:WAVEFORM:FORMAT` command sets the data transmission mode for waveform data output. This command controls how the data is formatted on the HP-IB when sent from the oscilloscope.

When the ASCII mode is selected, the data consists of ASCII digits with each data value separated by a comma. The ASCII mode cannot be used to transfer data into a controller.

WORD formatted data transfers as 16-bit binary integers in two bytes, with the most significant byte of each word sent first.

BYTE and COMPRESSED formatted data is transferred as 8-bit bytes.

The `FORMAT` query returns the current output format for the transfer of waveform data.



The `:WAVEFORM:FORMAT` command has no effect in the Rawdata mode. In this mode, data is always transferred in the WORD format.

Command Syntax: `:WAVEform:FORMat {ASCIi | WORD | BYTE | COMPressed}`

Example: `OUTPUT 707;":WAVEFORM:FORMAT WORD"`

Query Syntax: `:WAVEform:FORMat?`

Returned Format: `[[:WAVEform:FORMat] {ASCIi | WORD | BYTE | COMPressed}<NL>`

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

POINTs

POINTs

query

The :WAVEFORM:POINTS query returns the points value in the currently selected waveform preamble. The points value is the number of time buckets contained in the waveform selected with the WAVEFORM:SOURCE command.

Query Syntax: :WAVEform:POINTs?

Returned Format: [:WAVEform:POINTs] {500 | 8000}<NL>

Example: Dim Pts\$[50]
OUTPUT 707;":WAV:POINTS?"
ENTER 707;Pts\$
PRINT Pts\$

PREamble

command/query

The :WAVEFORM:PREAMBLE command sends a waveform preamble to a previously selected waveform memory in the instrument.

The PREAMBLE query sends a waveform preamble to the controller from the waveform source.

In the Rawdata mode, the format is always 2 for WORD, and the number of acquisitions is returned by the count parameter. The count parameter returns 1 in all other modes.

Command Syntax: :WAVEform:PREamble <preamble block>

Where:

<preamble block> ::= <format NR1>,<type NR1>,<points NR1>,<count NR1>,<xincrement NR3>,<xorigin NR3>,<xreference NR3>,<yincrement NR3>,<yorigin NR3>,<yreference NR3>

Query Syntax: :WAVEform:PREamble?

Returned Format: [:WAVEform:PREamble] <preamble block><NL>

Where:

<preamble block> ::= <format NR1>,<type NR1>,<points NR1>,<count NR1>,<xincrement NR3>,<xorigin NR3>,<xreference NR1>,<yincrement NR3>,<yorigin NR3>,<yreference NR1>

Where:

<format> ::= 0 for ASCII format
1 for BYTE format
2 for WORD format
4 for COMPRESSED format

PREamble

```
<type> ::=      0 for INVALID type
                 1 for Repetitive NORMAL type or Realtime
                 2 for Repetitive AVERAGE type
                 3 for Repetitive ENVELOPE type
                 4 for RAWDATA type
```

Example: This example program uses both the command and query form of the **PREAMBLE** command. First the preamble is queried (output to the controller). Then, the preamble is returned to a previously selected waveform memory.

```
10 OUTPUT 707;":WAVEFORM:SOURCE WMEM1"
20 DIM Pre$[120]
30 OUTPUT 707;":SYSTEM:HEADER OFF"
40 OUTPUT 707;":WAVEFORM:PREAMBLE?"
50 ENTER 707 USING "-K";Pre$
60 OUTPUT 707 USING "#,K";":WAVEFORM:PREAMBLE ";Pre$
70 END
```

Note

In line 60 of the program example, a space is inserted between the word "PREAMBLE " and the closed quotation mark. This space must be inside the quotation mark because in this format (#,K) the data is packed together. Failure to add the space produces a word that is not a proper command word.

Example: The following program example places the preamble in a numeric array.

```
10 OUTPUT 707;":WAVEFORM:SOURCE WMEM1"
20 DIM Preamble(1:10)
30 OUTPUT 707;":SYSTEM:HEADER OFF"
40 OUTPUT 707;":WAVEFORM:PREAMBLE?"
50 ENTER 707 ;Preamble(*)
60 OUTPUT 707;":WAVEFORM:PREAMBLE ";Preamble(*)
70 END
```

SOURce

command/query

The :WAVEFORM:SOURCE command selects the channel or waveform memory to be used as the source for the waveform commands. This command also selects the destination for data being transferred over the bus.

The SOURCE query returns the currently selected source for the waveform commands.

Command Syntax: :WAVeform:SOURce {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

Example: OUTPUT 707;":WAV:SOURCE WMEMORY3"

Query Syntax: :WAVeform:SOURce?

Returned Format: [:WAVeform:SOURce] {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}<NL>

Example:
 DIM Src\$[30]
 OUTPUT 707;":WAVEFORM:SOURCE?"
 ENTER 707;Src\$
 PRINT Src\$

Note

CHANnel3 and CHANnel4 are only available if the oscilloscope is an HP 54506B/54512B.

TYPE

TYPE

query

The :WAVEFORM:TYPE query returns the data type for the previously specified waveform source.

Query Syntax: :WAVEform:TYPE?

Returned Format: [:WAVEform:TYPE] {INValid | AVERage | ENVelope | NORMal | RAWData}<NL>

Example:
DIM Typ\$[30]
OUTPUT 707;":WAVEFORM:TYPE?"
ENTER 707;Typ\$
PRINT Typ\$

XINCrement**query**

The `:WAVEFORM:XINCREMENT` query returns the current x-increment value in the preamble for the current specified source. This value is the time difference between consecutive data points for NORMAL, AVERAGE, ENVELOPE, or RAWDATA.

For a 500 point waveform, the x-increment pixel width is in seconds. For 8000 point waveforms and RAWDATA waveforms, the xincrement value is the sample period. Therefore, these waveforms never contain the interpolated data points.

Query Syntax: `:WAVeform:XINCrement?`

Returned Format: `[:WAVeform:XINCrement] <value><NL>`

Where:

`<value>` ::= x-increment in the current preamble (exponential - NR3 format)

Example:

```
DIM Xin$[50]
OUTPUT 707;":WAV:XINCREMENT?"
ENTER 707;Xin$
PRINT Xin$
```

XORigin

XORigin

query

The :WAVEFORM:XORIGIN query returns the current x-origin value in the preamble for the current specified source. For a 500 point record, the x-origin value is the time of the first data point in the memory with respect to the trigger point. For an 8000 point record, the x-origin value is the point referenced by x-reference.

In the RAWDATA mode, the :WAVEFORM:XORIGIN query may return multiple values depending on the number of acquisitions specified. In this mode, the x-origin value is the time of the first data point in each acquisition.

Note

In the rawdata mode, the x-origin value returned by the preamble is the minimum time of the first point in memory that occurs in any acquired data record.

Query Syntax: :WAVEform:XORigin?

Returned Format: [:WAVEform:XORigin] <value>[,<value>]...<NL>

Where:

<value> ::= x-origin value in the current preamble (exponential - NR3 format)

Example:
DIM Xr\$[50]
OUTPUT 707;":WAV:XORIGIN?"
ENTER 707;Xr\$
PRINT Xr\$

XREFerence**query**

The :WAVEFORM:XREFERENCE query returns the current x-reference value in the preamble for the current specified source. This value specifies the number of the first point on the screen. For 500 point waveforms and the rawdata mode, xreference is zero. For 8000 point waveforms, xreference is the number of the first data point appearing on screen. This x-reference value is between zero and 7999. The exact number depends on the sweep speed; left, center, or right reference; and the pan and zoom if the waveform is stopped.

Query Syntax: :WAVeform:XREFerence?

Returned Format: [:WAVeform:XREFerence] <value><NL>

Where:

<value> ::= x-reference value in the current preamble (integer - NR1 format)

Example:

```
DIM Xrf$[50]
OUTPUT 707;":WAV:XREFERENCE?"
ENTER 707;Xrf$
PRINT Xrf$
```

YINCrement

YINCrement

query

The `:WAVEFORM:YINCREMENT` query returns the current y-increment value in the preamble for the current specified source. This value is the voltage difference between consecutive data points.

Query Syntax: `:WAVEform:YINCrement?`

Returned Format: `[:WAVEform:YINCrement] <value><NL>`

Where:

`<value>` ::= y-increment value in the current preamble (exponential - NR3 format)

Example:

```
DIM Yin$[50]
OUTPUT 707;":WAV:YINCREMENT?"
ENTER 707;Yin$
PRINT Yin$
```

YORigin**query**

The `:WAVEFORM:YORIGIN` query returns the current y-origin value in the preamble for the current specified source. This value is the voltage at the center of the screen.

Query Syntax: `:WAVEform:YORigin?`

Returned Format: `[:WAVEform:YORigin] <value><NL>`

Where:

`<value>` ::= y-origin in the current preamble (exponential - NR3 format)

Example:

```
Dim Yr$[50]
OUTPUT 707;":WAV:YORIGIN?"
ENTER 707;Yr$
PRINT Yr$
```

YREFerence

YREFerence

query

The :WAVEFORM:YREFERENCE query returns the current y-reference value in the preamble for the current specified source. This value specifies the data point where the y-origin occurs.

Query Syntax: :WAVEform:YREFerence?

Returned Format: [:WAVEform:YREFerence] <value><NL>

Where:

<value> ::= y-reference value in the current preamble (integer - NR1 format)

Example:

```
DIM Yrf$[50]
OUTPUT 707;":WAV:YREFERENCE?"
ENTER 707;Yrf$
PRINT Yrf$
```

Waveform Memory Subsystem

Introduction

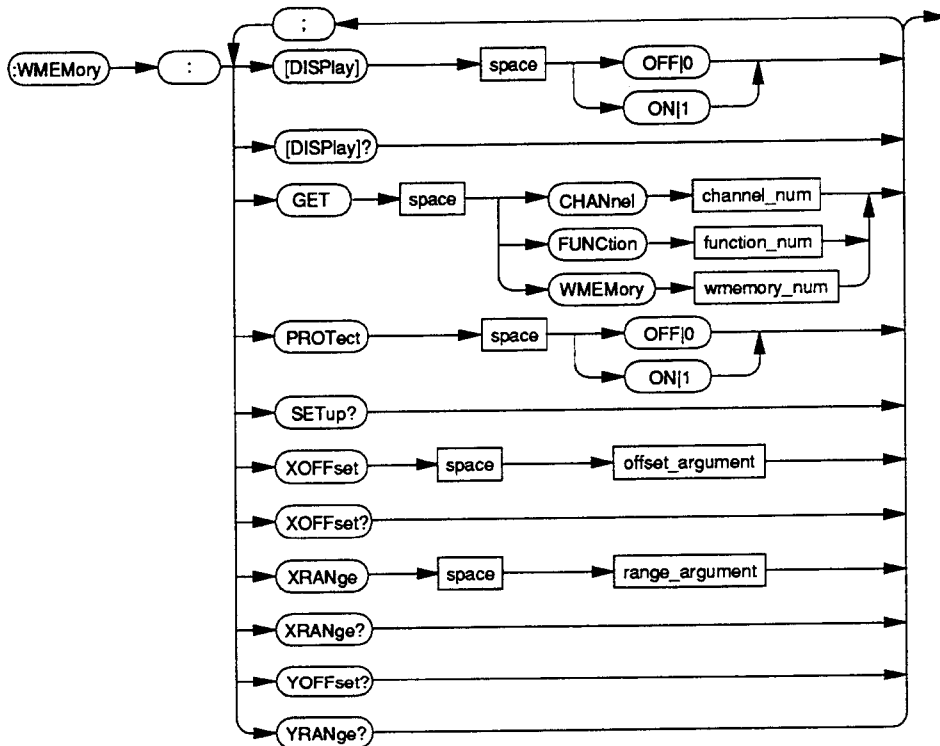
The WMEMORY subsystem commands control the four nonvolatile waveform memories of the oscilloscope. Each memory is independently programmable for protect, offset, and range functions. The memory number specified in the command selects the memory that is affected by the command.

The channel displays are toggled on and off with the root level commands VIEW and BLANK, or using the WMEMORY[:DISPLAY] command.

The Waveform Memory subsystem contains the following commands:

- DISPlay
- GET
- PROtect
- SETup
- XOFFset
- XRANge
- YOFFset
- YRANge

Figure 19-1 lists the syntax diagrams for the Waveform Memory subsystem commands.



- channel_number =** an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B).
- function_number =** an integer, 1 through 4.
- wmemory_number =** an integer, 1 through 4.
- offset_argument =** a real number, the maximum value depends on the range selected.
- range_argument =** a real number, 10 ns through 50 s (in a 1, 2, 5 sequence).

Figure 19-1. Waveform Memory Subsystem Commands Syntax Diagram

[DISPlay]**command/query**

The `:WMEMORY<N>[:DISPLAY]` command controls the individual waveform memory displays. On starts displaying, and off stops displaying the memory selected.

The `DISPLAY` query returns the current setting of the command.

Note Because `[:DISPlay]` is an implied command, only the parameters need be sent to turn a selected waveform memory on or off. For example, to turn waveform memory 1 display on send either `":WMEM1:DISP ON"` or `":WMEM1 ON"`.

Command Syntax: `:WMEmory<N>[:DISPlay] {{ON | 1} | {OFF | 0}}`

Where:

`<N> ::= 1 through 4`

Example: `OUTPUT 707;":WMEM2:DISP ON"`

Query Syntax: `:WMEmory<N>[:DISPlay]?`

Where:

`<N> ::= 1 through 4`

Returned Format: `[:WMEmory<N>[:DISPlay]] {1 | 0}<NL>`

Example:
`DIM Disp$[50]`
`OUTPUT 707;":WMEM1?"`
`ENTER 707;Disp$`
`PRINT Disp$`

GET

GET

command

The :WMEMORY<N>:GET command transfers selected channel, function, or waveform memory data into the waveform memory specified. When executed, an immediate erase of the waveform memory selected is performed, followed by a write of the new data. An error is generated if the destination waveform memory write protect is set to ON.

Command Syntax: :WMEmory<N>:GET {CHANnel{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4} | FUNction{1 | 2 | 3 | 4}}

Example: OUTPUT 707;":WMEM1:GET CHANNEL3"

Note CHANnel3 and CHANnel4 are only available if the oscilloscope is an HP 54506B/54512B.

PROTECT

command/query

The :WMEMORY<N> PROTECT command controls the write protect state of the selected nonvolatile memory. On enables the selected waveform memories write protect state. When on, any attempted save will generate an error, and the contents remain unchanged.

The PROTECT query returns the current setting of the command.

Command Syntax: :WMemory<N>PROTECT {{ON | 1} | {OFF | 0}}

Where:

<N> ::= 1 through 4

Example: OUTPUT 707;":WMEM2:PROT ON"

Query Syntax: :WMemory<N>PROTECT?

Where:

<N> ::= 1 through 4

Returned Format: [:WMemory<N>PROTECT] {1 | 0}<NL>

Example:
 DIM Disp\$[50]
 OUTPUT 707;":WMEM:PROT?"
 ENTER 707;Disp\$
 PRINT Disp\$

SETup

SETup

query

The :WMEMORY <N> :SETUP query returns the current settings for the WMEMory Subsystem commands.

Query Syntax: :WMEemory<N>:SETup?

Returned Format: :WMEemory<N>:DISPlay{ON | OFF};PROTECT{ON | OFF};XRANGE <xrange>;XOFFset <xoffset>;YRANGE <xrange>;YOFFset <xoffset><NL>

Where:

<N> ::= 1 through 4

<xrange> ::= 10 ns to 50 sec(exponential - NR3 format)

<xoffset> ::= time from trigger to display reference in seconds. Display reference is left, center, or right (exponential - NR3 format)

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

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

Example: DIM Stp\$[300]
OUTPUT 707;":WMEM1:SET?"
ENTER 707;Stp\$
PRINT Stp\$

XOFFset**command/query**

The :WMEMORY <N>:XOFFSET command sets the time base delay for the waveform memory selected, on 8000 point waveforms in realtime mode. This delay is the internal time between the trigger event and the on-screen delay reference point. The delay reference point is the left edge of the display, the right edge of the display, or the center of the display, and is set with the :TIMEBASE:REFERENCE command when the waveform was stored.

The XOFFSET query returns the current offset value.

Command Syntax: :WMemory<N>:XOFFset <xoffset>

Where:

<N> ::= 1 through 4

<xoffset> ::= time in seconds from trigger to the on screen delay reference point. The maximum value depends on the :WMemory:XRANGE setting.

Example: OUTPUT 707;":WMEM:XOFF 2MS"

Query Syntax: :WMemory<N>:XOFFset?

Returned Format: [:WMemory<N>:XOFFset] <xoffset><NL>

Where:

<N> ::= 1 through 4

<xoffset> ::= time from trigger to display reference in seconds. Display reference is left, center, or right (exponential - NR3 format)

Example: DIM Xofst\$[50]
OUTPUT 707;":WMEM1:XOFF?"
ENTER 707;Xofst\$
PRINT Xofst\$

XRANge

XRANge

command/query

The `:WMEMORY <N> :XRANGE` command sets the full-scale horizontal time in seconds for the waveform memory selected, on 8000 point waveforms in realtime mode. The `XRANGE` value is ten times the time per division.

The `XRANGE` query returns the current full-scale range value.

Command Syntax: `:WMEmory<N>:XRANge <xrange>`

Where:

`<N> ::= 1 through 4`

`<xrange> ::= 10 ns to 50 s in a 1,2,5 sequence"`

Example: `OUTPUT 707;":WMEM:RAN 100 MS"`

Query Syntax: `:WMEmory<N>:XRANge?`

Returned Format: `[:WMEmory<N>:XRANge] <xrange><NL>`

Where:

`<N> ::= 1 through 4`

`<xrange> ::= 10 ns to 50 s (exponential - NR3 format)`

Example:
`DIM Xrng$[50]
OUTPUT 707;":WMEMORY:XRANGE?"
ENTER 707;Xrng$
PRINT Xrng$`

YOFFset

query

The :WMEMORY<N>:YOFFSET query returns the voltage that is represented at center screen for the selected waveform memory.

Query Syntax: :WMemory<N>:YOFFset?

Returned Format: [:WMemory<N>:YOFFset] <yoffset><NL>

Where:

<N> ::= 1 through 4

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

Example:

```
DIM Yofst$[50]
OUTPUT 707;":WMEMORY2:YOFFSET?"
ENTER 707;Yofst$
PRINT Yofst$
```


YRANge

YRANge

query

The :WMEMORY <N>:YRANGE query returns the full-scale vertical axis of the selected waveform memory.

Query Syntax: :WMemory<N>:YRANGE?

Returned Format: [:WMemory<N>:YRANge] <yrange><NL>

Where:

<N> ::= 1 through 4

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

Example:
DIM Yrng\$[50]
OUTPUT 707;" :WMEM2:YRAN?"
ENTER 707;Yrng\$
PRINT Yrng\$

Pixel Memory Subsystem

Introduction

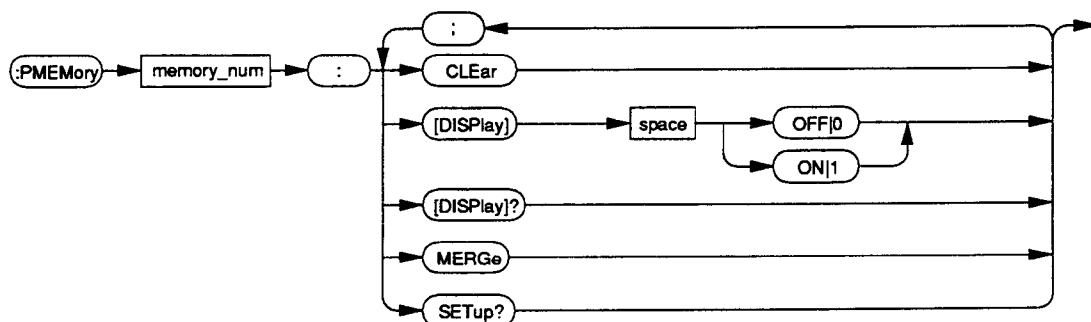
The PMEMORY subsystem commands control the two volatile pixel memories of the oscilloscope. The memory number specified in the command selects the memory that is affected by the command.

The pixel displays are toggled on and off with the root level commands VIEW and BLANK, or using the PMEMORY[:DISPLAY] command.

The Pixel Memory subsystem contains the following commands:

- CLEAr
- DISPlay
- MERGe
- SETUp

Figure 20-1 lists the syntax diagrams for the Pixel Memory subsystem commands.



memory_number = an integer, 1 or 2.

Figure 20-1. Pixel Memory Subsystem Commands Syntax Diagram

CLEAr

CLEAr

command

The `:PMEMORY<N>:CLEAR` command clears the contents of the selected pixel memory.

Command Syntax: `:PMEMy<N>:CLEAr`

Where:

`<N> ::= 1 or 2`

Example: `OUTPUT 707;":PMEM1:CLE"`

[DISPlay]

command/query

The `:PMEMORY<N>[:DISPLAY]` command controls the individual pixel memory displays. On displays, and off blanks the memory selected.

The `DISPLAY` query returns the current setting of the command.

Note Because `[:DISPlay]` is an implied command, only the parameters need be sent to turn a selected waveform memory on or off. For example, to turn waveform memory 1 display on send either `":PMEM1:DISP ON"` or `":PMEM1 ON"`.

Command Syntax: `:PMEMory<N>[:DISPlay] {{ON | 1} | {OFF | 0}}`

Where:

`<N> ::= 1 or 2`

Example: `OUTPUT 707;":PMEM2:DISP ON"`

Query Syntax: `:PMEMory<N>[:DISPlay]?`

Where:

`<N> ::= 1 or 2`

Returned Format: `[:PMEMory<N>[:DISPlay]] {1 | 0}<NL>`

Example:
`DIM Disp$[50]`
`OUTPUT 707;":PMEM1?"`
`ENTER 707;Disp$`
`PRINT Disp$`

MERGe

MERGe

command

The `:PMEMORY<N>:MERGe` command adds the contents of the active display to the current contents of the specified pixel memory.

This command is similar to the MERGe root command.

Command Syntax: `:PMEMy<N>:MERGe`

Where:

`<N> ::= 1 or 2`

Example: `OUTPUT 707;":PME2:MERG"`

SETup**query**

The `:PMEMORY<N>:SETUP` query returns the current settings for the PMemory Subsystem commands.

Query Syntax: `:PMEemory<N>:SETup?`

Returned Format: `:PMEemory<N>:DISP lay{ON | OFF}<NL>`

Example:
DIM Stp\$[30]
OUTPUT 707;":PMEEM1:SET?"
ENTER 707;Stp\$
PRINT Stp\$

Multiple Memory Subsystem

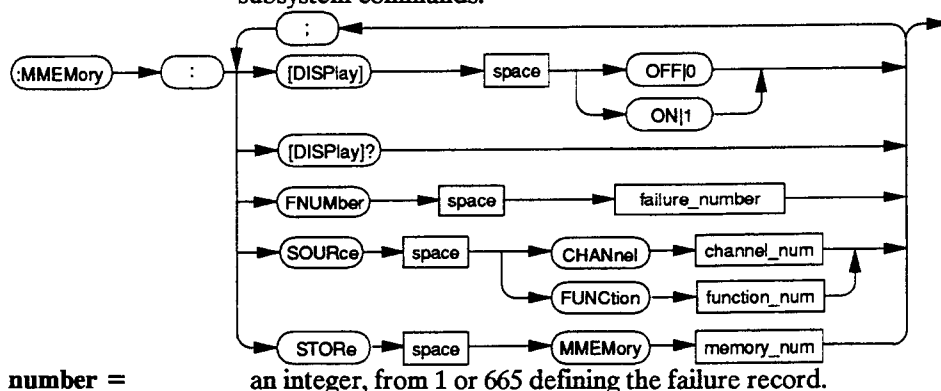
Introduction

The MMEMORY subsystem commands control the up to 665 volatile multiple memories of the oscilloscope. Multiple memories contain 501 points of waveform information resulting from the compare and limit test violations. Failure data stored in multiple memory can be displayed on screen, or transferred to nonvolatile waveform memory.

The Multiple Memory subsystem contains the following commands:

- DISPlay
- FNUMber
- SOURce
- STORe

Figure 21-1 lists the syntax diagrams for the Multiple Memory subsystem commands.



failure_number =

an integer, from 1 or 665 defining the failure record.

channel_number =

an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B).

function_number =

an integer, 1 through 4.

memory_number =

an integer, 1 through 4.

Figure 21-1. Multiple Memory Subsystem Commands Syntax Diagram

[DISPlay]

[DISPlay]

command/query

The :MMEMORY[:DISPLAY] command controls the multiple memory display. On allows failure data to be displayed on the screen. Specific failure data is selected using the MMEMORY:FNUMBER and :MMEMORY:SOURCE commands.

The DISPLAY query returns the current setting of the command.

Note

Because [:DISPlay] is an implied command, only the parameters need be sent to turn the multiple memory on or off. For example, to turn on the multiple memory display, send either ":MMEM:DISP ON" or ":MMEM ON". Valid data MUST be present before selecting multiple memory or an error will be generated.

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

Example: OUTPUT 707;":MMEM:DISP ON"

Query Syntax: :MMEMory[:DISPlay]?

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

Example:
DIM Disp\$[50]
OUTPUT 707;":MMEM?"
ENTER 707;Disp\$
PRINT Disp\$

FNUMBER**command/query**

The **:MMEMORY:FNUMBER** command selects the specific multiple memory failure to be viewed or transferred. Source of the failure is specified using the **:MMEMORY:SOURCE** command.

The **FNUMBER** query returns the currently selected failure for the multiple memory commands. A returned "0" indicates that there is not any failure data available for the source selected.

Command Syntax: `:MMEMory:FNUMBER <failure>`

Where:

`<failure> ::= 1 to 665`

Example: `OUTPUT 707;":MMEM:SOUR CHAN1"`
`OUTPUT 707;":MMEM:FNUM 35"`

Query Syntax: `:MMEMory:FNUMBER?`

Returned Format: `[:MMEMory:FNUMBER] <failure><NL>`

Where:

`<failure> ::= 0 to 665 (integer - NR1 format)`

Example: `OUTPUT 707;":MMEM:FNUM?"`
`ENTER 707;Fail`
`PRINT Fail`

SOURce

SOURce

command/query

The :MMEMORY:SOURCE command selects the channel or function to be used for the source of the multiple memory commands. Specific failure number is selected using the :MMEMORY:FNUMBER command.

The SOURCE query returns the currently selected source for the multiple memory commands.

Command Syntax: :MMEMory:SOURCE {CHANne1{1|2|3|4} | {FUNct ion{1|2|3|4}}

Example: OUTPUT 707;":MMEM:SOUR CHAN1"

Query Syntax: :MMEMory:SOURCE?

Returned Format: [:MMEMory:SOURCE] {CHANne1{1|2|3|4} | {FUNct ion{1|2|3|4}<NL>

Example:
DIM Src\$ [30]
OUTPUT 707;":MMEM:SOUR?"
ENTER 707;Src\$
PRINT Src\$

Note

CHANnel3 and CHANnel4 are only available when the oscilloscope is an HP 54506B/54512B.

STORE**Command**

The :MMEMORY:STORE command moves the selected multiple memory record (source and failure number) to a waveform memory. Specific multiple memory source and failure number is selected using the :MMEMORY:SOURCE and MMEMORY:FNUMBER commands.

Command Syntax: :MMEMory:STORE

Example: OUTPUT 707;":MMEM:SOUR CHAN1"
OUTPUT 707;":MMEM:FNUM 35"
OUTPUT 707;":MMEM:STOR WMEM1"

Sequential Subsystem

Introduction

The Sequential subsystem commands control how data is acquired, captured, and defined during sequential single-shot mode operation on the HP 54505B, HP 54506B, HP 54510B, and HP 54512B.

Sequential single-shot is available only when `:TIMEBASE:SAMPLE` is set to `REALtime`. When sequential single-shot mode is selected (`:SEQUENTIAL[:DISPLAY] ON`), all data acquisition is stopped. The acquisition is defined using the `:SEQUENTIAL:NPOINTS` and `:SEQUENTIAL:NSEGMENTS` commands. Data is acquired when the `DIGITIZE` root command is executed.

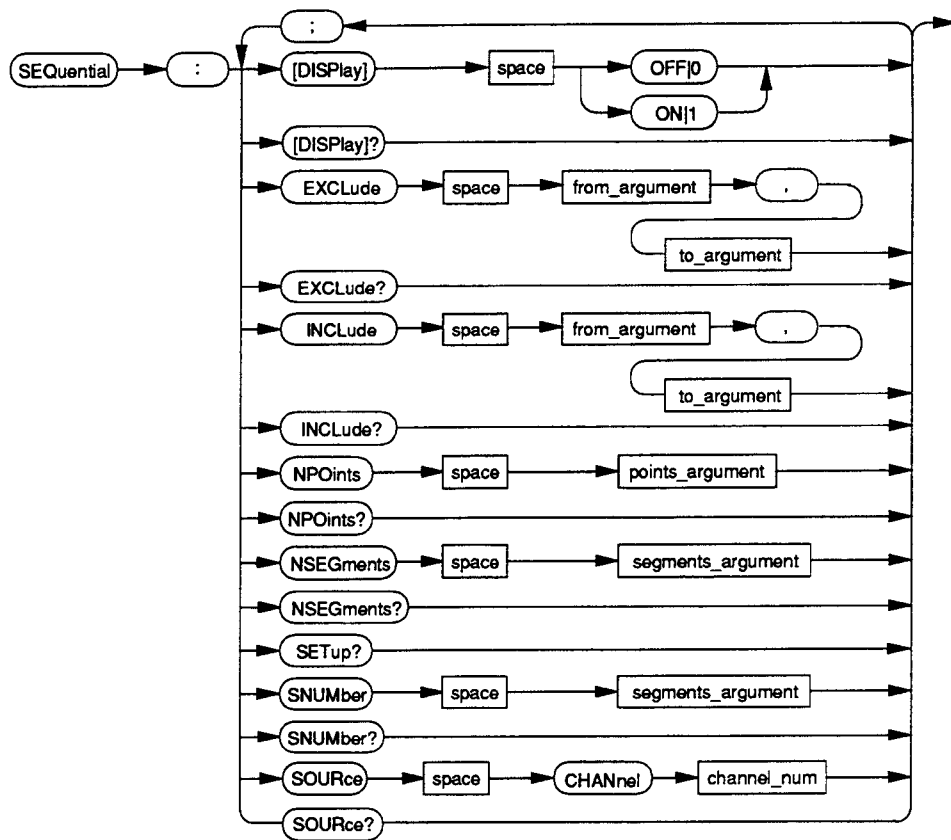
The `:ACQUIRE:TYPE` currently selected determines how the captured data is processed. See Chapter 9 "Acquire Subsystem" for more information on acquisition type available. Individual segment selection is possible using the `:SEQUENTIAL:INCLude`, `:SEQUENTIAL:EXCLude`, and `:SEQUENTIAL:SNUMber` commands.

Captured and processed segments can be displayed on the screen (`DISPLAY` Subsystem), measured (`MEASURE` Subsystem), and/or have waveform math functions (`FUNCTION` Subsystem) performed using the appropriate subsystem commands.

The Sequential subsystem contains the following commands:

- `DISPlay`
- `EXCLude`
- `INCLude`
- `NPOints`
- `NSEGments`
- `SETup`
- `SNUMber`
- `SOURce`

Figure 22-1 lists the syntax diagrams for the Sequential subsystem commands.



- points_argument =** an integer, from 4 to 8000 defining the points being acquired.
- segments_argument =** an integer, from 1 to 8888 dependent on the points_argument.
- channel_number =** an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B).
- from_argument =** an integer, dependent on the segments_argument.
- to_argument =** an integer, dependent on the segments_argument.

Figure 22-1. Sequential Subsystem Commands Syntax Diagram

[DISPlay]**command/query**

The `:SEQUENTIAL[:DISPLAY]` command controls sequential single-shot mode operation. On places the oscilloscope in sequential single-shot mode, and stops all data acquisition.

After data has been acquired (using the `DIGITIZE` root command), captured segments are displayed on the screen. The type of data displayed is selected using the `:ACQUIRE:TYPE` command. When `:ACQUIRE:TYPE` is `NORMAL`, a specific segment can be selected using the `:SEQUENTIAL:SNUMBER` command.

The `DISPLAY` query returns the current setting of the command.

Note 

Because `[:DISPlay]` is an implied command, only the parameters need be sent to turn the sequential single-shot mode display on or off. For example, to turn on the display, send either `":SEQ:DISP ON"` or `":SEQ ON"`.

Command Syntax: `:SEQuentia1[:DISPlay] {{ON | 1} | {OFF | 0}}`

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

Query Syntax: `:SEQuentia1[:DISPlay]?`

Returned Format: `[:SEQuentia1[:DISPlay]] {1 | 0}<NL>`

Example:
`DIM Disp$[50]`
`OUTPUT 707;":SEQ?"`
`ENTER 707;Disp$`
`PRINT Disp$`

EXCLude

EXCLude

command/query

The `:SEQUENTIAL:EXCLUDE` command selects a range of previously captured segments that **WILL NOT BE USED** for future sequential single-shot operations.

When entering the range of segment numbers:

- All segment numbers specified must have been previously acquired (as defined by the `SEQUENTIAL:NSEGMENTS` command).
- The "from segment" specified cannot exceed the "to segment" specified.

The `EXCLUDE` query returns a list of currently selected excluded segments for the sequential single-shot commands.

Command Syntax: `:SEQuential:EXCLude <from_argument>,<to_argument>`

Where:

`<from_argument>` ::= any previously captured segment number less than or equal to the `<to_argument>` specified.
`<to_argument>` ::= any previously captured segment number greater than or equal to the `<from_argument>` specified.

Example:

```
OUTPUT 707;":TIMEBASE:SAMPLE REALTIME"  
OUTPUT 707;":SEQ ON  
OUTPUT 707;":ACQUIRE:TYPE AVER"  
OUTPUT 707;":SEQUENTIAL:NPOINTS 500"  
OUTPUT 707;":SEQUENTIAL:NSEGMENTS 10"  
OUTPUT 707;":SEQUENTIAL:SOURCE CHAN1"  
OUTPUT 707;":DIGITIZE CHAN1"  
OUTPUT 707;"::SEQUENTIAL:EXCLUDE 5,10"
```

EXCLude

Query Syntax: :SEquential:EXCLude?

Returned Format: [:SEquential:EXCLude] <exclude_list><NL>

Where:

<exclude_list> ::= a list of previously captured segment numbers separated by commas (integer - NR1 format).

Example: DIM Exc\$[50]
OUTPUT 707;":SEQ:EXCL?"
ENTER 707;Exc\$
PRINT Exc\$

INCLude

INCLude

command/query

The `:SEQUENTIAL:INCLUDE` command selects a range of previously captured and excluded segments that **WILL BE USED** for future sequential single-shot operations.



Note

Because all segments are initially acquired as `INCLUDE`, this command should only be used to include segments that were previously excluded using the `:SEQUENTIAL:EXCLUDE` command.

When entering the range of segment numbers:

- All segment numbers specified must have been previously acquired (as defined by the `SEQUENTIAL:NSEGMENTS` command).
- The "from segment" specified cannot exceed the "to segment" specified.

The `INCLUDE` query returns a list of currently selected included segments for the sequential single-shot commands.

Command Syntax: `:SEQuential:INCLude <from_argument>,<to_argument>`

Where:

`<from_argument>` ::= any previously captured segment number less than or equal to the `<to_argument>` specified.

`<to_argument>` ::= any previously captured segment number greater than or equal to the `<from_argument>` specified.

Example: `OUTPUT 707;"::SEQUENTIAL:INCLUDE 5,10"`

INCLude

Query Syntax: :SEquential:INCLude?

Returned Format: [:SEquential:INCLude] <include_list><NL>

Where:

<include_list> ::= a list of previously captured segment numbers separated by commas (integer - NR1 format).

Example: DIM Inc\$[50]
OUTPUT 707;":SEQ:INCL?"
ENTER 707;Inc\$
PRINT Inc\$

NPoints

NPoints

command/query

The `:SEQUENTIAL:NPOINTS` command specifies the number of time buckets for each acquired segment. The number of points specified also effects the total number of `SEQUENTIAL:NSEGMENTS` available.

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

Note



501 points are acquired for a normal screen display.

Command Syntax: `:SEQuential:NPOints <points_argument>`

Where:

`<points_argument> ::= 4 to 8000.`

Example: `OUTPUT 707;":SEQUENTIAL:NPOINTS 500"`

Query Syntax: `:SEQuential:NPOints?`

Returned Format: `[[:SEQuential:NPOints] <points_argument><NL>`

Where:

`<points_argument> ::= 4 to 8000 (integer - NR1 format).`

Example:
`DIM Pts$[50]
OUTPUT 707;":SEQ:NPO?"
ENTER 707;Pts$
PRINT Pts$`

NSEGments**command/query**

The `:SEQUENTIAL:NSEGMENTS` command specifies the number of segments that will be acquired for the sequential single-shot mode. The number of time buckets acquired for each segment is specified using the `:SEQUENTIAL:NPOINTS` command.

The specified number of segments are acquired when the `RUN` command is executed. After data acquisition, the oscilloscope is placed in the stopped state.

The `NSEGMENTS` query returns the number of segments to be acquired.

Note

The range of segment numbers is from 1 to 8888; however, the maximum number of segments allowable is dependent on the current number of points specified using the `SEQUENTIAL:NPOINTS` command, and the active channel.

Command Syntax: `:SEQuential:NSEGments <segments_argument>`

Where:

`<segments_argument> ::= 1 to 8888` dependent on the `SEQUENTIAL:NPOINTS` selection.

Example: `OUTPUT 707;":SEQUENTIAL:NPOINTS 8000"`
`OUTPUT 707;":SEQUENTIAL:NSEGMENTS 49"`

NSEGments

Query Syntax: :SEquential:NSEGments?

Returned Format: [:SEquential:NSEGments <segments_argument><NL>

Where:

<segments_argument>::= 1 to 8888 dependent on the SEQUENTIAL:NPOINTS selection (integer - NR1 format).

Example: DIM Seg\$[50]
OUTPUT 707;":SEQ:NSEG?"
ENTER 707;Seg\$
PRINT Seg\$

SETup

query

The :SEQUENTIAL:SETUP query returns the current settings for the SEQUential Subsystem commands.

Query Syntax: :SEQuential:SETup?

Returned Format: :SEQuential:DISPlay {ON|OFF};EXCLude <exclude_list>;INCLude <include_list>;NPOints <points>;NSEGments <segment>;SNUMber <number>;SOURce {CHANnel1|2|3|4}<NL>

Where:

<points> ::= 4 to 8000 (integer - NR1 format)
 <segments> ::= 1 to 8888 dependent on the points selected (integer - NR1 format)
 <number> ::= 1 to 8888 dependent on the segment selected (integer - NR1 format)
 <include_list> ::= a list of previously captured segment numbers separated by commas (integer - NR1 format)
 <exclude_list> ::= a list of previously captured segment numbers separated by commas (integer - NR1 format)

Example:
 DIM Stp\$[300]
 OUTPUT 707;":SEQ:SET?"
 ENTER 707;Stp\$
 PRINT Stp\$

Note

CHANnel3 and CHANnel4 are returned only when the oscilloscope is an HP 54506B/54512B.

SNUMBER

SNUMBER

command/query

The `:SEQUENTIAL:SNUMBER` command selects one previously defined and captured segment. `ACQUIRE:TYPE NORMAL` must be specified to select an individual segment.

The `SNUMBER` query returns a segment number dependent on the `ACQUIRE:TYPE` currently selected.

- `ACQUIRE:TYPE NORMAL`: returns the segment number selected using the `:SEQUENTIAL:SNUMBER` command.
- `ACQUIRE:TYPE AVERAGE` or `ENVELOPE`: returns the total number of segments used to generate the data.

Command Syntax: `:SEQuential:SNUMBER <segment_number>`

Where:

`<segment_number>` ::= 1 to 8888 dependent on the number of segments acquired.

Example: `OUTPUT 707;":SEQUENTIAL:SNUMBER 5"`

Query Syntax: `:SEQuential:SNUMBER?`

Returned Format: `[:SEQuential:SNUMBER] <segment_number><NL>`

Where:

`<segment_number>` ::= 1 to 8888 dependent on the number of segments acquired and `ACQUIRE:TYPE` currently selected (integer - NR1 format).

Example:
`DIM Seg$[10]`
`OUTPUT 707;":SEQ:SNUM?"`
`ENTER 707;Seg$`
`PRINT Seg$`

SOURCE**command/query**

The :SEQUENTIAL:SOURCE command selects the channel to be used for the source of the sequential single-shot commands. Data can only be acquired from an active channel. Attempting to acquire data from a channel turned off will cause an error to be generated.

The SOURCE query returns the currently selected source for the sequential single-shot commands.

Command Syntax: :SEQuentia1:SOURce {CHANne1{1|2|3|4}}

Example: OUTPUT 707;":SEQ:SOUR CHAN1"

Query Syntax: :SEQuentia1:SOURce?

Returned Format: [:SEQuentia1:SOURce] {CHANne1{1|2|3|4}<NL>

Example: DIM Src\$[30]
OUTPUT 707;":SEQ:SOUR?"
ENTER 707;Src\$
PRINT Src\$

Note

CHANnel3 and CHANnel4 are available only when the oscilloscope is an HP 54506B/54512B.

Marker Subsystem

Introduction

The Marker subsystem commands control the two x-axis (vertical) and y-axis (horizontal) markers. Each set of markers are independently programmable for source and position.

The Marker subsystem contains the following commands:

- DISPlay
- SETup?
- XDELta?
- X1Position
- X1Position?
- X2Position
- X2Position?
- X1Y1source
- X2Y2source
- YDELta?
- Y1Position
- Y2Position

Figure 23-1 lists the syntax diagrams for the Sequential subsystem commands.

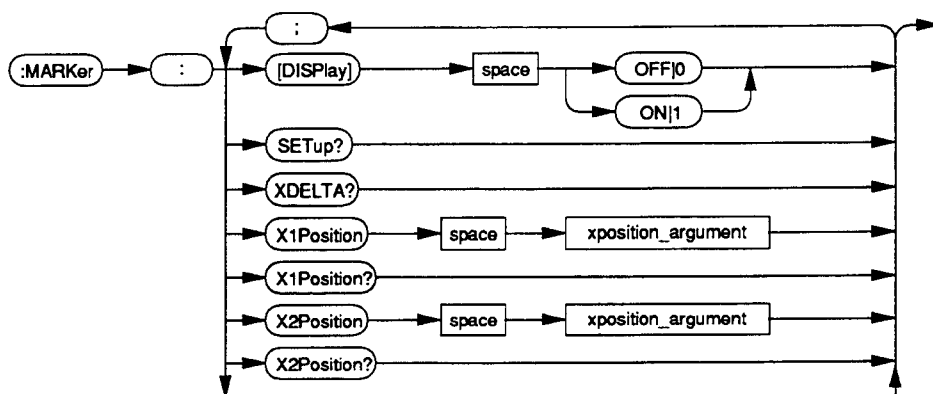
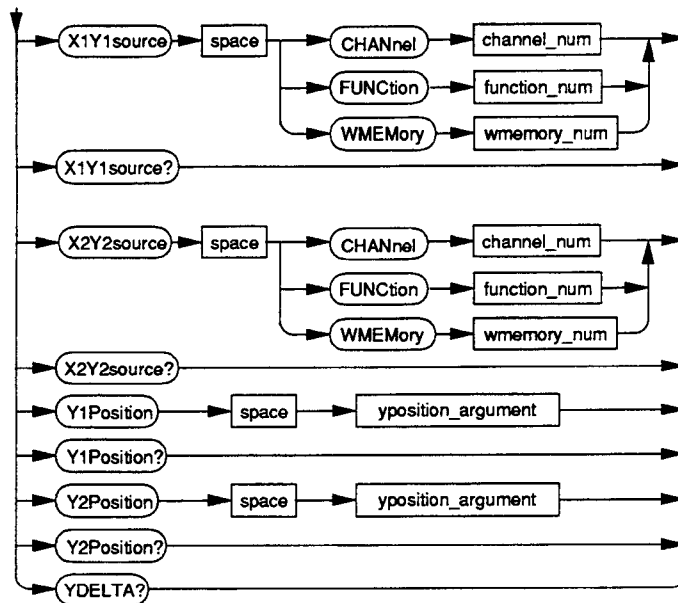


Figure 23-1. Marker Subsystem Commands Syntax Diagram



- xposition_argument =** time in seconds from the trigger
- yposition_argument =** a real number within the voltage range.
- channel_number =** an integer, 1 or 2 (HP 54505B/54510B) or 1 through 4 (HP 54506B/54512B).
- function_number =** an integer, 1 through 4.
- wmemory_number =** an integer, 1 through 4.

Figure 23-1. Marker Subsystem Commands Syntax Diagram (continued)

[DISPlay]**command/query**

The `:MARKER[:DISPLAY]` command controls the marker displayed on the screen. On displays x/y markers, menus, and current position and statistic data on the screen.

The `DISPLAY` query returns the current setting of the command.

Note 

Because `[:DISPlay]` is an implied command, only the parameters need be sent to turn the marker display on or off. For example, to turn on the display, send either `":MARK:DISP ON"` or `":MARK ON"`.

Command Syntax: `:MARKer[:DISPlay] {{ON | 1} | {OFF | 0}}`

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

Query Syntax: `:MARKer[:DISPlay]?`

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

Example:
`DIM Disp$[50]
OUTPUT 707;":MARK?"
ENTER 707;Disp$
PRINT Disp$`

SETup

SETup

query

The :MARKER:SETUP query returns the current settings for the MARKer Subsystem commands.

Query Syntax: :MARKer:SETup?

Returned Format: :MARKer:DISP lay {0N|OFF};XDELta <xdelta>;X1Position <xposition>;X2Position <xposition>;X1Y1source {CHANnel1|2|3|4} | {FUNction1|2|3|4} | {WMEMory1|2|3|4};X2Y2source {CHANnel1|2|3|4} | {FUNction1|2|3|4} | {WMEMory1|2|3|4};YDELta <ydelta>;Y1Position <yposition>;Y2Position <yposition> <NL>

Where:

<xdelta>::= difference between x1 and x2 markers (exponential - NR3 format)
<xposition>::= xmarker in seconds or hertz (exponential - NR3 format)
<ydelta>::= difference between y1 and y2 markers (exponential - NR3 format)
<yposition>::= ymarker in volts or power (exponential - NR3 format)

Example: DIM Stp\$[300]
OUTPUT 707;":MARK:SET?"
ENTER 707;Stp\$
PRINT Stp\$

Note



CHANnel3 and CHANnel4 are returned only when the oscilloscope is an HP 54506B/54512B.

XDELta

query

The :MARKER:XDELTA query returns the time or frequency difference between the x1 and x2 markers:

$$x\delta = x2\text{marker} - x1\text{marker}$$

Time is returned in seconds for all modes except FFT, where frequency is returned in hertz. If the returned value is negative, x1marker is located at a point previous to the x2marker.

Note 

This command is identical to the MEASURE:TDELTA command, and is provided for compatibility with other oscilloscopes.

Query Syntax: :MARKer:XDELta?

Returned Format: [:MARKer:XDELta] <xvalue><NL>

Where:

<xvalue> ::= difference between x1 and x2 markers (exponential - NR3 format)

Example:

```
DIM Xd1$[50]
OUTPUT 707;" :MARKER:XDELTA?"
ENTER 707;Xd1$
PRINT Xd1$
```

X1Position

X1Position

command/query

The `:MARKER:X1POSITION` command moves the x1marker to the specified time with respect to the trigger time. Positive values set the marker to to right of the trigger point (post-trigger), while negative values set the marker to the left of the trigger point (pre-trigger). Trigger point is set using the `TIMEBASE:REFERENCE` command.

The `X1POSITION` query returns the position of the x1marker. Time is returned in seconds for all modes except FFT, where frequency is returned in hertz.



Note

This command is identical to the `MEASURE:TSTART` command, and is provided for compatibility with other oscilloscopes.

Command Syntax: `:MARKer:X1Position <xposition>`

Where:

`<xposition>::= xmarker time in seconds (all modes except FFT)`
`<xposition>::=frequency in hertz (FFT mode)`

Example: `OUTPUT 707;":MARKER:X1POSITION 30 NS"`

Query Syntax: `:MARKer:X1Position?`

Returned Format: `[:MARKer:X1Position] <xposition><NL>`

Where:

`<xposition>::= xmarker time in seconds, or frequency in hertz (exponential - NR3 format)`

Example:
`DIM X1p$`
`OUTPUT 707;":MARKER:X1POSITION?"`
`ENTER 707;X1p$`
`PRINT X1p$`

X2Position**command/query**

The `:MARKER:X2POSITION` command moves the x2marker to the specified time with respect to the trigger time. Positive values set the marker to to right of the trigger point (post-trigger), while negative values set the marker to the left of the trigger point (pre-trigger). Trigger point is set using the `TIMEBASE:REFERENCE` command.

The `X2POSITION` query returns the position of the x1marker. Time is returned in seconds for all modes except FFT, where frequency is returned in hertz.



This command is identical to the `MEASURE:TSTOP` command, and is provided for compatibility with other oscilloscopes.

Command Syntax: `:MARKer:X2Position <xposition>`

Where:

`<xposition>::= xmarker time in seconds (all modes except FFT)`
`<xposition>::=frequency in hertz (FFT mode)`

Example: `OUTPUT 707;":MARKER:X2POSITION 40 NS"`

Query Syntax: `:MARKer:X2Position?`

Returned Format: `[:MARKer:X2Position] <xposition><NL>`

Where:

`<xposition>::= xmarker time in seconds, or frequency in hertz (exponential - NR3 format)`

Example:
`DIM X2p$`
`OUTPUT 707;":MARKER:X2POSITION?"`
`ENTER 707;X2p$`
`PRINT X2p$`

X1Y1source

X1Y1source

command/query

The `:MARKER:X1Y1SOURCE` command selects the channel, function, or waveform memory to be used for the source of the X1 and Y1 marker commands.

The `X1Y1SOURCE` query returns the currently selected source for the X1 and Y1 marker commands.

Command Syntax: `:MARKer:X1Y1source {CHANne1{1|2|3|4} | FUNCTion{1|2|3|4} | WMEMory{1|2|3|4}}`

Example: `OUTPUT 707;":MARK:X1Y1 CHAN1"`

Query Syntax: `:MARKer:X1Y1source?`

Returned Format: `[:MARKer:X1Y1source] {CHANne1{1|2|3|4} | FUNCTion{1|2|3|4} | WMEMory{1|2|3|4}<NL>`

Example:
`DIM Src${30}`
`OUTPUT 707;":MARK:X1Y1?"`
`ENTER 707;Src$`
`PRINT Src$`

Note



CHANne13 and CHANne14 are available only when the oscilloscope is an HP 54506B/54512B.

X2Y2source**command/query**

The **:MARKER:X2Y2SOURCE** command selects the channel, function, or waveform memory to be used for the source of the X2 and Y2 marker commands.

The **X2Y2SOURCE** query returns the currently selected source for the X2 and Y2 marker commands.

Command Syntax: `:MARKer:X2Y2source {CHANne1{1|2|3|4} | FUNCTion{1|2|3|4} | WMEMory{1|2|3|4}`

Example: `OUTPUT 707;":MARK:X2Y2 CHAN1"`

Query Syntax: `:MARKer:X2Y2source?`

Returned Format: `[[:MARKer:X2Y2source] {CHANne1{1|2|3|4} | FUNCTion{1|2|3|4} | WMEMory{1|2|3|4}<NL>`

Example:
`DIM Src$[30]
OUTPUT 707;":MARK:X2Y2?"
ENTER 707;Src$
PRINT Src$`

Note

CHANnel3 and CHANnel4 are available only when the oscilloscope is an HP 54506B/54512B.

YDELta

YDELta

query

The :MARKER:YDELTA query returns the voltage or power difference between the y1 and y2 markers:

$$y\delta = y2\text{marker} - y1\text{marker}$$

Level is returned in volts for all modes except FFT, where power is returned in dBm. If the returned value is negative, the y1marker is located at a lower level than the y2marker.

Note

This command is identical to the MEASURE:VDELTA command, and is provided for compatibility with other oscilloscopes.

Query Syntax: :MARKer:YDELta?

Returned Format: [:MARKer:YDELta] <yvalue><NL>

Where:

<yvalue> ::= difference between y1 and y2 markers (exponential - NR3 format)

Example:
DIM Yd1\$[50]
OUTPUT 707;":MARKER:YDELTA?"
ENTER 707;Yd1\$
PRINT Yd1\$

Y1Position

command/query

The :MARKER:Y1POSITION command moves the y1marker to the specified level. The acceptable values are limited to the currently defined channel, function, or waveform memory range.

The Y1POSITION query returns the position of the y1marker. Level is returned in volts for all modes except FFT, where power is returned in dBm.

Note

This command is identical to the MEASURE:VSTART command, and is provided for compatibility with other oscilloscopes.

Command Syntax: :MARKer:y1Position <yposition>

Where:

<yposition>::= ymarker voltage (all modes except FFT)
<yposition>::=power in dBm (FFT mode)

Example: OUTPUT 707;":MARKER:y1POSITION -10MV"

Query Syntax: :MARKer:y1Position?

Returned Format: [:MARKer:y1Position] <yposition><NL>

Where:

<yposition>::= ymarker level in volts, or power in dBm (exponential - NR3 format)

Example: DIM Y1p\$[50]
OUTPUT 707;":MARKER:Y1POSITION?"
ENTER 707;X1p\$
PRINT Y1p\$

Y2Position

Y2Position

command/query

The `:MARKER:Y2POSITION` command moves the y2marker to the specified level. The acceptable values are limited to the currently defined channel, function, or waveform memory range.

The `Y2POSITION` query returns the position of the y2marker. Level is returned in volts for all modes except FFT, where power is returned in dBm.



Note

This command is identical to the `MEASURE:VSTOP` command, and is provided for compatibility with other oscilloscopes.

Command Syntax: `:MARKer:y2Position <yposition>`

Where:

`<yposition>::= ymarker voltage (all modes except FFT)`
`<yposition>::=power in dBm (FFT mode)`

Example: `OUTPUT 707;":MARKER:y2POSITION -10MV"`

Query Syntax: `:MARKer:y2Position?`

Returned Format: `[:MARKer:y2Position] <yposition><NL>`

Where:

`<yposition>::= ymarker level in volts, or power in dBm (exponential - NR3 format)`

Example:
`DIM Y2p$[50]`
`OUTPUT 707;":MARKER:Y2POSITION?"`
`ENTER 707;Y2p$`
`PRINT Y2p$`

Status Reporting

Introduction

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

The bits in the status byte act as summary bits for the data structures residing behind them. In the case of queues, the summary bit is set if the queue is not empty. For registers, the summary bit is set if any enabled bit in the event register is set. The events are enabled with the corresponding event enable register. Events captured by an event register remain set until the register is read or cleared. Registers are read with their associated commands. The *CLS command clears all event registers and all queues except the output queue. If *CLS is sent immediately following a program message terminator, the output queue is also cleared.

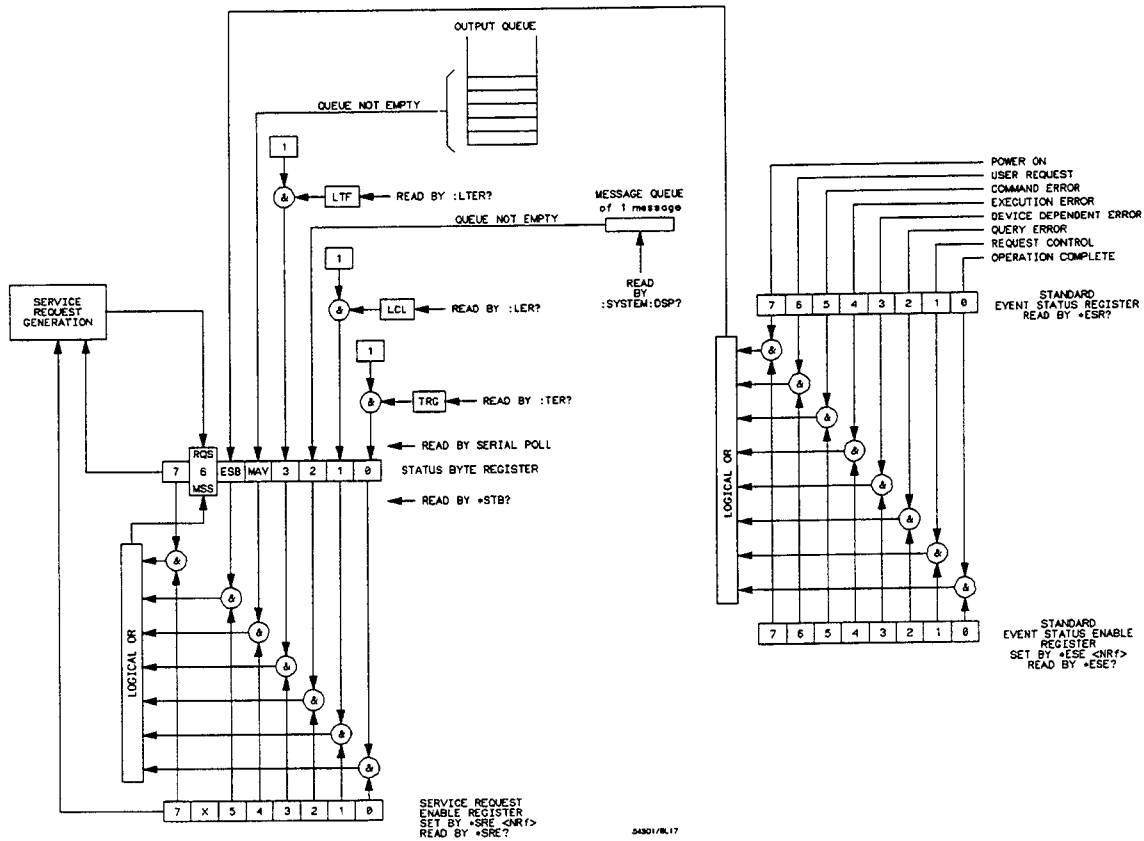


Figure 24-1. Status Reporting Data Structures

Bit Definitions **MAV - message available.** Indicates whether there is a response in the output queue.

ESB - event status bit. Indicates if any of the conditions in the Standard Event Status Register are set and enabled.

MSS - master summary status. Indicates whether the device has a reason for requesting service. This bit is returned for the *STB? query.

RQS - request service. Indicates if the device is requesting service. This bit is returned during a serial poll. RQS is set to 0 after it is read via a serial poll (MSS is not reset by *STB?).

MSG - Message. Indicates whether there is a message in the message queue.

PON - power on. Always 0 in the HP 5451XB.

URQ - user request. Indicates whether a front-panel key has been pressed.

CME - command error. Indicates whether the parser detected an error.

EXE - execution error. Indicates whether a parameter was out of range, or inconsistent with the current settings.

DDE - device specific error. Indicates whether the device was unable to complete an operation for device dependent reasons.

QYE - query error. Indicates whether the protocol for queries has been violated.

RQC - request control. Indicates whether the device is requesting control. The HP 54510A never requests control.

OPC - operation complete. Indicates whether the device has completed all pending operations.

LCL - local. Indicates whether a remote-to-local transition has occurred.

TRG - trigger. Indicates whether a trigger has been received.

LTF - limit test failure. Indicates whether a limit test or compare test failure has occurred.

Operation Complete (*OPC)

The IEEE 488.2 structure provides one technique which can be used to find out if any operation is finished. The *OPC command, when sent to the instrument after the operation of interest, sets the OPC bit in the Standard Event Status Register when all pending device operations have finished. If the OPC bit and the RQS bit have been enabled, a service request is generated.

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

Trigger Bit (TRG)

The Trigger (TRG) bit indicates if the device has received a trigger. The TRG event register will stay set after receiving a trigger until it is cleared by reading it or using the *CLS command. If your application needs to detect multiple triggers, the TRG event register must be cleared after each one.

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

```
OUTPUT 707;"*SRE 1"           ! enables a trigger service request.
                                ! the next trigger will generate an SRQ.

OUTPUT 707;" :TER?"           ! queries the TRG event register, thus
ENTER 707;A$                   ! clearing it.
                                ! the next trigger can now generate an
                                ! SRQ
```

Status Byte

If the device is requesting service (RQS set), and the controller serial polls the device, the RQS bit is cleared. The MSS bit (read with *STB?) is not cleared by reading it. The status byte is not cleared when read, except for the RQS bit.

Serial Poll

The HP 54505B, HP 54506, HP 54510B, and HP 54512B each support the IEEE 488.1 serial poll feature. When a serial poll of the instrument is requested, the RQS bit is returned on bit 6 of the status byte.

Using Serial Poll

This example shows how to use the service request by conducting a serial poll of all instruments on the bus. In this example, assume that there are two instruments on the bus: an oscilloscope at address 7 and a printer at address 1. These address assumptions are made throughout this manual. It is also assumed that you are operating on Interface Select Code 7.

The program command for serial poll using HP BASIC 5.0 is `Stat = SPOLL(707)`. The address 707 is the address of the oscilloscope in this example. The command for checking the printer is `Stat = SPOLL(701)` because the address of that instrument is 01 on bus address 7. This command reads the contents of the 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 instrument's status register is high, use the following command line.

IF BIT (Stat, 6) then

5. If bit 6 of the instrument at address 1 is not high, then check the instrument at address 7 to see if bit 6 of its status register is high.
6. As soon as the instrument with status bit 6 high is found, check the rest of the status bits to determine what is required.

The SPOLL(707) command causes much more to happen on the bus than simply reading the register. This command clears the bus, automatically addresses the talker and listener, sends SPE (serial poll enable) and SPD (serial poll disable) bus commands, and reads the data. For more information about serial poll, refer to your controller manual and programming language reference manuals.

After the serial poll is completed, the RQS bit in the oscilloscope's Status Byte Register is reset if it was set. Once a bit in the Status Byte Register is set, it remains set until the status is cleared with a *CLS command, or the instrument is reset. If these bits do not get reset, they cannot generate another SRQ.

Error Messages

This chapter lists the error messages that are returned by the parser on the HP 54505B, HP 54506B, HP 54510B, and HP 54512B digitizing oscilloscope.

| Error Number | Description |
|---------------------|---------------------------------|
| 11 | Questionable horizontal scaling |
| 12 | Edges required not found |
| 70 | RAM write protected |
| -100 | Command error (unknown command) |
| -101 | Invalid character |
| -102 | Syntax error |
| -103 | Invalid separator |
| -104 | Data type error |
| -105 | GET not allowed |
| -108 | Parameter not allowed |
| -109 | Missing parameter |
| -112 | Program mnemonic too long |
| -113 | Undefined header |
| -121 | Invalid character in number |
| -123 | Numeric overflow |
| -124 | Too many digits |
| -128 | Numeric data not allowed |
| -130 | Suffix error |
| -131 | Invalid suffix |
| -138 | Suffix not allowed |
| -140 | Character data error |
| -141 | Invalid character data |
| -144 | Character data too long |
| -148 | Character data not allowed |

| Error Number | Description |
|---------------------|--|
| -150 | String data error |
| -151 | Invalid string data |
| -158 | String data not allowed |
| -160 | Block data error |
| -161 | Invalid block data |
| -168 | Block data not allowed |
| -170 | Expression error |
| -171 | Invalid expression |
| -178 | Expression data not allowed |
| -200 | Execution error |
| -211 | Trigger ignored |
| -221 | Settings conflict |
| -222 | Data out of range |
| -223 | Too much data |
| -310 | System error |
| -350 | Too many errors |
| -400 | Query error |
| -410 | Query INTERRUPTED |
| -420 | Query UNTERMINATED |
| -430 | Query DEADLOCKED |
| -440 | Query UNTERMINATED after indefinite response |

Algorithms

Introduction

One of the primary features of the oscilloscope is its ability to make automatic measurements on displayed waveforms. This chapter provides details on how automatic measurements are calculated and offers some tips on how to improve results.

Measurement Setup

Measurements typically should be made at the fastest possible sweep speed for the most accurate measurement results. The entire portion of the waveform that is to be measured must be displayed on the oscilloscope:

- At least one complete cycle must be displayed for period or frequency measurements.
 - The entire pulse must be displayed for width measurements.
 - The leading edge of the waveform must be displayed for rise time measurements and all other edge measurements.
 - The trailing edge of the waveform must be displayed for fall time measurements and all other edge measurements.
-

Making Measurements

If more than one waveform, edge, or pulse is displayed, the measurements are made on the first (leftmost) portion of the displayed waveform that can be used. If there are not enough data points, the oscilloscope displays \leq with the measurement results. This is to remind you that the results may not be as accurate as possible. It is recommended that you re-scale the displayed waveform and make your measurement again.

When any of the standard measurements are requested, the oscilloscope first determines the top-base voltage levels at 100%-0%. From this information, it can determine the other important voltage values (10%, 90%, and 50%) needed to make the measurements. The 10% and 90% voltage values are used in the rise time and fall time measurements, as well as in all other edge measurements. The 10% and 90% values are also used to determine the 50% value. The 50% voltage value is used for measuring frequency, period, pulse width, and duty cycle.

Automatic Top-Base

Top-Base is the heart of most automatic measurements. It is used to determine V_{top} and V_{base} , the 0% and 100% voltage levels at the top and bottom of the waveform. From this information the oscilloscope can determine the 10%, 50%, and 90% points, which are also used in most measurements. The top or base of the waveform is not necessarily the maximum or minimum voltage present on the waveform. Consider a pulse that has slight overshoot. It would be wrong to select the highest point of the waveform as the top since the waveform normally rests below the perturbation.

Top-Base performs a histogram on the waveform and finds the most prevalent point above and below the waveform midpoint. The most prevalent point is one that represents greater than approximately 5% of the total display points (501) and is considered to be either the top or base. If no point accounts for more than 5% of the total, then the top is chosen as the absolute maximum and the base is chosen as the absolute minimum.

Edge Definition

Both rising and falling edges are defined as transitional edges that must cross three thresholds.

A rising edge must cross the lower threshold in a positive direction (defining it as a rising edge), cross the mid threshold (any number of crossings, both positive and negative are permissible) and then cross the upper threshold without any additional crossing of the lower threshold.

A falling edge must cross the upper threshold in a negative direction, cross the mid threshold (any number of times), and then cross the lower threshold without any additional crossing of the upper threshold.



Most time measurements are made based on the position of the first crossing of the middle threshold.

Algorithm Definitions

This section lists the definitions that the measurements in the HP 54505B, HP 54506B, HP 54510B, and HP 54512B are based on.

delay There are three types of delay measurements:

- Jitter.
- Standard.
- User-defined.

Jitter occurs only under the following circumstances:

- The standard/user-defined key is set to standard.
- Two delay parameters are the same.
- The display mode is set to envelope.

if

first edge on minimum waveform is rising

then

**delay = mid-threshold of first rising edge of max waveform minus
mid-threshold of first rising edge on min waveform**

else

**delay = mid-threshold of first falling edge on min waveform minus
mid-threshold of first falling edge on max waveform**

The standard delay measurement occurs in the standard mode (not user-defined) and is not a jitter measurement.

standard delay = mid-threshold of the first edge of the second parameter minus the mid-threshold of the first edge of the first parameter

Note 

Negative delay is possible.

User defined delay = second channel edge minus first channel edge

+ width The + width algorithm has standard and user-defined considerations.

if

first edge is rising

then

+ width = mid-threshold crossing of first falling edge minus mid-threshold crossing of first rising edge

else

+ width = mid-threshold crossing of second falling edge minus mid-threshold crossing of first rising edge

The user-defined definition is the same as the standard definition except for the user-defined threshold.

- width The - width algorithm has standard and user-defined considerations:

if

first edge is rising

then

- width = second rising edge minus first falling edge


else

- width = first rising edge minus first falling edge

Period if
 first edge is rising
 then
 period = second rising edge minus first rising edge
 else
 period = second falling edge minus first falling edge

Frequency frequency = 1/period

Duty Cycle duty cycle = (+ width/period) * 100

Note  The + width for duty cycle is always calculated using the mid-threshold.

Rise time rise time = time at upper threshold minus time at lower threshold

Fall time fall time = time at lower threshold minus time at upper threshold

V_{max} V_{max} = voltage of the maximum point on the screen

V_{min} V_{min} = voltage of the minimum point on the screen

V_{p-p} V_{p-p} = V_{max} minus V_{min}

V_{top} V_{top} = most prevalent point above waveform midpoint

V_{base} V_{base} = most prevalent point below waveform midpoint

V_{amp} V_{amp} = V_{top} minus V_{base}

Vavg The average voltage of the first cycle of the displayed signal is measured. If a complete cycle is not present, the oscilloscope averages all data points.

Vrms (ac) The ac rms voltage of the first cycle of the displayed signal is measured. If a complete cycle is not present, the measurement computes ac rms on all data points.

$$V_{\text{rms(ac)}} = \left[\frac{1}{n} \sum_{j=0}^{n-1} V_j^2 - \left(\frac{1}{n} \sum_{j=0}^{n-1} V_j \right)^2 \right]^{\frac{1}{2}}$$

Vrms (dc) The true rms voltage of the first cycle of the displayed signal is measured.

$$V_{\text{rms(dc)}} = \left[\frac{1}{n} \sum_{j=0}^{n-1} (V_j)^2 \right]^{\frac{1}{2}}$$

Integrate

$$I_n = \sum_{i=0}^{n-1} C_i \Delta t$$

The equation is the integral of the channel, where I represents the integral and C represents the channel. The integral is calculated by adding the voltage points multiplied by the time bucket width, Δt .

Differentiate $d_1 = 0$

$$d_n = \frac{c(n) - c(n-1)}{\Delta t}$$

The equation is the differential waveform of the channel, where d represents the differential and c represents the channel. The differential is the voltage differences between consecutive points in time divided by the time bucket width, Δt .

Message Communication and System Functions

Introduction

This chapter describes the operation of instruments that operate in compliance with the IEEE 488.2 standard. Instruments that are compatible with IEEE 488.2 must also be compatible with IEEE 488.1; however, IEEE 488.1 compatible instruments may or may not conform to the IEEE 488.2 standard. The IEEE 488.2 standard defines the message exchange protocols by which the instrument and the controller 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 instrument is parsing and executing commands. The oscilloscope's input buffer holds 300 characters or bytes of data.

Output Queue

The output queue of the instrument is the memory area where all output data (response messages) are stored until read by the controller. The oscilloscope's output queue holds 300 characters; however, the instrument will handle block data of greater than 300 characters where appropriate.

Parser

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

Protocol Overview

The instrument and controller communicate using program messages and response messages. These messages serve as the containers into which sets of program commands or instrument responses are placed. Program messages are sent by the controller to the instrument, and response messages 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 only talks 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 only talks when prompted to, and it then expects to talk before being told to do something else.

Protocol Operation

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

The instrument and the controller communicate by exchanging complete program messages and response messages. This means that the controller should always terminate a program message before attempting to read a response. The instrument terminates response messages 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." Multiple queries in a query message are separated by semicolons. The responses to each of the queries in a compound query are also separated by semicolons.

Commands are executed in the order they are received. This also applies to 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, the instrument 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 due to an error, the device does not indicate a query error, but simply waits 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 is reported if the instrument detects a syntax error or an unrecognized command header.

Execution Error

An execution error is 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 is reported if the instrument is unable to execute a command for a strictly device dependent reason.

Query Error

A query error is 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 is generated. The parser resets itself, and the response is 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 generates a query error. The unread portion of the response is then discarded by the instrument. The interrupting program message is not 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 breaks 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 is also set.

Syntax Diagrams

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

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

Syntax Overview

This overview is intended to give a quick glance at the syntax defined by IEEE 488.2. It should allow you to understand many of the things about the syntax you need to know. This chapter also contains 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 27-1 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 units are separated by a semicolon.
2. A program message is terminated by an < NL > (new line), an < 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 chapter 4, "Programming and Documentation Conventions").
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.

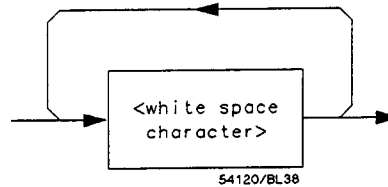


Figure 27-2. White Space

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. For more information, refer to the Tree Traversal Rules in chapter 4, "Programming and Documentation Conventions."

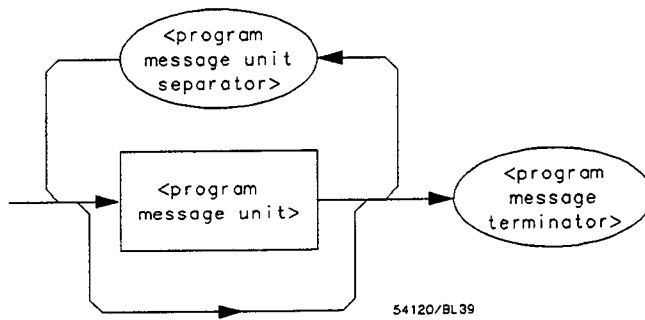


Figure 27-3. Program Message

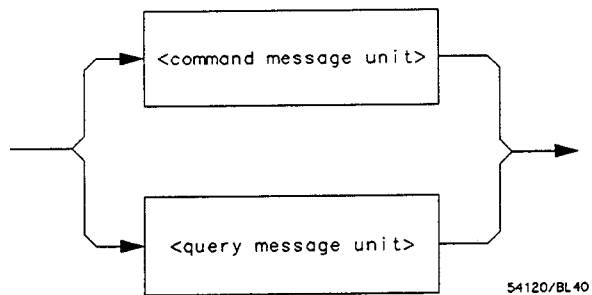


Figure 27-4. Program Message Unit

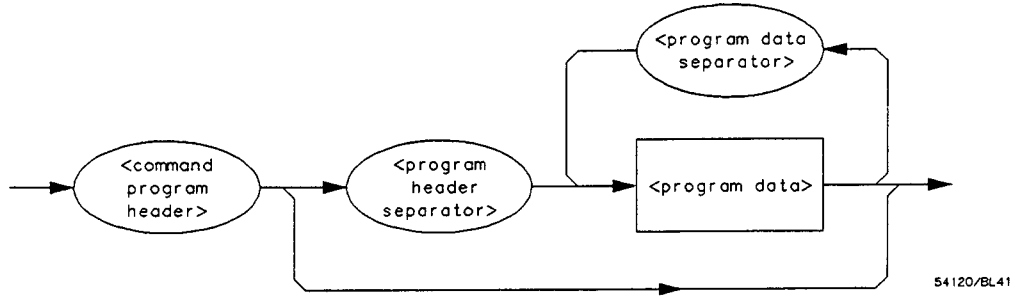


Figure 27-5. Command Message Unit

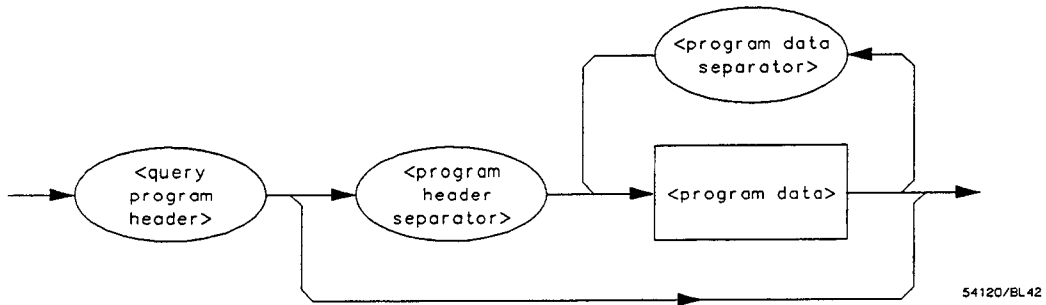
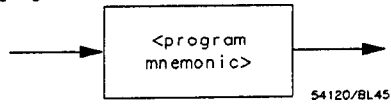


Figure 27-6. Query Message Unit

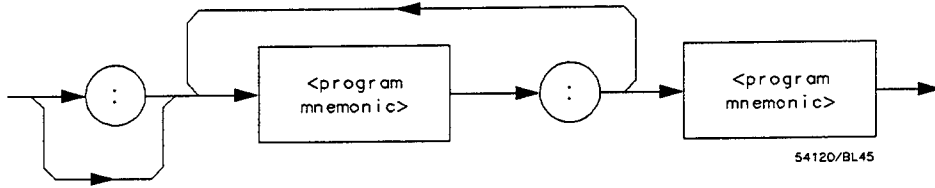
Program Message Unit

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

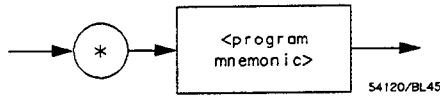
Where simple command program header is defined as



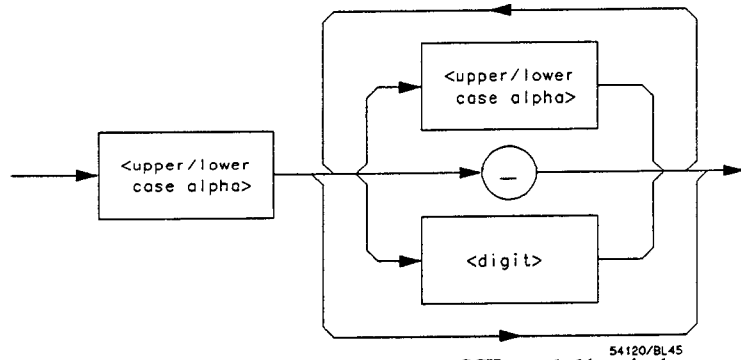
Where compound command program header is defined as



Where common command program header is defined as



Where program mnemonic is defined as

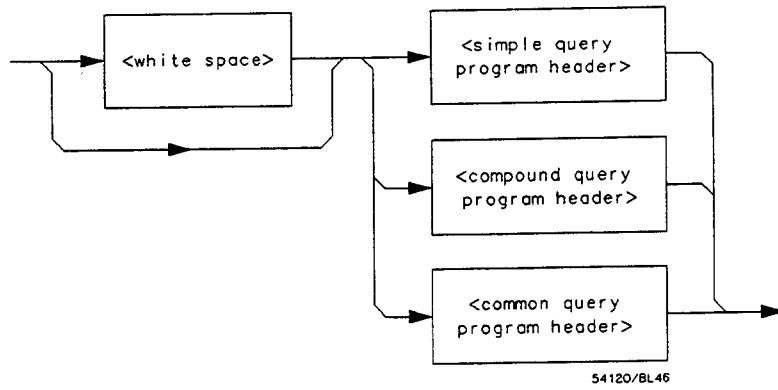


Where upper/lower case alpha is defined as a single ASCII encoded byte in the range 41 - 5A, 61 - 7A (65 - 90, 97 - 122 decimal).

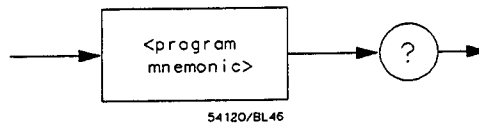
Where digit is defined as a single ASCII encoded byte in the range 30 - 39 (48 - 57 decimal).

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

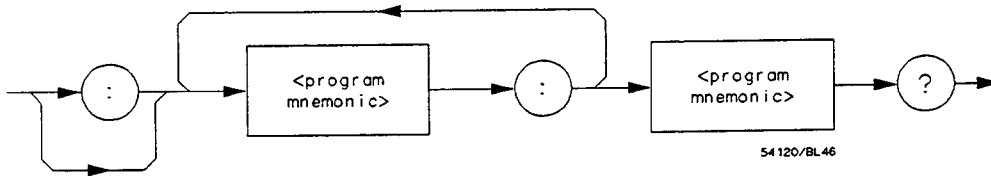
Figure 27-8. Command Program Header (continued)



Where simple query program header is defined as



Where compound query program header is defined as



Where common query program header is defined as

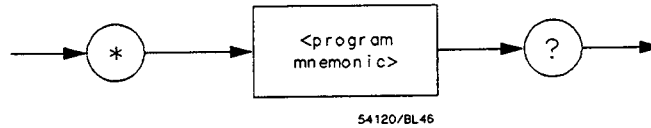


Figure 27-9. Query Program Header

Program Data

The program data element represents the possible types of data which may be sent to the instrument. The oscilloscope accepts the following data types: character program data, decimal numeric program data, suffix program data, string program data, and arbitrary block program data.

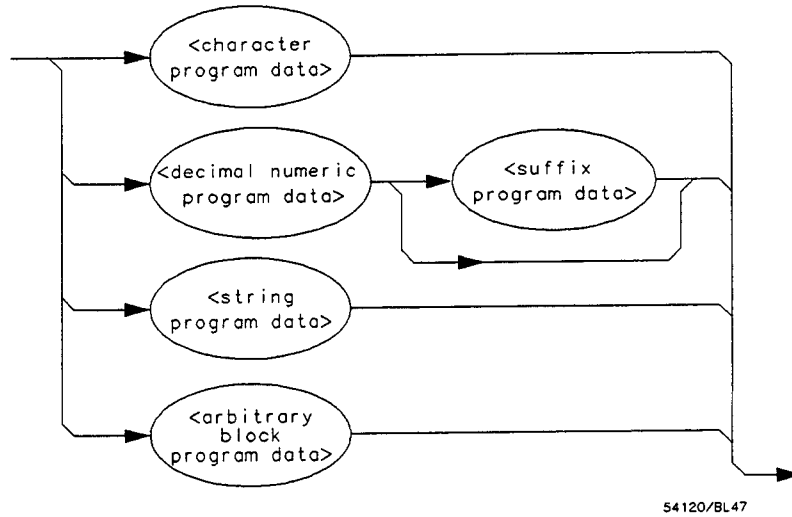


Figure 27-10. Program Data

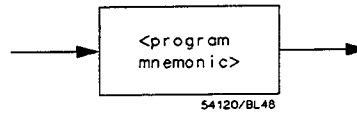
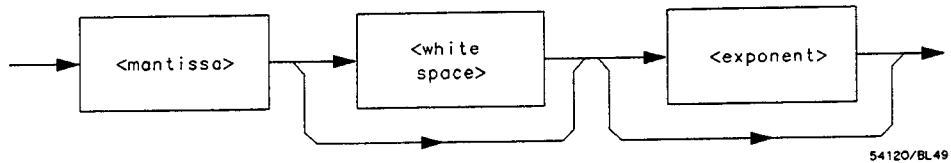
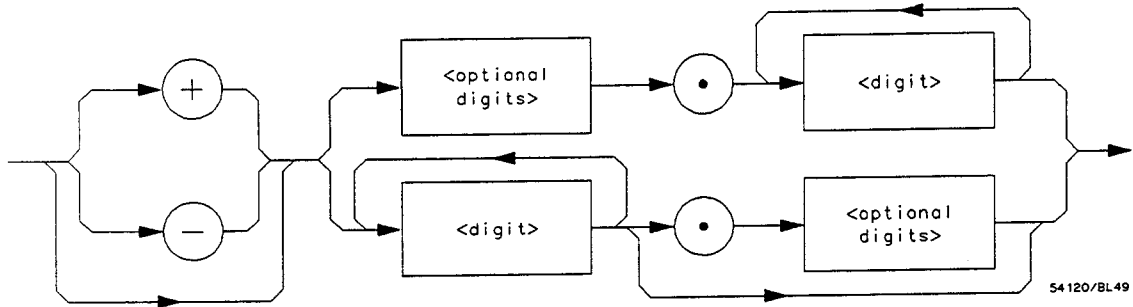


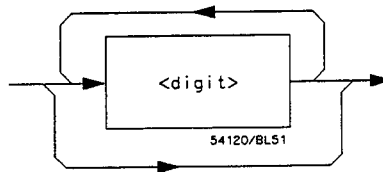
Figure 27-11. Character Program Data



Where mantissa is defined as



Where optional digits is defined as



Where exponent is defined as

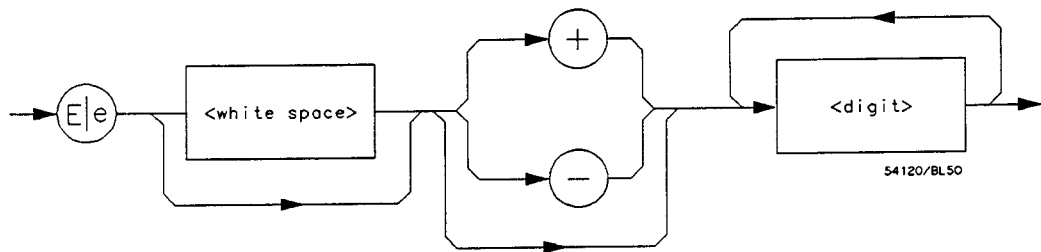


Figure 27-12. Decimal Numeric Program Data

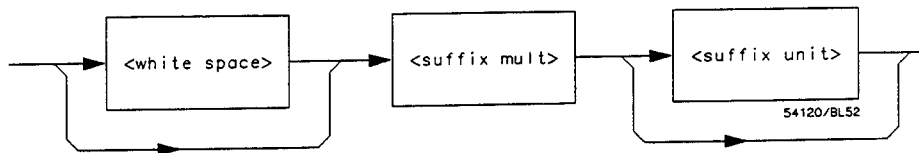


Figure 27-13. Suffix Program Data

Suffix Multiplier

The suffix multipliers that the instrument accepts are shown in table 27-1.

Table 27-1. Suffix Multiplier

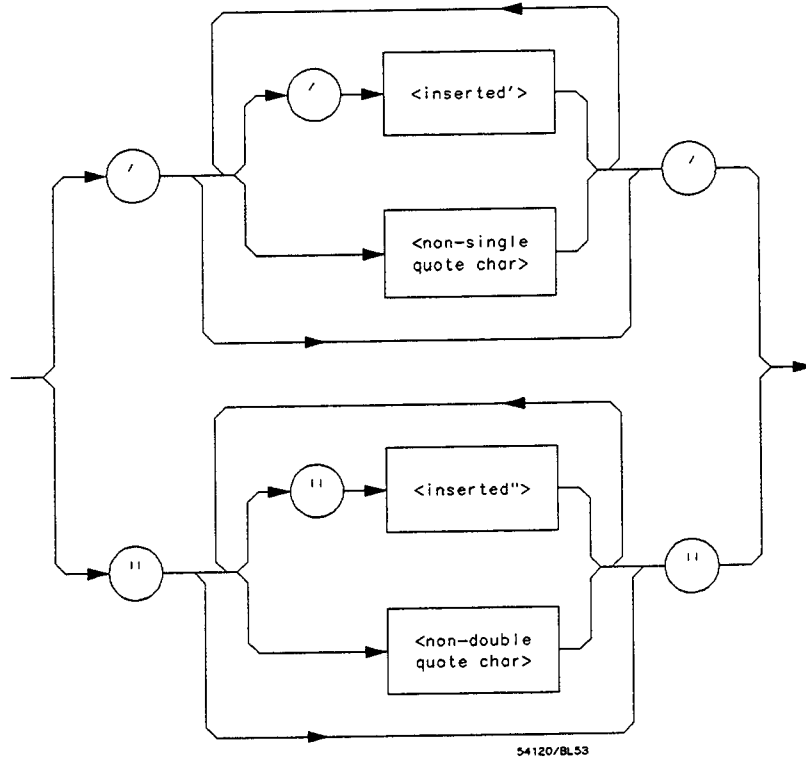
| Value | Mnemonic |
|-------|----------|
| 1E18 | EX |
| 1E15 | PE |
| 1E12 | T |
| 1E9 | G |
| 1E6 | MA |
| 1E3 | K |
| 1E-3 | M |
| 1E-6 | U |
| 1E-9 | N |
| 1E-12 | P |
| 1E-15 | F |
| 1E-18 | A |

Suffix Unit

The suffix units that the instrument accepts are shown in table 27-2.

Table 27-2. Suffix Unit

| Suffix | Referenced Unit | Suffix | Reference Unit |
|--------|-----------------|--------|----------------|
| V | Volt | HZ | Hertz |
| S | Second | PCT | Percent |



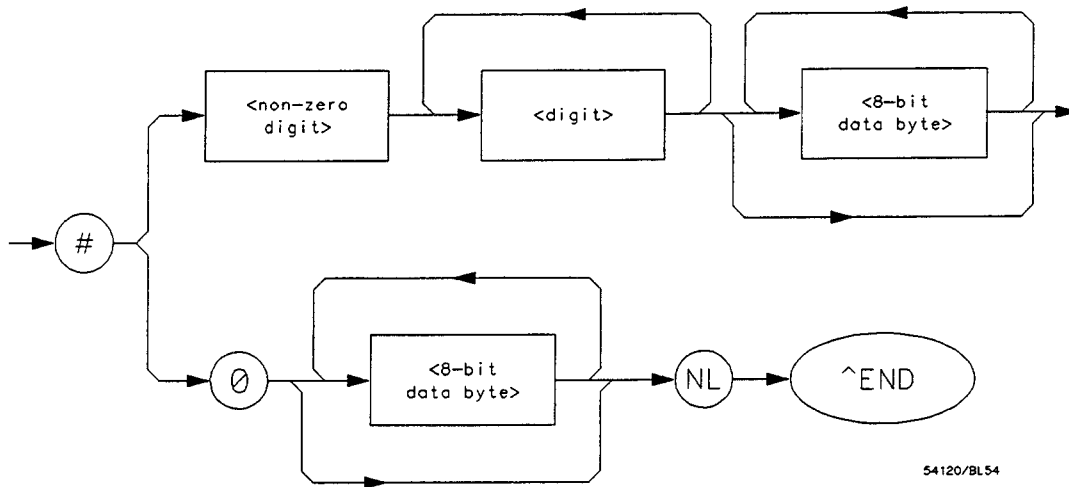
Where <inserted'> is defined as a single ASCII character with the value 27 (39 decimal).

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 27-14. 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 27-15. Arbitrary Block Program Data

Program Data Separator

A comma separates multiple data parameters of a command from one another.

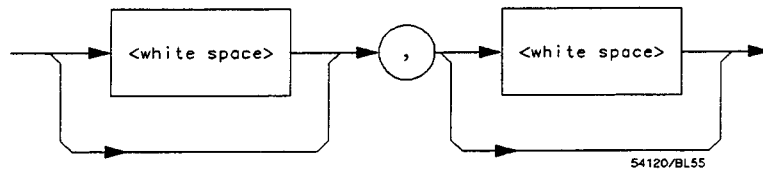


Figure 27-16. Program Data Separator

Program Header Separator

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

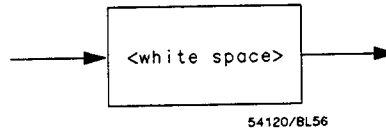
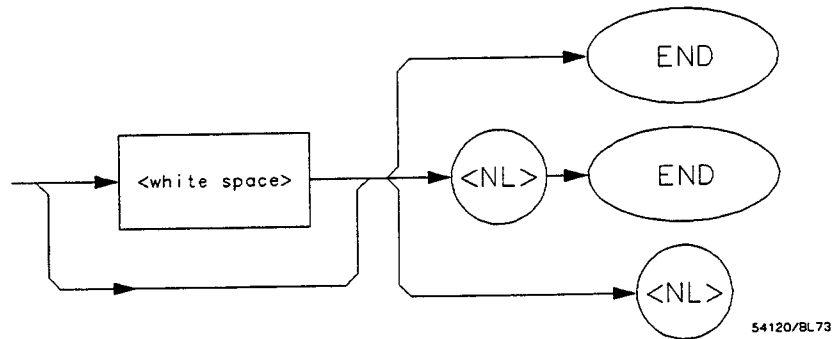


Figure 27-17. Program Header Separator

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 begins 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 27-18. Program Message Terminator

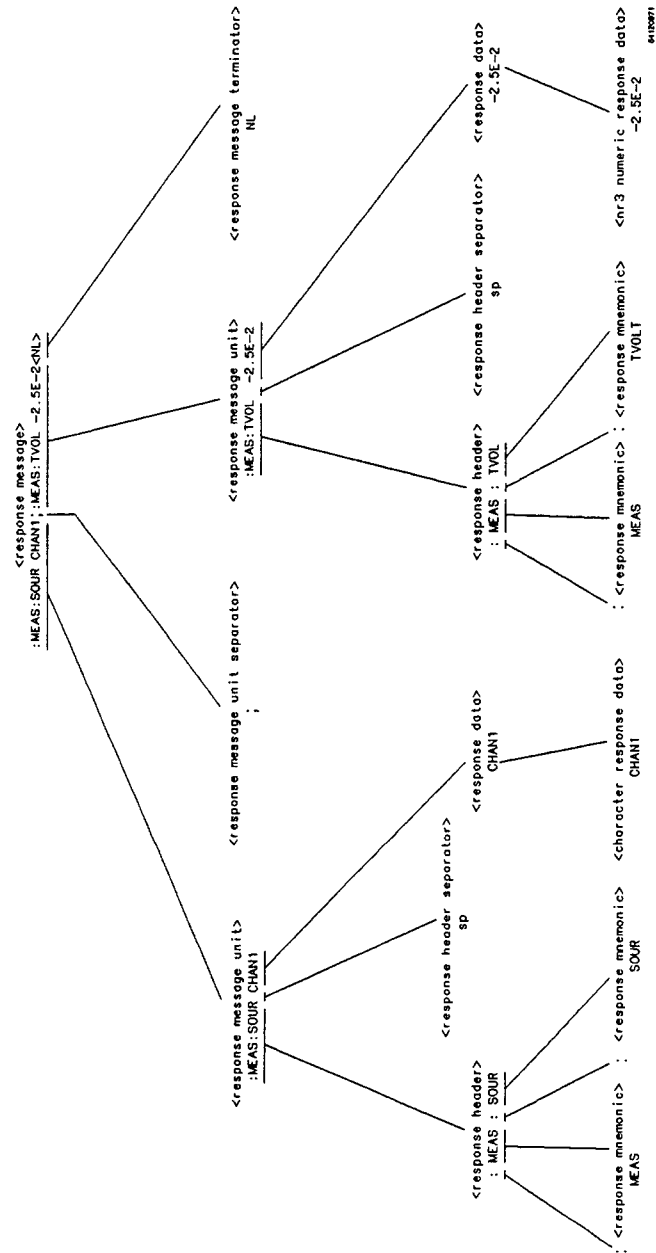


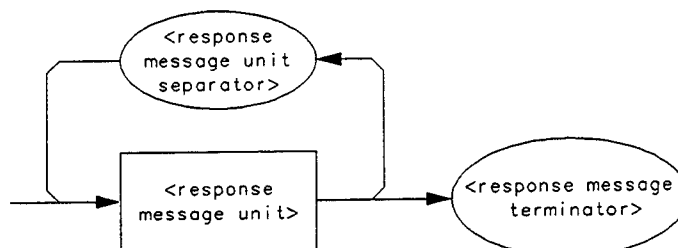
Figure 27-19. Response Message Tree

Device Talking Syntax

The talking syntax of IEEE 488.2 is designed to be more precise than the listening syntax. This allows the programmer to write routines which can 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 does not pad messages which are being sent to the controller with spaces.

Response Message

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



54120/BL57

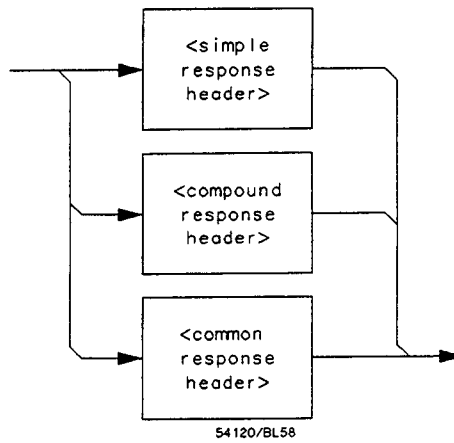
Figure 27-20. Response Message

Response Message Unit

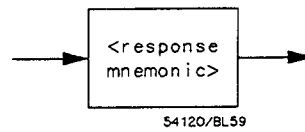
This element serves as the container of individual pieces of a response. Typically a query message unit generates one response message unit, although a query message unit may generate multiple response message units.

Response Header

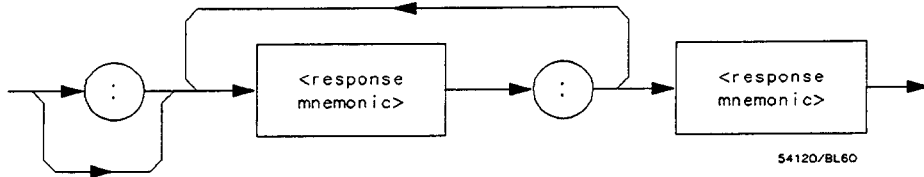
The response header, when returned, indicates what the response data represents.



Where <simple response mnemonic> is defined as



Where <compound response header> is defined as



Where <common response header> is defined as

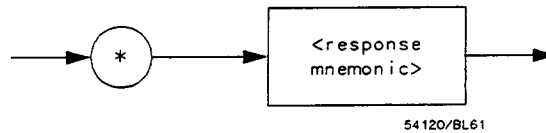
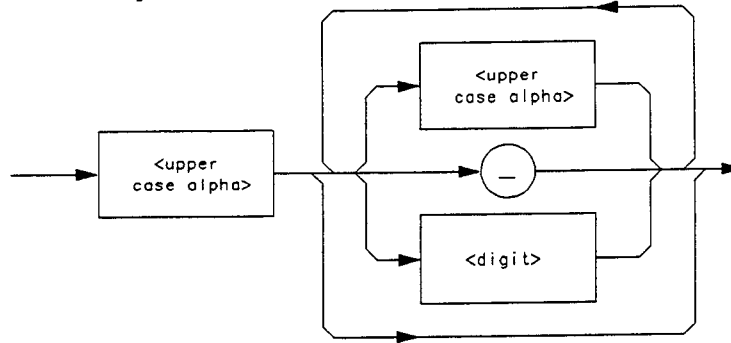


Figure 27-21. Response Message Unit

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). 54120/BL62

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

Figure 27-21. Response Message Unit (continued)

Response Data

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

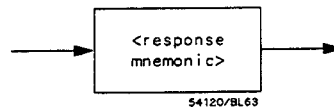


Figure 27-22. Character Response Data

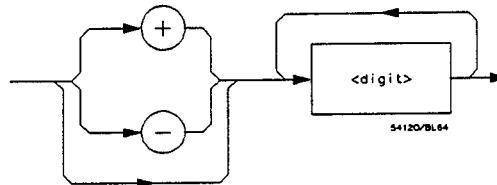


Figure 27-23. NR1 Numeric Response Data

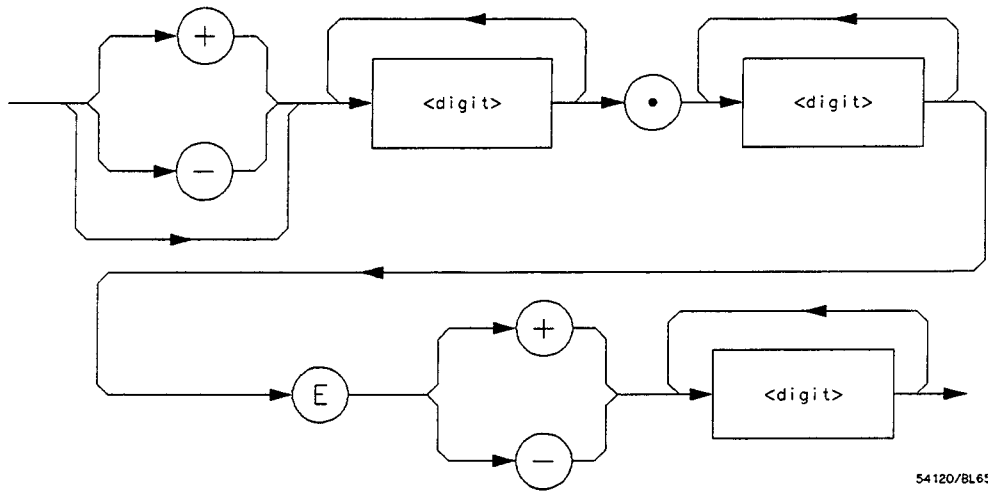


Figure 27-24. NR3 Numeric Response Data

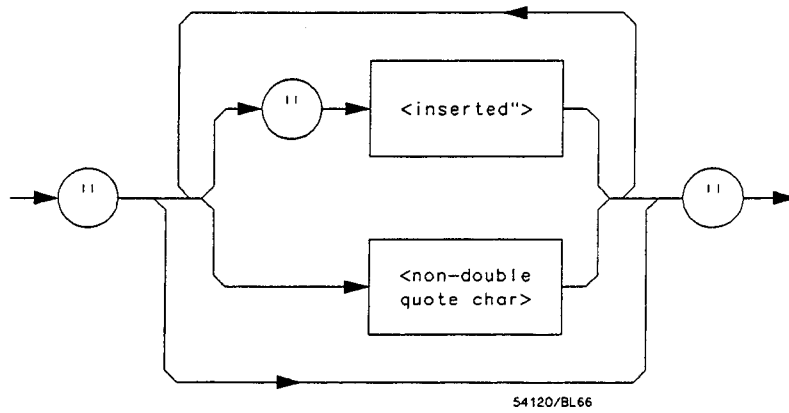


Figure 27-25. String Response Data

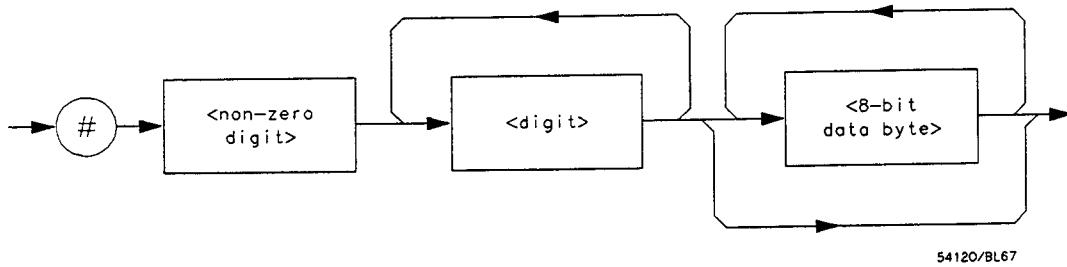
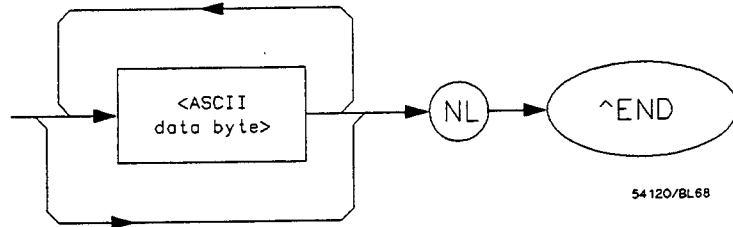


Figure 27-26. Definite Length Arbitrary Block



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

1. The END message provides an unambiguous termination to an element that contains arbitrary ASCII characters.
2. The IEEE 488.1 END message serves the dual function for terminating this element as well as terminating the < RESPONSE MESSAGE >. It is only sent once with the last byte of the indefinite block data. The NL is presented for consistency with the < RESPONSE MESSAGE TERMINATOR >.

Figure 27-27. Arbitrary ASCII Response Data

Response Data Separator

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

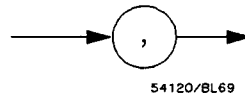


Figure 27-28. Response Data Separator

Response Header Separator

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

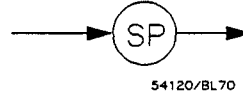


Figure 27-29. Response Header Separator

Response Message Unit Separator

A semicolon delimits the response message units if multiple responses are returned.

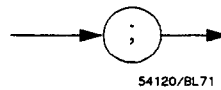


Figure 27-30. Response Message Unit Separator

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.



If you do not read the response message terminator, the oscilloscope produces an interrupted error on the next message.

Common Commands

IEEE 488.2 defines a set of common commands. These commands perform functions which are common to any type of instrument. They can therefore be implemented in a standard way across a wide variety of instrumentation. All the common commands of IEEE 488.2 begin with an asterisk. There is one key difference between the IEEE 488.2 common commands and the rest of the commands found in this instrument. The IEEE 488.2 common commands do not affect the parser's position within the command tree. More information about the command tree and tree traversal can be found in chapter 4, "Programming and Documentation Conventions."

**Table 22-3. HP 54505B/54506B/54510B/54512B
IEEE 488.2 Common Commands**

| Command | Command Name |
|---------|--------------------------------|
| *CLS | Clear Status Command |
| *DMC | Define Macro Command |
| *EMC | Enable Macro Command |
| *EMC? | Enable Macro Query |
| *ESE | Event Status Enable Command |
| *ESE? | Event Status Enable Query |
| *ESR? | Event Status Register Query |
| *GMC? | Get Macro Contents Query |
| *IDN | Identification Query |
| *LMC? | Learn Macro Query |
| *LRN? | Learn Device Setup Query |
| *OPC | Operation Complete Command |
| *OPC? | Operation Complete Query |
| *OPT? | Option Identification Query |
| *PMC | Purge Macro Command |
| *RCL | Recall Command |
| *RST | Reset Command |
| *SAV | Save Command |
| *SRE | Service Request Enable Command |
| *SRE? | Service Request Enable Query |
| *STB? | Read Status Byte Query |

**Table 22-3. HP 54505B/54506B/54510B/54512B
IEEE 488.2 Common Commands (continued)**

| Command | Command Name |
|---------|--------------------------|
| *TRG | Trigger Command |
| *TST? | Self-Test Query |
| *WAI | Wait-to-Continue Command |

Quick Reference Guide

Introduction

The following section lists the commands and queries with their corresponding arguments and returned formats. The arguments for each command list the minimum argument required. The part of the command or query listed in uppercase letters refers to the short form of that command or query. The long form is the combination of uppercase and lowercase letters.

Conventions The following conventions are used in this section:

< > Angular brackets enclose words or characters that symbolize a program code parameter or an HP-IB command.

::= "is defined as." For example, <A> ::= indicates that <A> can be replaced by in any statement containing <A>.

| "or." Indicates a choice of one element from a list. For example, <A> | indicates <A> or but not both.

... An ellipsis (trailing dots) indicate that the preceding element may be repeated one or more times.

[] Square brackets indicate that the enclosed items are optional.

{ } When several items are enclosed by braces, one, and only one of these elements may be selected.

Suffix Multipliers The suffix multipliers available for arguments are:

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

For more information on specific commands or queries, refer to chapters 6 through 23 of this manual.

| | | |
|--|--------------------------------|----------------------|
| *CLS | (Clear Status) | command |
| Command Syntax: *CLS | | |
| *DMC | (Define Macro) | command |
| Command Syntax: *DMC <ascii_string>,<block_data> | | |
| *EMC | (Enable Macro) | command/query |
| Command Syntax: *EMC {{ON 1} {OFF 0}} | | |
| Query Syntax: *EMC? | | |
| Returned Format: {1 0}<NL> | | |
| *ESE | (Event Status Enable) | command/query |
| Command Syntax: *ESE {0 to 255} | | |
| Query Syntax: *ESE? | | |
| Returned Format: {integer, 0 to 255}<NL> | | |
| *ESR | (Event Status Register) | query |
| Query Syntax: *ESR? | | |
| Returned Format: {integer, 0 to 255}<NL> | | |
| *GMC | (Get Macro Contents) | query |
| Query Syntax: *GMC? <macro_name> | | |
| Returned Format: <block_data><NL> | | |
| *IDN | (Identification Number) | query |
| Query Syntax: *IDN? | | |
| Returned Format: HEWLETT-PACKARD,545XXB,XXXXAYYYYY,MMDD<NL> | | |
| *LMC | (Learn Macro Contents) | query |
| Query Syntax: *LMC? | | |
| Returned Format: <ascii_string><NL> | | |

| | | |
|--|---------------------------------|----------------------|
| *PMC | (*PMC) | command |
| Command Syntax: *PMC | | |
| *LRN | (Learn) | query |
| Query Syntax: *LRN? | | |
| Returned Format: :SYSTem:SETup #800001703<learn string><NL> | | |
| *OPC | (Operation Complete) | command/query |
| Command Syntax: *OPC | | |
| Query Syntax: *OPC? | | |
| Returned Format: 1<NL> | | |
| *OPT | (Option) | query |
| Query Syntax: *OPT? | | |
| Returned Format: 0<NL> | | |
| *RCL | (Recall) | command |
| Command Syntax: *RCL {0 to 9} | | |
| *RST | (Reset) | command |
| Command Syntax: *RST | | |
| *SAV | (Save) | command |
| Command Syntax: *SAV {1 to 9} | | |
| *SRE | (Service Request Enable) | command/query |
| Command Syntax: *SRE {0 to 255} | | |
| Query Syntax: *SRE? | | |
| Returned Format: <mask><NL> | | |
| Where: <mask> ::= sum of all bits set - integer, 0 to 255 | | |

| | | |
|--------------------------|---|----------------------|
| *STB | (Status Byte) | query |
| Query Syntax: | *STB? | |
| Returned Format: | {integer, 0 to 255}<NL> | |
| *TRG | (Trigger) | command |
| Command Syntax: | *TRG | |
| *TST | (Test) | query |
| Query Syntax: | *TST? | |
| Returned Format: | {0 or non-zero value}<NL> | |
| Where: | 0 ::= test passed non-zero ::= test failed | |
| *WAI | (Wait) | command |
| Command Syntax: | *WAI | |
| :ACQUIRE:COMPLETE | | command/query |
| Command Syntax: | :ACQUIRE:COMPLETE {0 to 100} | |
| Query Syntax: | :ACQUIRE:COMPLETE? | |
| Returned Format: | [:ACQUIRE:COMPLETE] {integer, 0 to 100}<NL> | |
| :ACQUIRE:COUNT | | command/query |
| Command Syntax: | :ACQUIRE:COUNT {1 to 2048} | |
| Query Syntax: | :ACQUIRE:COUNT? | |
| Returned Format: | [:ACQUIRE:COUNT] {integer, 1 to 2048}<NL> | |
| :ACQUIRE:POINTS | | command/query |
| Command Syntax: | :ACQUIRE:POINTS <points_argument> | |
| Query Syntax: | :ACQUIRE:POINTS? | |
| Returned Format: | [:ACQUIRE:POINTS] <points_argument><NL> | |
| Where: | <points_argument> ::= 500 in repetitive mode 500 or 8000 in real-time mode | |

:ACQUIRE:TYPE**command/query**

Command Syntax: :ACQUIRE:TYPE {NORMAL | AVERAGE | ENVELOPE |
RAWData[,<length>][,<acquisitions>][{NORMAL | AVERAGE | ENVELOPE }]}

Query Syntax: :ACQUIRE:TYPE?

Returned Format: [:ACQUIRE:TYPE] {NORMAL | AVERAGE | ENVELOPE |
RAWData,<length>,<acquisitions>}<NL>

Where: <length> ::= integer, 4 to 8000.
<acquisitions> ::= dependent on length of acquisitions and buffer size.

:AUTOSCALE**command**

Command Syntax: :AUTOSCALE

:BEEP**command/query**

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

Query Syntax: :BEEP?

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

:BLANK**command**

Command Syntax: :BLANK {CHANNEL{1 | 2 | 3 | 4} | FUNCTION{1 | 2 | 3 | 4} | MEMORY{1 | 2 | 3
| 4} | PMEMORY{1 | 2}}

NOTE: CHANNEL13 and CHANNEL14 available only on HP 54512B

:BNC**command/query**

Command Syntax: :BNC {PROBE | TRIGGER}

Query Syntax: :BNC?

Returned Format: [:BNC] {PROBE | TRIGGER}<NL>

:CALIBRATE:DATA:ASCII**query**

Query Syntax: :CALIBRATE:DATA:ASCII?

Returned Format: [:CALIBRATE:DATA:ASCII] <calibration_data>,<calibration_data>,...<NL>

:CALibrate:SETup**query**

Query Syntax: :CALibrate:SETup?
Returned Format: :CALibrate:TNUL1] <null_value2>,<null_value3>,<null_value4><NL>
Where: <null_value> ::= channel 1 to channel 2, 3, and 4 skew

NOTE: <null_value3> and <null_value4> available only on HP 54506B/54512B

:CALibrate:TNUL1**command/query**

Command Syntax: :CALibrate:TNUL1 <null_value2>,<null_value3>,<null_value4>
Query Syntax: :CALibrate:TNUL1?
Returned Format: [:CALibrate:TNUL1] <null_value2>,<null_value3>,<null_value4><NL>
Where: <null_value> ::= channel 1 to channel 2, 3, and 4 skew

NOTE: <null_value3> and <null_value4> available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:COUPling**command/query**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}:COUPling {AC | DC | DCFifty}
Query Syntax: :CHANnel{1 | 2 | 3 | 4}:COUPling?
Returned Format: [:CHANnel{1 | 2 | 3 | 4}:COUPling] {AC | DC | DCFifty}<NL>

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}[:DISP lay]**command/query**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}[:DISP lay] {{ON | 1} | {OFF | 0}}
Query Syntax: :CHANnel{1 | 2 | 3 | 4}[:DISP lay]?
Returned Format: [:CHANnel{1 | 2 | 3 | 4}[:DISP lay] {1 | 0}<NL>

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:ECL**command**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}:ECL

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:HFReject **command/query**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}:HFReject {{ON | 1} | {OFF | 0}}

Query Syntax: :CHANnel{1 | 2 | 3 | 4}:HFReject?

Returned Format: [:CHANnel{1 | 2 | 3 | 4}:HFReject] {1 | 0}<NL>

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:LFReject **command/query**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}:LFReject {{ON | 1} | {OFF | 0}}

Query Syntax: :CHANnel{1 | 2 | 3 | 4}:LFReject?

Returned Format: [:CHANnel{1 | 2 | 3 | 4}:LFReject] {1 | 0}<NL>

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:OFFSet **command/query**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}:OFFSet <offset value>

Query Syntax: :CHANnel{1 | 2 | 3 | 4}:OFFSet?

Returned Format: [:CHANnel{1 | 2 | 3 | 4}:OFFSet] <offset value><NL>

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:PROBe **command/query**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}:PROBe {0.9 to 1000}

Query Syntax: :CHANnel{1 | 2 | 3 | 4}:PROBe?

Returned Format: [:CHANnel{1 | 2 | 3 | 4}:PROBe] {exponential, 0.9 to 1000}<NL>

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:RANGe **command/query**

Command Syntax: :CHANnel{1 | 2 | 3 | 4}:RANGe <full-scale range>

Query Syntax: :CHANnel{1 | 2 | 3 | 4}:RANGe?

Returned Format: [:CHANnel{1 | 2 | 3 | 4}:RANGe] <exponential full-scale range><NL>

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:SETup**query**

Query Syntax: :CHANnel{1 | 2 | 3 | 4}:SETup?
Returned Format: :CHANnel{1 | 2 | 3 | 4}:DISPlay {ON|OFF};RANGE <range>;OFFSET <value>;COUP {AC|DC|DCFiFty};HFReject {ON|OFF};LFReject {ON|OFF};PROBE <attenuation><NL>
Where: <range> ::= exponential full-scale range value
<value> ::= exponential offset value in volts
<attenuation> ::= exponential, 0.9 to 1000

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:CHANnel{1 | 2 | 3 | 4}:TTL**command****Command Syntax:** :CHANnel{1 | 2 | 3 | 4}:TTLNOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:DIGitize**command****Command Syntax:** :DIGitize CHANnel{1 | 2 | 3 | 4}[,CHANnel{1 | 2 | 3 | 4}],CHANnel{1 | 2 | 3 | 4}],CHANnel{1 | 2 | 3 | 4}]**:DISPlay:COLumn****command/query**

Command Syntax: :DISPlay:COLumn {0 to 72}
Query Syntax: :DISPlay:COLumn?
Returned Format: [:DISPlay:COLumn] {integer, 0 to 72}<NL>

:DISPlay:CONNect**command/query**

Command Syntax: :DISPlay:CONNect {{ON | 1} | {OFF | 0}}
Query Syntax: :DISPlay:CONNect?
Returned Format: [:DISPlay:CONNect] {1 | 0}<NL>

:DISPlay:DATA**command/query**

Command Syntax: :DISPlay:DATA #800016576<data>
Query Syntax: :DISPlay:DATA?
Returned Format: [:DISPlay:DATA] #800016576<data><NL>

| | |
|--|----------------------|
| :DISPlay:FORMat | command/query |
| Command Syntax: :DISPlay:FORMat {1 2 4} | |
| Query Syntax: :DISPlay:FORMat? | |
| Returned Format: [:DISPlay:FORMat] {1 2 4}<NL> | |
| :DISPlay:GRATicule | command/query |
| Command Syntax: :DISPlay:GRATicule {OFF GRID AXES FRAME} | |
| Query Syntax: :DISPlay:GRATicule? | |
| Returned Format: [:DISPlay:GRATicule] {OFF GRID AXES FRAME}<NL> | |
| :DISPlay:INVerse | command/query |
| Command Syntax: :DISPlay:INVerse {{ON 1} {OFF 0}} | |
| Query Syntax: :DISPlay:INVerse? | |
| Returned Format: [:DISPlay:INVerse] {1 0}<NL> | |
| :DISPlay:LINE | command |
| Command Syntax: :DISPlay:LINE <quoted string> | |
| :DISPlay:MASK | command/query |
| Command Syntax: :DISPlay:MASK {0 to 255} | |
| Query Syntax: :DISPlay:MASK? | |
| Returned Format: [:DISPlay:MASK] {integer, 0 to 255}<NL> | |
| :DISPlay:PERsistence | command/query |
| Command Syntax: :DISPlay:PERsistence {SINGle INFinite 0.1 to 11} | |
| Query Syntax: :DISPlay:PERsistence? | |
| Returned Format: [:DISPlay:PERsistence] <value><NL> | |
| Where: <value> ::= {0 0.5 to 10 11} in repetitive mode {SINGle INFinite} in real-time mode | |
| :DISPlay:ROW | command/query |
| Command Syntax: :DISPlay:ROW {0 to 24} | |
| Query Syntax: :DISPlay:ROW? | |
| Returned Format: [:DISPlay:ROW] {integer, 0 to 24}<NL> | |

:DISPlay:SCReen **command/query**

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

Query Syntax: :DISPlay:SCReen?

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

:DISPlay:SETup **query**

Query Syntax: :DISPlay:SETup?

Returned Format: :DISPlay:ROW {1 to 24};COL {0 to 72};FORMat {1|2|4};INVerse {ON|OFF};GRATicule {OFF|GRID|AXES|FREMe};PERS {<value>};SCReen {ON|OFF};SOURce {PMEMory{0|1|2|3};MASK <mask>;MARKer {ON|OFF} <NL>

Where: <value> ::= exponential, {0 | .5 to 10 | 11} in the repetitive mode
1.100E+1 = infinite

<value> ::= {SINGle | INFinite} in the real-time mode

<mask> ::= (integer, 0 to 255)

:DISPlay:SOURce **command/query**

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

Query Syntax: :DISPlay:SOURce?

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

:DISPlay:STRing **command**

Command Syntax: :DISPlay:STRing <quoted string>

:DISPlay:TEXT **command**

Command Syntax: :DISPlay:TEXT BLANK

:DISPlay:MARKer|TMARKer|VMARKer **command/query**

Command Syntax: :DISPlay:MARKer|TMARKer|VMARKer {{ON | 1} | {OFF | 0}}

Query Syntax: :DISPlay:MARKer|TMARKer|VMARKer?

Returned Format: [:DISPlay:MARKer|TMARKer|VMARKer] {1 | 0}<NL>

| | |
|--|----------------------|
| :ERASe | command |
| Command Syntax: :ERASe PMemory{0 1 2} | |
| :FUNCTION{1 2 3 4}:ADD | command |
| Command Syntax: :FUNCTION{1 2 3 4}:ADD <operand>,<operand> | |
| Where: <operand> ::= {CHANne1{1 2 3 4} WMemory{1 2 3 4}} | |
| NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B | |
| :FUNCTION{1 2 3 4}:DIFF | command |
| Command Syntax: :FUNCTION{1 2 3 4}:DIFF {CHANne1{1 2 3 4} WMemory{1 2 3 4}} | |
| NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B | |
| :FUNCTION{1 2 3 4}[:DISPLay] | command/query |
| Command Syntax: :FUNCTION{1 2 3 4}[:DISPLay] {{ON 1} {OFF 0}} | |
| Query Syntax: :FUNCTION{1 2 3 4}[:DISPLay]? | |
| Returned Format: [:FUNCTION{1 2 3 4}[:DISPLay] {1 0}<NL> | |
| :FUNCTION{1 2 3 4}:FFT | command |
| Command Syntax: :FUNCTION{1 2 3 4}:FFT {CHANne1{1 2 3 4} WMemory{1 2 3 4}} | |
| NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B | |
| :FUNCTION{1 2 3 4}:FREQuency | command/query |
| Command Syntax: :FUNCTION{1 2 3 4}:FREQuency <center freq value> | |
| Query Syntax: :FUNCTION{1 2 3 4}:FREQuency? | |
| Returned Format: [:FUNCTION{1 2 3 4}:FREQuency] <center freq value><NL> | |

:FUNCTION{1 | 2 | 3 | 4}:INTEgrate **command**

Command Syntax: :FUNCTION{1 | 2 | 3 | 4}:INTEgrate {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:FUNCTION{1 | 2 | 3 | 4}:INVert **command**

Command Syntax: :FUNCTION{1 | 2 | 3 | 4}:INVert {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:FUNCTION{1 | 2 | 3 | 4}:LEVEl **command/query**

Command Syntax: :FUNCTION{1 | 2 | 3 | 4}:LEVEl <search level value>

Query Syntax: :FUNCTION{1 | 2 | 3 | 4}:LEVEl?

Returned Format: [:FUNCTION{1 | 2 | 3 | 4}:LEVEl] <search level value><NL>

:FUNCTION{1 | 2 | 3 | 4}:MAGNify **command/query**

Command Syntax: :FUNCTION{1 | 2 | 3 | 4}:MAGNify {{ON | 1} | {OFF | 0}}

Query Syntax: :FUNCTION{1 | 2 | 3 | 4}:MAGNify?

Returned Format: [:FUNCTION{1 | 2 | 3 | 4}:MAGNify] {1 | 0}<NL>

:FUNCTION{1 | 2 | 3 | 4}:MULTiply **command**

Command Syntax: :FUNCTION{1 | 2 | 3 | 4}:MULTiply <operand>,<operand>

Where: <operand> ::= {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:FUNCTION{1 | 2 | 3 | 4}:OFFSet **command/query**

Command Syntax: :FUNCTION{1 | 2 | 3 | 4}:OFFSet <offset value>

Query Syntax: :FUNCTION{1 | 2 | 3 | 4}:OFFSet?

Returned Format: [:FUNCTION{1 | 2 | 3 | 4}:OFFSet] <exponential offset value><NL>

:FUNCTION{1 | 2 | 3 | 4}:ONLY**command****Command Syntax:** :FUNCTION{1 | 2 | 3 | 4}:ONLY {CHANnel{1 | 2 | 3 | 4} | WMemory{1 | 2 | 3 | 4}}NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:FUNCTION{1 | 2 | 3 | 4}:PEAK**command****Command Syntax:** :FUNCTION{1 | 2 | 3 | 4}:PEAK <peak1>,<peak2>**Where:** <peak> ::= {1 through 99}

:FUNCTION{1 | 2 | 3 | 4}:POINTS**command/query****Command Syntax:** :FUNCTION{1 | 2 | 3 | 4}:POINTS (512 | 1024 | 2048 | 4096 | 8192)**Query Syntax:** :FUNCTION{1 | 2 | 3 | 4}:POINTS?**Returned Format:** [:FUNCTION{1 | 2 | 3 | 4}:POINTS] (512 | 1024 | 2048 | 4096 | 8192)<NL>

:FUNCTION{1 | 2 | 3 | 4}:RANGE**command/query****Command Syntax:** :FUNCTION{1 | 2 | 3 | 4}:RANGE <range setting>**Query Syntax:** :FUNCTION{1 | 2 | 3 | 4}:RANGE?**Returned Format:** [:FUNCTION{1 | 2 | 3 | 4}:RANGE] <exponential range setting><NL>

:FUNCTION{1 | 2 | 3 | 4}:SETup**query****Query Syntax:** :FUNCTION{1 | 2 | 3 | 4}:SETup?**Returned Format:** :FUNCTION<N>:FREQuency<value>;LEVel<level>;MAGNify{ON|OFF};OFFSet<offset>;POINts<points>;RANGE<range>;SPAN<frequency>;WINDow{RECT|HANN|FLAT}<NL>**Where:**
<value> ::= exponential freq in hertz from 0 to 1.5X span
<level> ::= 0 +\ -600 dBm
<offset> ::= exponential offset value in volts or +\ -200 dBm for FFT
<points> ::= {512 | 1024 | 2048 | 4096 | 8192}
<range> ::= exponential full-scale range value
<frequency> ::= exponential freq in hertz

:FUNCTION{1 | 2 | 3 | 4}:SPAN**command/query****Command Syntax:** :FUNCTION{1 | 2 | 3 | 4}:SPAN <freq value>**Query Syntax:** :FUNCTION{1 | 2 | 3 | 4}:SPAN?**Returned Format:** [:FUNCTION{1 | 2 | 3 | 4}:SPAN] <freq value><NL>

:FUNCTION{1 | 2 | 3 | 4}:SUBTRACT **command**

Command Syntax: :FUNCTION{1 | 2}:SUBTRACT <operand>,<operand>

Where: <operand> ::= {CHANNEL{1 | 2 | 3 | 4} | WMEMORY{1 | 2 | 3 | 4}}

NOTE: CHANNEL3 and CHANNEL4 available only on HP 54506B/54512B

:FUNCTION{1 | 2 | 3 | 4}:VERSUS **command**

Command Syntax: :FUNCTION{1 | 2}:VERSUS <operand>,<operand>

Where: <operand> ::= {CHANNEL{1 | 2 | 3 | 4} | WMEMORY{1 | 2 | 3 | 4}}

NOTE: CHANNEL3 and CHANNEL4 available only on HP 54506B/54512B

:FUNCTION{1 | 2 | 3 | 4}:WINDOW **command/query**

Command Syntax: :FUNCTION:WINDOW {RECTANGULAR | HANNING | FLATTOP}

Query Syntax: :FUNCTION:WINDOW?

Returned Format: [:HARDcopy:PAGE] {RECTANGULAR | HANNING | FLATTOP}<NL>

:HARDcopy:LENGTH **command/query**

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

Query Syntax: :HARDcopy:LENGTH?

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

:HARDcopy:MODE **command/query**

Command Syntax: :HARDcopy:MODE {PRINT | PLOT}

Query Syntax: :HARDcopy:MODE?

Returned Format: [:HARDcopy:MODE] {PRINT | PLOT}<NL>

:HARDcopy:PAGE **command/query**

Command Syntax: :HARDcopy:PAGE {MANUAL | AUTOMATIC}

Query Syntax: :HARDcopy:PAGE?

Returned Format: [:HARDcopy:PAGE] {MANUAL | AUTOMATIC}<NL>

:HARDcopy:PLOT:AREA**command/query**

Command Syntax: :HARDcopy:PLOT:AREA {ALL | DISPlay | FACTors | GRATICuLe | LABeLed}
Query Syntax: :HARDcopy:PLOT:AREA?
Returned Format: [:HARDcopy:PLOT:AREA] {ALL | DISPlay | FACTors | GRATICuLe | LABeLed}<NL>

:HARDcopy:PLOT:INITialize**command/query**

Command Syntax: :HARDcopy:PLOT:INITialize {{ON | 1} | {OFF | 0}}
Query Syntax: :HARDcopy:PLOT:INITialize?
Returned Format: [:HARDcopy:PLOT:INITialize] {1 | 0}<NL>

:HARDcopy:PLOT:PEN|COLor**command/query**

Command Syntax: :HARDcopy:PLOT:PEN|COLor <item>,{0 to 8}
Query Syntax: :HARDcopy:PLOT:PEN|COLor? <item>
Returned Format: [:HARDcopy:PLOT:PEN|COLor] {0 to 8}<NL>
Where: <item> ::= {CHANneL{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4} | FUNCTion{1 | 2 | 3 | 4} | PMEMory{1 | 2} | VMARKer{1 | 2} | STARTmarker | STOPmarker | GRATICuLe | TRIGger | TIMEbase | MEASure | TITLes}

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:LER**(Local Event Register)****query**

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

:LTER**(Limit Test Event Register)****query**

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

:MARKer[:DISPlay]**command/query**

Command Syntax: :MARKer[:DISPlay] {{ON | 1} | {OFF | 0}}
Query Syntax: :MARKer[:DISPlay]?
Returned Format: [:MARKer[:DISPlay]] {1 | 0}<NL>

:MARKer:SETup**query**

Query Syntax: :MARKer:SETup?
Returned Format: :MARKer:DISPlay {ON|OFF};XDELta <xdelta>;X1Position <xposition>;X2Position <xposition>;X1Y1source {CHANne1|2|3|4} | {FUNction1|2|3|4} | {WMEMory1|2|3|4};X2Y2source {CHANne1|2|3|4} | {FUNction1|2|3|4} | {WMEMory1|2|3|4};YDELta <ydelta>;Y1Position <yposition>;Y2Position <yposition><NL>
Where: <xdelta>::= exponential difference between x1 and x2 markers
<xposition>::= exponential xmarker position in seconds or hertz
<ydelta>::= exponential difference between y1 and y2 markers
<yposition>::= exponential ymarker position in volts or power

:MARKer:XDELta**query**

Query Syntax: :MARKer:XDELta?
Returned Format: [:MARKer:XDELta] <xdelta><NL>
Where: <xdelta>::= exponential difference between x1 and x2 markers

:MARKer:X1Position**command/query**

Command Syntax: :MARKer:X1Position <xposition>
Query Syntax: :MARKer:X1Position?
Returned Format: [:MARKer:X1Position] <xposition><NL>
Where: <xposition>::= exponential xmarker position in seconds or hertz

:MARKer:X2Position**command/query**

Command Syntax: :MARKer:X2Position <xposition>
Query Syntax: :MARKer:X2Position?
Returned Format: [:MARKer:X2Position] <xposition><NL>
Where: <xposition>::= exponential x2marker position in seconds or hertz

:MARKer:X1Y1source**command/query**

Command Syntax: :MARKer:X1Y1source {CHANne1{1 | 2 | 3 | 4} | {FUNction{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}
Query Syntax: :MARKer:X1Y1source?
Returned Format: [:MARKer:X1Y1source] {CHANne1{1 | 2 | 3 | 4} | {FUNction{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}<NL>

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:MARKer:X2Y2source**command/query**

Command Syntax: :MARKer:X2Y2source {CHANne1{1 | 2 | 3 | 4} | {FUNctio{n}{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}
Query Syntax: :MARKer:X2Y2source?
Returned Format: [:MARKer:X2Y2source] {CHANne1{1 | 2 | 3 | 4} | {FUNctio{n}{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}<NL>

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:MARKer:YDELta**query**

Query Syntax: :MARKer:YDELta?
Returned Format: [:MARKer:YDELta] <ydelta><NL>
Where: <ydelta>:= exponential difference between y1 and y2 markers

:MARKer:Y1Position**command/query**

Command Syntax: :MARKer:Y1Position <yposition>
Query Syntax: :MARKer:Y1Position?
Returned Format: [:MARKer:Y1Position] <yposition><NL>
Where: <yposition>:= exponential y1marker position in volts or power

:MARKer:Y2Position**command/query**

Command Syntax: :MARKer:Y2Position <yposition>
Query Syntax: :MARKer:Y2Position?
Returned Format: [:MARKer:Y2Position] <yposition><NL>
Where: <yposition>:= exponential y1marker position in volts or power

:MEASure:ALL**query**

Query Syntax: :MEASure:ALL?
Returned Format: [:MEASure:FREQuency] <result>; [PERiod] <result>; [PWIDth] <result>; [NWIDth] <result>; [RISetime] <result>; [FALLtime] <result>; [VAMPlitude] <result>; [VPP] <result>; [PREShoot] <result>; [OVERshoot] <result>; [DUTycycle] <result>; [VRMS] <result>; [VMAX] <result>; [VMIN] <result>; [VTOP] <result>; [VBASe] <result>; [VAverage] <result><NL>

:MEASure:COMPare**command/query**

Command Syntax: :MEASure:COMPare <measurement>,<upper_limit>,<lower_limit>
Query Syntax: :MEASure:COMPare? <measurement>
Returned Format: [:MEASure:COMPare] <measurement>,<upper_limit>,<lower_limit><NL>
Where: <measurement> ::= {RISetime | FALLtime | FREQuency | PERiod | PWIDth |
NWIDth | VAMPLitude | VBASE | VTOP | VPP | VAverage | VMAX | VMIN | VRMS |
DUTYcycle | DELay}

:MEASure:CURSor**query**

Query Syntax: :MEASure:CURSor? {DELTA | START | STOP}
Returned Format: [:MEASure:CURSor] <time>,<voltage><NL>

:MEASure:DEFine**command/query**

Command Syntax: :MEASure:DEFine <measurement_spec>
Query Syntax: :MEASure:DEFine? {DELay | PWIDth | NWIDth}
Returned Format: [:MEASure:DEFine] <measurement_spec><NL>
Where: <measurement_spec> ::= {DELay,<polarity>,<edge_number>,<level>,<polarity>,
<edge_number>,<level> | PWIDth,<level> | NWIDth,<level>}
<polarity> ::= {POSitive | NEGative}
<edge_number> ::= integer, 1 to 4000
<level> ::= {MIDDLE | UPPER | LOWER}

:MEASure:DELay**command/query**

Command Syntax: :MEASure:DELay
Query Syntax: :MEASure:DELay?
Returned Format: [:MEASure:DELay] <delay_value><NL>

:MEASure:DESTination**command/query**

Command Syntax: :MEASure:DESTination {CHANnel{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4} |
PMEMemory{1 | 2} | OFF}
Query Syntax: :MEASure:DESTination?
Returned Format: [:MEASure:DESTination] {CHANnel{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4} |
PMEMemory{1 | 2} | OFF}<NL>

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:MEASure:DUTcycle**command/query**

Command Syntax: :MEASure:DUTcycle
Query Syntax: :MEASure:DUTcycle?
Returned Format: [:MEASure:DUTcycle] <value><NL>

:MEASure:ESTart**command/query**

Command Syntax: :MEASure:ESTart <edge_number>
Query Syntax: :MEASure:ESTart?
Returned Format: [:MEASure:ESTart] <edge_number><NL>

:MEASure:ESTop**command/query**

Command Syntax: :MEASure:ESTop <edge_number>
Query Syntax: :MEASure:ESTop?
Returned Format: [:MEASure:ESTop] <edge_number><NL>

:MEASure:FALLtime**command/query**

Command Syntax: :MEASure:FALLtime
Query Syntax: :MEASure:FALLtime?
Returned Format: [:MEASure:FALLtime] <value><NL>

:MEASure:FREQuency**command/query**

Command Syntax: :MEASure:FREQuency
Query Syntax: :MEASure:FREQuency?
Returned Format: [:MEASure:FREQuency] <value><NL>

:MEASure:LIMittest**command**

Command Syntax: :MEASure:LIMittest {MEASure | OFF}

:MEASure:LOWer**command/query**

Command Syntax: :MEASure:LOWer <lower_threshold>
Query Syntax: :MEASure:LOWer?
Returned Format: [:MEASure:LOWer] <lower_threshold><NL>

:MEASure:MODE **command/query**

Command Syntax: :MEASure:MODE {STANdard | USER}
Query Syntax: :MEASure:MODE?
Returned Format: [:MEASure:MODE] {STANdard | USER}<NL>

:MEASure:NWIDth **command/query**

Command Syntax: :MEASure:NWIDth
Query Syntax: :MEASure:NWIDth?
Returned Format: [:MEASure:NWIDth] <negat ive_width><NL>

:MEASure:OVERshoot **command/query**

Command Syntax: :MEASure:OVERshoot
Query Syntax: :MEASure:OVERshoot?
Returned Format: [:MEASure:OVERshoot] <va lue><NL>

:MEASure:PERiod **command/query**

Command Syntax: :MEASure:PERiod
Query Syntax: :MEASure:PERiod?
Returned Format: [:MEASure:PERiod] <va lue><NL>

:MEASure:POSTfailure **command/query**

Command Syntax: :MEASure:POSTfailure {CONTInue | STOP}
Query Syntax: :MEASure:POSTfailure?
Returned Format: [:MEASure:POSTfailure] {CONTInue | STOP}<NL>

:MEASure:PREShoot **command/query**

Command Syntax: :MEASure:PREShoot
Query Syntax: :MEASure:PREShoot?
Returned Format: [:MEASure:PREShoot] <va lue><NL>

:MEASure:PWIDth **command/query**

Command Syntax: :MEASure:PWIDth
Query Syntax: :MEASure:PWIDth?
Returned Format: [:MEASure:PWIDth] <positive_width><NL>

:MEASure:RESults **query**

Query Syntax: :MEASure:RESults?
Returned Format: [:MEASure:RESults] {integer, 0 to 8}[:<measurement> <result>]...<NL>

:MEASure:RISetime **command/query**

Command Syntax: :MEASure:RISet ime
Query Syntax: :MEASure:RISet ime?
Returned Format: [:MEASure:RISet ime] <va lue><NL>

:MEASure:SCRatch **(Clear Results)** **command**

Command Syntax: :MEASure:SCRatch

:MEASure:SOURce **command/query**

Command Syntax: :MEASure:SOURce <source>[,<source>]
Query Syntax: :MEASure:SOURce?
Returned Format: [:MEASure:SOURce] <source>[,<source>]<NL>
Where: <source> ::= {CHANne1{1 | 2 | 3 | 4} | FUNctio{n}{1 | 2 | 3 | 4} | WMEMo{1 | 2 | 3 | 4}}

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:MEASure:STATistics **command/query**

Command Syntax: :MEASure:STATistics {{ON | 1} | {OFF | 0}}
Query Syntax: :MEASure:STATistics?
Returned Format: [:MEASure:STATistics] {1 | 0}<NL>

:MEASure:TDELta **query**

Query Syntax: :MEASure:TDELta?
Returned Format: [:MEASure:TDELta] <va lue><NL>

| | |
|---|----------------------|
| :MEASure:TMAX | query |
| Query Syntax: :MEASure:TMAX? | |
| Returned Format: [:MEASure:TMAX] <time at maximum voltage><NL> | |
| :MEASure:TMIN | query |
| Query Syntax: :MEASure:TMIN? | |
| Returned Format: [:MEASure:TMIN] <time at minimum voltage><NL> | |
| :MEASure:TSTArt | command/query |
| Command Syntax: :MEASure:TSTArt <start marker time> | |
| Query Syntax: :MEASure:TSTArt? | |
| Returned Format: [:MEASure:TSTArt] <start marker time><NL> | |
| :MEASure:TSTOp | command/query |
| Command Syntax: :MEASure:TSTOp <stop marker time> | |
| Query Syntax: :MEASure:TSTOp? | |
| Returned Format: [:MEASure:TSTOp] <stop marker time><NL> | |
| :MEASure:TVOLT | query |
| Query Syntax: :MEASure:TVOLT? <voltage>,<slope><occurrence> | |
| Returned Format: [:MEASure:TVOLT] <time of voltage crossing><NL> | |
| :MEASure:UNITs | command/query |
| Command Syntax: :MEASure:UNITs {PERCent VOLTs} | |
| Query Syntax: :MEASure:UNITs? | |
| Returned Format: [:MEASure:UNITs] {PERCent VOLTs}<NL> | |
| :MEASure:UPPer | command/query |
| Command Syntax: :MEASure:UPPer <upper_threshold> | |
| Query Syntax: :MEASure:UPPer? | |
| Returned Format: [:MEASure:UPPer] <upper_threshold><NL> | |

:MEASure:VACRms **command/query**

Command Syntax: :MEASure:VACRms
Query Syntax: :MEASure:VACRms?
Returned Format: [:MEASure:VACRms] <ac_rms voltage><NL>

:MEASure:VAMPLitude **command/query**

Command Syntax: :MEASure:VAMPLitude
Query Syntax: :MEASure:VAMPLitude?
Returned Format: [:MEASure:VAMPLitude] <value><NL>

:MEASure:VAverage **command/query**

Command Syntax: :MEASure:VAverage
Query Syntax: :MEASure:VAverage?
Returned Format: [:MEASure:VAverage] <average voltage><NL>

:MEASure:VBASe **command/query**

Command Syntax: :MEASure:VBASe
Query Syntax: :MEASure:VBASe?
Returned Format: [:MEASure:VBASe] <base voltage><NL>

:MEASure:VDCRms **command/query**

Command Syntax: :MEASure:VDCRms
Query Syntax: :MEASure:VDCRms?
Returned Format: [:MEASure:VDCRms] <dc_rms voltage><NL>

:MEASure:VDELta **query**

Query Syntax: :MEASure:VDELta?
Returned Format: [:MEASure:VDELta] <value><NL>

:MEASure:VFIFty **command**

Command Syntax: :MEASure:VFIFty

:MEASure:VMAX **command/query**

Command Syntax: :MEASure:VMAX
Query Syntax: :MEASure:VMAX?
Returned Format: [:MEASure:VMAX] <maximum voltage><NL>

:MEASure:VMIN **command/query**

Command Syntax: :MEASure:VMIN
Query Syntax: :MEASure:VMIN?
Returned Format: [:MEASure:VMIN] <minimum voltage><NL>

:MEASure:VPP **command/query**

Command Syntax: :MEASure:VPP
Query Syntax: :MEASure:VPP?
Returned Format: [:MEASure:VPP] <peak-to-peak voltage><NL>

:MEASure:VRELative **command/query**

Command Syntax: :MEASure:VRELative {0 to 100}
Query Syntax: :MEASure:VRELative?
Returned Format: [:MEASure:VRELative] {integer, 50 to 100}<NL>

:MEASure:VRMS **(AC RMS)** **command/query**

Command Syntax: :MEASure:VRMS
Query Syntax: :MEASure:VRMS?
Returned Format: [:MEASure:VRMS] <ac_rms voltage><NL>

:MEASure:VSTArt **command/query**

Command Syntax: :MEASure:VSTArt <y1marker voltage>
Query Syntax: :MEASure:VSTArt?
Returned Format: [:MEASure:VSTArt] <y1marker voltage><NL>

:MEASure:VSTOp **command/query**

Command Syntax: :MEASure:VSTOp <y2marker voltage>
Query Syntax: :MEASure:VSTOp?
Returned Format: [:MEASure:VSTOp] <y2marker voltage><NL>

:MEASure:VTIME**query**

Query Syntax: :MEASure:VTIME? <time from trigger>
Returned Format: [:MEASure:VTIME] <voltage at specified time><NL>

:MEASure:VTOP**command/query**

Command Syntax: :MEASure:VTOP
Query Syntax: :MEASure:VTOP?
Returned Format: [:MEASure:VTOP] <top_voltage><NL>

:MEASure:WCOMpare:ALLOWance**command/query**

Command Syntax: :MEASure:WCOMpare:ALLOWance <allow value>
Query Syntax: :MEASure:WCOMpare:ALLOWance?
Returned Format: [:MEASure:WCOMpare:ALLOWance] <allow value><NL>

:MEASure:WCOMpare:COMPare**command/query**

Command Syntax: :MEASure:WCOMpare:COMPare {CHANne1{1 | 2 | 3 | 4} | FUNctioN{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}
Query Syntax: :MEASure:WCOMpare:COMPare?
Returned Format: [:MEASure:WCOMpare:COMPare] {CHANne1{1 | 2 | 3 | 4} | FUNctioN{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}<NL>

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:MEASure:WCOMpare:DESTination**command/query**

Command Syntax: :MEASure:WCOMpare:DESTination {{CHANne1{1 | 2 | 3 | 4} | FUNctioN{1 | 2 | 3 | 4}},{OFF | WMEMory{1 | 2 | 3 | 4} | MULTiple}} | {{SCReen},{OFF | PMEMory{1 | 2}}}}

Query Syntax: :MEASure:WCOMpare:DESTination? {CHANne1{1 | 2 | 3 | 4} | FUNctioN{1 | 2 | 3 | 4} | SCReen}

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

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:MEASure:WCOMpare:POSTfailure **command/query**

Command Syntax: :MEASure:WCOMpare:POSTfailure {CONTInue | STOP}
Query Syntax: :MEASure:WCOMpare:POSTfailure?
Returned Format: [:MEASure:WCOMpare:POSTfailure] {CONTInue | STOP}<NL>

:MEASure:WCOMpare:WTEST **command/query**

Command Syntax: :MEASure:WCOMpare:WTEST {MEASure | OFF}
Query Syntax: :MEASure:WCOMpare:WTEST?
Returned Format: [:MEASure:WCOMpare:WTEST] {MEASure | OFF}<NL>

:MENU **command/query**

Command Syntax: :MENU {TIMebase | CHANnel | TRIGger | DISPlay | DELTa | MATH | SAVE | MEASure | UTILity | SHOW}
Query Syntax: :MENU?
Returned Format: [:MENU] {TIMebase | CHANnel | TRIGger | DISPlay | DELTa | MATH | SAVE | MEASure | UTILity | SHOW}<NL>

:MERGE **command**

Command Syntax: :MERGE PMEMory{1 | 2}

:MMEMory[:DISPlay] **command/query**

Command Syntax: :MMEMory[:DISPlay] {{ON | 1} | {OFF | 0}}
Query Syntax: :MMEMory[:DISPlay]?
Returned Format: [:MMEMory[:DISPlay]] {1 | 0}<NL>

:MMEMory:FNUMber **command/query**

Command Syntax: :MMEMory:FNUMber {integer, 1 to 665}
Query Syntax: :MMEMory:FNUMber?
Returned Format: [:MMEMory:FNUMber {integer, 1 to 665}<NL>

:MMEMemory:SOURce **command/query**

Command Syntax: :MMEMemory:SOURce {CHANne1{1 | 2 | 3 | 4} | FUNction{1 | 2 | 3 | 4}}

Query Syntax: :MMEMemory:SOURce?

Returned Format: [:MMEMemory:SOURce] {CHANne1{1 | 2 | 3 | 4} | FUNction{1 | 2 | 3 | 4}}

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:MMEMemory:STORE **command**

Command Syntax: :MMEMemory:STORE

:PMEMemory{1 | 2}:CLEAr **command**

Command Syntax: :PMEMemory{1 | 2}:CLEAr

:PMEMemory{1 | 2}[:DISPlay] **command/query**

Command Syntax: :PMEMemory{1 | 2}[:DISPlay] {{ON | 1} | {OFF | 0}}

Query Syntax: :PMEMemory{1 | 2}[:DISPlay]?

Returned Format: [:PMEMemory{1 | 2}[:DISPlay]] {1 | 0}<NL>

:PMEMemory{1 | 2}:MERGe **command**

Command Syntax: :PMEMemory{1 | 2}:MERGe

:PMEMemory{1 | 2}:SETup **query**

Query Syntax: :PMEMemory{1 | 2}SETup?

Returned Format: PMEMemory{1 | 2}:DISPlay {ON | OFF}

:PLOT **query**

Query Syntax: :PLOT?

:PRINT **query**

Query Syntax: :PRINT?

:RUN**command****Command Syntax:** :RUN

:SEquential[:DISPlay]**command/query****Command Syntax:** :SEquential[:DISPlay] {{ON | 1} | {OFF | 0}}**Query Syntax:** :SEquential[:DISPlay]?**Returned Format:** [:SEquential[:DISPlay]] {1 | 0}<NL>

:SEquential:EXCLude**command/query****Command Syntax:** :SEquential:EXCLude <from>,<to>**Query Syntax:** :SEquential:EXCLude?**Returned Format:** [:SEquential:EXCLude] <exclude list><NL>**Where:** <from> ::= interer, any previously captured segment number less than or equal to the value of <to>

<to> ::= interer, any previously captured segment number greater than or equal to the value of <from>

<exclude list> ::= list of excluded segments separated by commas

:SEquential:INCLude**command/query****Command Syntax:** :SEquential:INCLude <from>,<to>**Query Syntax:** :SEquential:INCLude?**Returned Format:** [:SEquential:INCLude] <include list><NL>**Where:** <from> ::= interer, any previously captured segment number less than or equal to the value of <to>

<to> ::= interer, any previously captured segment number greater than or equal to the value of <from>

<include list> ::= list of included segments separated by commas

:SEquential:NPOints**command/query****Command Syntax:** :SEquential:NPOints {4 to 8000}**Query Syntax:** :SEquential:NPOints?**Returned Format:** [:SEquential:NPOints] {integer, 4 to 8000}<NL>

:SEquential:NSEgments**command/query**

Command Syntax: :SEquential:NSEgments {1 to 8888}
Query Syntax: :SEquential:NSEgments?
Returned Format: [:SEquential:NSEgments] {integer, 1 to 8888}<NL>

:SEquential:SETup**query**

Query Syntax: :SEquential:SETup?
Returned Format: :SEquential:DISPlay {ON|OFF};EXCLude <exclude_list>;INCLude
<include_list>;NPOints <points>;NSEgments <segment>;SNUMber <number>;SOURce
{CHANne1|2|3|4}<NL>
Where: <points> ::= integer, 4 to 8000
<segment> ::= integer, 1 to 8888
<number> ::= integer, 1 to 8888
<include_list> ::= integer list of included segment numbers separated by
commas
<exclude_list> ::= integer list of excluded segment numbers separated by
commas

:SEquential:SNUMber**command/query**

Command Syntax: :SEquential:SNUMber {1 to 8888}
Query Syntax: :SEquential:SNUMber?
Returned Format: [:SEquential:SNUMber] {integer, 1 to 8888}<NL>

:SEquential:SOURce**command/query**

Command Syntax: :SEquential:SOURce {CHANne1{1 | 2 | 3 | 4}}
Query Syntax: :SEquential:SOURce?
Returned Format: [:SEquential:SOURce] {CHANne1{1 | 2 | 3 | 4}}

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:SERial**(Serial Number)****command**

Command Syntax: :SERial <string>
Where: <string> ::= 10 character alphanumeric serial number within quotes

:STATus**query**

Query Syntax: :STATus? {CHANnel{1 | 2 | 3 | 4} | FUNCTION{1 | 2 | 3 | 4} | WMemory{1 | 2 | 3 | 4} | PMemory{1 | 2}}

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

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:STOP**command**

Command Syntax: :STOP

:STORe**command**

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

Where: <source> ::= {CHANnel{1 | 2 | 3 | 4} | FUNCTION{1 | 2 | 3 | 4} | WMemory{1 | 2 | 3 | 4}}

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

NOTE: CHANnel3 and CHANnel4 available only on HP 54506B/54512B

:SYSTEM:DATE**command/query**

Command Syntax: :SYSTEM:DATE <year>,<month>,<day>

Query Syntax: :SYSTEM:DATE?

Returned Format: [:SYSTEM:DATE] "DD,MM,YYYY"<NL>

:SYSTEM:DSP**command/query**

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

Query Syntax: :SYSTEM:DSP?

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

Where: <string> ::= last information written on the advisory line

:SYSTEM:ERRor**query**

Query Syntax: :SYSTEM:ERRor {NUMBER | STRING | (no_parameter)}

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

| | | |
|-------------------------|---|----------------------|
| :SYSTem:HEADer | | command/query |
| Command Syntax: | :SYSTem:HEADer {{ON 1} {OFF 0}} | |
| Query Syntax: | :SYSTem:HEADer? | |
| Returned Format: | [:SYSTem:HEADer] {1 0}<NL> | |
| :SYSTem:KEY | | command/query |
| Command Syntax: | :SYSTem:KEY {1 to 44} | |
| Query Syntax: | :SYSTem:KEY? | |
| Returned Format: | [:SYSTem:KEY] {integer, 0 to 44}<NL> | |
| :SYSTem:LONGform | | command/query |
| Command Syntax: | :SYSTem:LONGform {{ON 1} {OFF 0}} | |
| Query Syntax: | :SYSTem:LONGform? | |
| Returned Format: | [:SYSTem:LONGform] {1 0}<NL> | |
| :SYSTem:PIMacro | | command |
| Command Syntax: | :SYSTem:PIMacro <quoted ascii string> | |
| :SYSTem:SETup | | command/query |
| Command Syntax: | :SYSTem:SETup #800001703<setup data string> | |
| Query Syntax: | :SYSTem:SETup? | |
| Returned Format: | [:SYSTem:SETup] #800001703<setup data string><NL> | |
| :SYSTem:TIME | | command/query |
| Command Syntax: | :SYSTem:TIME <hour>, <minute>, <second> | |
| Query Syntax: | :SYSTem:TIME? | |
| Returned Format: | [:SYSTem:TIME] "HH:MM:SS"<NL> | |
| :TER | (Trigger Event Register) | query |
| Query Syntax: | :TER? | |
| Returned Format: | [:TER] {1 0}<NL> | |

:TIMebase:DELAy **command/query**

Command Syntax: :TIMebase:DELAy <delay time>
Query Syntax: :TIMebase:DELAy?
Returned Format: [:TIMebase:DELAy] <delay time><NL>

:TIMebase:MODE **command/query**

Command Syntax: :TIMebase:MODE {AUTO | TRIGgered | SINGle}
Query Syntax: :TIMebase:MODE?
Returned Format: [:TIMebase:MODE] {AUTO | TRIGgered | SINGle}<NL>

:TIMebase:RANGe **command/query**

Command Syntax: :TIMebase:RANGe {10 ns to 50 s}
Query Syntax: :TIMebase:RANGe?
Returned Format: [:TIMebase:RANGe] {exponential, 10 ns to 50 s}<NL>

:TIMebase:REFerence **command/query**

Command Syntax: :TIMebase:REFerence {LEFT | CENTer | RIGHT}
Query Syntax: :TIMebase:REFerence?
Returned Format: [:TIMebase:REFerence] {LEFT | CENTer | RIGHT}<NL>

:TIMebase:SAMPle **command/query**

Command Syntax: :TIMebase:SAMPle {REALtime | REPetitive}
Query Syntax: :TIMebase:SAMPle?
Returned Format: [:TIMebase:SAMPle] {REALtime | REPetitive}<NL>

:TIMebase:SETup **query**

Query Syntax: :TIMebase:SETup?
Returned Format: :TIMebase:MODE {AUTO|TRIG|SING};RANGe <range>;DELAy <delay>;REF
{LEFT|CENT|RIGH};SAMPle {REAL|REP}<NL>
Where: <range> ::= exponential, 10ns to 50s
<delay> ::= time from trigger to display reference in seconds

| | |
|--|----------------------|
| :TRIGger:CENTERed | command |
| Command Syntax: :TRIGger:CENTERed | |
| :TRIGger:CONDition | command/query |
| Command Syntax: :TRIGger:CONDition <argument> | |
| Query Syntax: :TRIGger:CONDition? | |
| Returned Format: [:TRIGger:CONDition] <argument><NL> | |
| Where: <argument> ::= {ENTer EXIT GT,<value> LT,<value> RANGE,<range_greater_than>,<range_less_than> TRUE FALSE} | |
| :TRIGger:COUPling | command/query |
| Command Syntax: :TRIGger:COUPling {DC DCFifty} | |
| Query Syntax: :TRIGger:COUPling? | |
| Returned Format: [:TRIGger:COUPling] {DC DCFifty}<NL> | |
| :TRIGger:DELay | command/query |
| Command Syntax: :TRIGger:DELay {TIME,<time_value> EVENT,<event_value>} | |
| Query Syntax: :TRIGger:DELay? | |
| Returned Format: [:TRIGger:DELay] {TIME,<time_value> EVENT,<event_value>}<NL> | |
| Where: <time_value> ::= time of delay - exponential, 30 ns to 160 ms <event_value> ::= number of events - integer, 1 to 16000000 | |
| :TRIGger:DELay:SLOPe | command/query |
| Command Syntax: :TRIGger:DELay:SLOPe {POSitive NEGative} | |
| Query Syntax: :TRIGger:DELay:SLOPe? | |
| Returned Format: [:TRIGger:DELay:SLOPe] {POSitive NEGative}<NL> | |
| :TRIGger:DELay:SOURce | command/query |
| Command Syntax: :TRIGger:DELay:SOURce {CHANne1{1 2 3 4} EXTERna1} | |
| Query Syntax: :TRIGger:DELay:SOURce? | |
| Returned Format: [:TRIGger:DELay:SOURce] {CHANne1{1 2 3 4} EXTERna1}<NL> | |
| NOTE: EXTERna1 available only on HP 54505B/54510B. CHANne13 and CHANne14 available only on HP 54506B/54512B | |

:TRIGger:FIELD **command/query**

Command Syntax: :TRIGger:FIELD { 1 | 2 }
Query Syntax: :TRIGger:FIELD?
Returned Format: [:TRIGger:FIELD] { 1 | 2 }<NL>

:TRIGger:GLITCh:CENTERed **command**

Command Syntax: :TRIGger:GLITCh:CENTERed

:TRIGger:GLITCh:HOLDoff **command/query**

Command Syntax: :TRIGger:GLITCh:HOLDoff <holdoff>
Query Syntax: :TRIGger:GLITCh:HOLDoff?
Returned Format: [:TRIGger:GLITCh:HOLDoff] <holdoff><NL>
Where: <holdoff> ::= exponential, 40 ns to 320 ms

:TRIGger:GLITCh:LEVEL **command/query**

Command Syntax: :TRIGger:GLITCh:LEVEL <level>
Query Syntax: :TRIGger:GLITCh:LEVEL?
Returned Format: [:TRIGger:GLITCh:LEVEL] <level><NL>
Where: <level> ::= trigger level in volts

:TRIGger:GLITCh:SOURce **command/query**

Command Syntax: :TRIGger:GLITCh:SOURce {CHANne1{1 | 2 | 3 | 4} | EXTERna1},{HIGH | LOW}
Query Syntax: :TRIGger:GLITCh:SOURce?
Returned Format: [:TRIGger:GLITCh:SOURce] {CHANne1{1 | 2 | 3 | 4} | EXTERna1},{HIGH | LOW}<NL>

NOTE: EXTERna1 available only on HP 54505B/54510B. CHANne13 and CHANne14 available only on HP 54506B/54512B

:TRIGger:GLITCh:WIDTH **command/query**

Command Syntax: :TRIGger:GLITCh:WIDTH {GT | LT},<width>
Query Syntax: :TRIGger:GLITCh:WIDTH?
Returned Format: [:TRIGger:GLITCh:WIDTH] {GT | LT},<width><NL>
Where: <width> ::= exponential, 5 ns to 160 ms

:TRIGger:HOLDoff **command/query**

Command Syntax: :TRIGger:HOLDoff {TIME,<holdoff_value> | EVENT,<event_argument>}
Query Syntax: :TRIGger:HOLDoff?
Returned Format: [:TRIGger:HOLDoff] {TIME,<holdoff_value> | EVENT,<event_argument>}<NL>
Where: <holdoff_value> ::= exponential, 40 ns to 320 ms
<event_argument> ::= integer, 1 to 16000000

:TRIGger:LEVEL **command/query**

Command Syntax: :TRIGger:LEVEL <level>
Query Syntax: :TRIGger:LEVEL?
Returned Format: [:TRIGger:LEVEL] <level><NL>
Where: <level> ::= trigger level in volts

:TRIGger:LINE **command/query**

Command Syntax: :TRIGger:LINE {1 to 625}
Query Syntax: :TRIGger:LINE?
Returned Format: [:TRIGger:LINE] {integer, 1 to 625}<NL>

:TRIGger:LOGic **command/query**

Command Syntax: :TRIGger:LOGic {HIGH | LOW | DONTcare}
Query Syntax: :TRIGger:LOGic?
Returned Format: [:TRIGger:LOGic] {HIGH | LOW | DONTcare}<NL>

:TRIGger:MODE **command/query**

Command Syntax: :TRIGger:MODE {EDGE | PATtern | STATE | DELay | TV | GLITCh}
Query Syntax: :TRIGger:MODE?
Returned Format: [:TRIGger:MODE] {EDGE | PATtern | STATE | DELay | TV | GLITCh}<NL>

:TRIGger:OCCurrence **command/query**

Command Syntax: :TRIGger:OCCurrence {1 to 16000000}
Query Syntax: :TRIGger:OCCurrence?
Returned Format: [:TRIGger:OCCurrence] {integer, 1 to 16000000}<NL>

:TRIGger:OCCurrence:SLOPe **command/query**

Command Syntax: :TRIGger:OCCurrence:SLOPe {POSitive | NEGative}
Query Syntax: :TRIGger:OCCurrence:SLOPe?
Returned Format: [:TRIGger:OCCurrence:SLOPe] {POSitive | NEGative}<NL>

:TRIGger:OCCurrence:SOURce **command/query**

Command Syntax: :TRIGger:OCCurrence:SOURce {CHANne1{1 | 2 | 3 | 4} | EXTerna1}
Query Syntax: :TRIGger:OCCurrence:SOURce?
Returned Format: [:TRIGger:OCCurrence:SOURce] {CHANne1{1 | 2 | 3 | 4} | EXTerna1}<NL>

NOTE: EXTerna1 available only on HP 54505B/54510B. CHANne13 and CHANne14 available only on HP 54506B/54512B

:TRIGger:PATH **command/query**

Command Syntax: :TRIGger:PATH {CHANne1{1 | 2 | 3 | 4} | EXTerna1}
Query Syntax: :TRIGger:PATH?
Returned Format: [:TRIGger:PATH] {CHANne1{1 | 2 | 3 | 4} | EXTerna1}<NL>

NOTE: EXTerna1 available only on HP 54505B/54510B. CHANne13 and CHANne14 available only on HP 54506B/54512B

:TRIGger:POLarity **command/query**

Command Syntax: :TRIGger:POLarity {POSitive | NEGative}
Query Syntax: :TRIGger:POLarity?
Returned Format: [:TRIGger:POLarity] {POSitive | NEGative}<NL>

:TRIGger:PROBe **command/query**

Command Syntax: :TRIGger:PROBe {0.9 to 1000}
Query Syntax: :TRIGger:PROBe?
Returned Format: [:TRIGger:PROBe] {exponential, 0.9 to 1000}<NL>

:TRIGger:QUALify **command/query**

Command Syntax: :TRIGger:QUALify {EDGE | PATtern | STATE | LOW | HIGH}
Query Syntax: :TRIGger:QUALify?
Returned Format: [:TRIGger:QUALify] {EDGE | PATtern | STATE | LOW | HIGH}<NL>

:TRIGger:SENSitivity**command/query**

Command Syntax: :TRIGger:SENSitivity {NORMal | LOW}
Query Syntax: :TRIGger:SENSitivity?
Returned Format: [:TRIGger:SENSitivity] {NORMal | LOW}<NL>

:TRIGger:SETup**query**

Query Syntax: :TRIGger:SETup?
Returned Format: :TRIGger:COND <argument>;COUP {DC|DCF};DEL {TIME, <time_value>|EVEN, <event_value>};DEL:SLOP {POS|NEG};DEL:SOUR <source>;FIEL {1|2};GLIT:HOLD <holdoff_value>;GLIT:LEV <level>;GLIT:SOUR <source>;GLIT:WIDT {GT|LT}, <width>;HOLD {TIME, <holdoff_value>|EVEN, <event_argument>};LEV <level>;LINE <line_number>;LOG {HIGH|LOW|DONT};MODE {EDGE|PATT|STAT|DEL|TV|GLIT};OCC <occ_value>;OCC:SLOP {POS|NEG};OCC:SOUR <source>;PATH <source>;POL {POS|NEG};PROBE <attenuation_factor>;QUAL <qualify_parameter>;SENS {NORM|LOW};SLOP {NEG|POS};SOUR <source>;STAN {525|625|USER}<NL>

Where: <argument> ::=dependent on trigger mode (refer to TRIGger:CONDition command)
<time_value> ::=exponential, 30ns to 160ms
<event_value> ::= integer, 1 to 16000000
<source> ::= CHANne11, CHANne12, EXTernal (HP 54505B/54510B)
<source> ::= CHANne11|CHANne12|CHANne13|CHANne14 (HP 54506B/54512B)
<holdoff_value> ::= exponential, 40 ns to 320 ms
<level> ::= trigger level in volts
<width> ::= exponential, 5 ns to 160 ms
<event_argument> ::= integer, 1 to 16000000
<attenuation_factor> ::= exponential, 0.9 to 1000
<line_number> ::= integer, 1 to 625
<occ_value> ::= integer, 1 to 16000000
<qualify_parameter> ::=dependent on trigger mode (refer to TRIGger:QUALify command)

:TRIGger:SLOPe**command/query**

Command Syntax: :TRIGger:SLOPe {NEGative | POSitive}
Query Syntax: :TRIGger:SLOPe?
Returned Format: [:TRIGger:SLOPe] {POSitive | NEGative}<NL>

:TRIGger:SOURce**command/query****Command Syntax:** :TRIGger:SOURce {CHANne1{1 | 2 | 3 | 4} | EXTERna1 | LINE}**Query Syntax:** :TRIGger:SOURce?**Returned Format:** [:TRIGger:SOURce] {CHANne1{1 | 2 | 3 | 4} | EXTERna1 | LINE}<NL>NOTE: EXTERna1 available only on HP 54505B/54510B. CHANne13 and CHANne14 available only on HP 54506B/54512B

:TRIGger:STANdard**command/query****Command Syntax:** :TRIGger:STANdard {525 | 625 | USER}**Query Syntax:** :TRIGger:STANdard?**Returned Format:** [:TRIGger:STANdard] {525 | 625 | USER}<NL>

:VIEW**command****Command Syntax:** :VIEW {CHANne1{1 | 2 | 3 | 4} | FUNctio{n}{1 | 2 | 3 | 4} | PMEMory{1 | 2} | WMEMory{1 | 2 | 3 | 4}}NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:WAVEform:DATA**command/query****Command Syntax:** :WAVEform:DATA <binary block data in # format>**Query Syntax:** :WAVEform:DATA?**Returned Format:** [:WAVEform:DATA] <binary block length bytes><binary block><NL>

:WAVEform:FORMat**command/query****Command Syntax:** :WAVEform:FORMat {ASCii | WORD | BYTE | COMPRESSED}**Query Syntax:** :WAVEform:FORMat?**Returned Format:** [:WAVEform:FORMat] {ASCii | WORD | BYTE | COMPRESSED}<NL>

:WAVEform:POINts**query****Query Syntax:** :WAVEform:POINts?**Returned Format:** [:WAVEform:POINts] {500 | 8000}<NL>

:WAVeform:PREamble**command/query****Command Syntax:** :WAVeform:PREamble <preamble block>**Query Syntax:** :WAVeform:PREamble?**Returned Format:** [:WAVeform:PREamble] <preamble block><NL>**Where:** <preamble block> ::= <format NR1>, <type NR1>, <points NR1>, <count NR1>, <xincrement NR3>, <xorigin NR3>, <xreference NR1>, <yincrement NR3>, <yorigin NR3>, <yreference NR1><format> ::= 0 for ASCII format
1 for BYTE format
2 for WORD format
4 for COMPRESSED format<type> ::= 0 for INVALID type
1 for NORMAL type or REALTIME
2 for AVERAGE type
3 for ENVELOPE type
4 for RAWDATA type

:WAVeform:SOURce**command/query****Command Syntax:** :WAVeform:SOURce {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}**Query Syntax:** :WAVeform:SOURce?**Returned Format:** [:WAVeform:SOURce] {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4}}<NL>

NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:WAVeform:TYPE**query****Query Syntax:** :WAVeform:TYPE?**Returned Format:** [:WAVeform:TYPE] {INValid | AVERage | ENVelope | NORMAl | RAWData}<NL>

:WAVeform:XINCrement**query****Query Syntax:** :WAVeform:XINCrement?**Returned Format:** [:WAVeform:XINCrement] <x-increment value><NL>

:WAVeform:XORigin**query****Query Syntax:** :WAVeform:XORigin?**Returned Format:** [:WAVeform:XORigin] <x-origin value>[, <x-origin value>]...<NL>

:WAVeform:XREFerence**query**

Query Syntax: :WAVeform:XREFerence?
Returned Format: [:WAVeform:XREFerence] <x-reference value><NL>

:WAVeform:YINCrement**query**

Query Syntax: :WAVeform:YINCrement?
Returned Format: [:WAVeform:YINCrement] <y-increment value><NL>

:WAVeform:YORigin**query**

Query Syntax: :WAVeform:YORigin?
Returned Format: [:WAVeform:YORigin] <y-origin value><NL>

:WAVeform:YREFerence**query**

Query Syntax: :WAVeform:YREFerence?
Returned Format: [:WAVeform:YREFerence] <y-reference value><NL>

:WMEMory{1 | 2 | 3 | 4}[:DISPlay]**command/query**

Command Syntax: :WMEMory{1 | 2 | 3 | 4}[:DISPlay] {{ON | 1} | {OFF | 0}}
Query Syntax: :WMEMory{1 | 2 | 3 | 4}[:DISPlay]?
Returned Format: [:WMEMory{1 | 2 | 3 | 4}[:DISPlay]] {1 | 0}<NL>

:WMEMory{1 | 2 | 3 | 4}:GET**command**

Command Syntax: :WMEMory{1 | 2 | 3 | 4}:GET {CHANne1{1 | 2 | 3 | 4} | WMEMory{1 | 2 | 3 | 4} | FUNCtion{1 | 2 | 3 | 4}}
NOTE: CHANne13 and CHANne14 available only on HP 54506B/54512B

:WMEMory{1 | 2 | 3 | 4}:PROTECT**command/query**

Command Syntax: :WMEMory{1 | 2 | 3 | 4}:PROTECT {{ON | 1} | {OFF | 0}}
Query Syntax: :WMEMory{1 | 2 | 3 | 4}:PROTECT?
Returned Format: [:WMEMory{1 | 2 | 3 | 4}:PROTECT] {1 | 0}<NL>

:WMEemory{1 | 2 | 3 | 4}:SETup**query**

Query Syntax: :WMEemory{1 | 2 | 3 | 4}:SETup?
Returned Format: :WMEemory{1 | 2 | 3 | 4}:DISPlay{ON | OFF};PROTeCt {ON | OFF};XRANGe
<xrange>;XOFFset <xoffset>;YRANGe <xrange>;YOFFset <xoffset><NL>
Where: <xrange> ::= exponential, 10 ns to 50 s
<xoffset> ::= exponentialtime from trigger to display reference in seconds.
Display reference is left, center, or right
<yrange> ::= exponential full-scale range value
<yoffset> ::= exponential offset value in volts

:WMEemory{1 | 2 | 3 | 4}:XOFFset**command/query**

Command Syntax: :WMEemory{1 | 2 | 3 | 4}:XOFFset <xoffset>
Query Syntax: :WMEemory{1 | 2 | 3 | 4}:XOFFset?
Returned Format: [:WMEemory{1 | 2 | 3 | 4}:XOFFset] <xoffset><NL>
Where: <xoffset> ::= exponentialtime from trigger to display reference in seconds.
Display reference is left, center, or right

:WMEemory{1 | 2 | 3 | 4}:XRANGe**command/query**

Command Syntax: :WMEemory{1 | 2 | 3 | 4}:XRANGe <xrange>
Query Syntax: :WMEemory{1 | 2 | 3 | 4}:XRANGe?
Returned Format: [:WMEemory{1 | 2 | 3 | 4}:XRANGe] <xrange><NL>
Where: <xrange> ::= exponential, 10 ns to 50 s

:WMEemory{1 | 2 | 3 | 4}:YOFFset**query**

Query Syntax: :WMEemory{1 | 2 | 3 | 4}:YOFFset?
Returned Format: [:WMEemory{1 | 2 | 3 | 4}:YOFFset] <yoffset><NL>
Where: <yoffset> ::= exponential offset value in volts

:WMEemory{1 | 2 | 3 | 4}:YRANGe**query**

Query Syntax: :WMEemory{1 | 2 | 3 | 4}:YRANGe?
Returned Format: [:WMEemory{1 | 2 | 3 | 4}:YRANGe] <yrange><NL>
Where: <yrange> ::= exponential full-scale range value

Index

A

- AC RMS, 15-52, 15-64
- Acquire Subsystem, 9-1
 - COMPLet Command/Query, 9-6
 - COUNt Command/Query, 9-7
 - POINtS Command/Query, 9-9
 - Syntax Diagram, 9-1
 - TYPE Command/Query, 9-11
- ADD Command, 13-4
- add to memory, 7-15
- Addressing, 3-2
- Addressing the instrument, 3-2
- adjust trigger mode, 17-15
- advisory line, 8-4
- Algorithms, 26-1
- ALL Query, 15-13
- alpha argument, 4-1
- arbitrary ASCII response data, 27-23
- arbitrary block program data, 27-14
- Arguments, 1-5
- ASCII format, 18-9
- AUTO time base mode, 16-4
- automatic measurements, 26-1
- Automatic Top-Base, 26-2
- Autoscale
 - undo, 6-18
- AUToscale Command, 7-5
- AVERAGE Type, 18-4
- Averaging Mode, 9-4

B

- BASIC, 1-2
- BEEPer Command/Query, 7-6
- BLANk Command, 7-7, 7-24
- Block data, 1-4, 2-11, 27-23
- BNC Command/Query, 7-8
- Buffer Deadlock, 27-5
- BYTE format, 18-9

C

- Calibrate Subsystem, 10-1
 - DATA:ASCii Query, 10-2
 - SETup Query, 10-3
 - Syntax Diagram, 10-1
 - TNULl Command/Query, 10-3
- carriage return, 4-14
- CENTERed Command, 17-15
- Channel Subsystem, 11-1
 - COUPling Command/Query, 11-4
 - DISPlay Command/Query, 11-5
 - ECL Command, 11-6
 - HFReject Command/Query, 11-7
 - LFReject Command/Query, 11-8
 - OFFSet Command/Query, 11-9
 - PROBe Command/Query, 11-10
 - RANGE Command/Query, 11-11
 - SETup Query, 11-12
 - Syntax Diagram, 11-2
 - TTL Command, 11-13
- Character data, 1-10, 2-8

Character program data, 1-10, 2-8, 27-14
 character response data, 27-23
 CLEAR DISPLAY, 7-11
 Clear measurement, 15-40
 *CLS (clear status) Command, 6-4
 CME - command error, 24-3
 COLumn Command/Query, 12-5
 Combining commands, 1-7
 comma, 27-6
 Command, 1-3, 2-7
 Acquire Subsystem, 9-1
 ACQUIRE:COMPLETE, 9-6
 ACQUIRE:COUNT, 9-7
 ACQUIRE:POINTS, 9-8
 ACQUIRE:TYPE, 9-10
 AUTOScale, 7-5
 BEEPER, 7-6
 BLANK, 7-7
 BNC, 7-8
 Calibrate Subsystem, 10-1
 CALIBRATE:TNULL, 10-4
 Channel Subsystem, 11-1
 CHANNELN:COUPLING, 11-4
 CHANNELN:ECL, 11-6
 CHANNELN:HFREJECT, 11-7
 CHANNELN:LFRREJECT, 11-8
 CHANNELN:OFFSET, 11-9
 CHANNELN:PROBE, 11-10
 CHANNELN:RANGE, 11-11
 CHANNELN:SETUP, 11-12
 CHANNELN:TTL, 11-13
 *CLS, 6-3 / 6-4
 Common Commands, 4-10, 6-1
 DATA:ASCII, 10-2
 DESTINATION, 15-20
 DIGITize, 2-4, 7-9
 Display Subsystem, 12-1
 DISPLAY:COLUMN, 12-5
 DISPLAY:CONNECT, 12-6
 DISPLAY:DATA, 12-7
 DISPLAY:FORMAT, 12-9

Command (continued)
 DISPLAY:GRATICULE, 12-10
 DISPLAY:INVERSE, 12-11
 DISPLAY:LINE, 12-12
 DISPLAY:MASK, 12-13
 DISPLAY:MARKER, 12-23
 DISPLAY:PERSISTENCE, 12-15
 DISPLAY:ROW, 12-17
 DISPLAY:SCREEN, 12-18
 DISPLAY:SETUP, 12-19
 DISPLAY:SOURCE, 12-20
 DISPLAY:STRING, 12-21
 DISPLAY:TEXT, 12-22
 DISPLAY:TMARKER, 12-23
 DISPLAY:VMARKER, 12-23
 *DMC, 6-5
 *EMC, 6-6
 EOI, 4-28
 ERASE, 7-11
 *ESE, 6-7
 *ESR, 6-9
 Function Subsystem, 13-1
 FUNCTIONN:ADD, 13-5
 FUNCTIONN:DIFF, 13-6
 FUNCTIONN:DISPLAY, 13-7
 FUNCTIONN:FFT, 13-8
 FUNCTIONN:FREQUENCY, 13-9
 FUNCTIONN:INTEGRATE, 13-10
 FUNCTIONN:INVERT, 13-11
 FUNCTIONN:LEVEL, 13-12
 FUNCTIONN:MAGNIFY, 13-13
 FUNCTIONN:MULTIPLY, 13-14
 FUNCTIONN:OFFSET, 13-15
 FUNCTIONN:ONLY, 13-16
 FUNCTIONN:PEAK, 13-17
 FUNCTIONN:POINTS, 13-18
 FUNCTIONN:RANGE, 13-21
 FUNCTIONN:SETUP, 13-22
 FUNCTIONN:SPAN, 13-23
 FUNCTIONN:SUBTRACT, 13-24
 FUNCTIONN:VERSUS, 13-25

Command (continued)

FUNcTION:WINDow, 13-26
Hardcopy Subsystem, 14-1
HARDcopy:LENGth, 14-4
HARDcopy:MODE, 14-5
HARDcopy:PAGE, 14-6
HARDcopy:PLOT:AREA, 14-7
HARDcopy:PLOT:INITialize, 14-8
HARDcopy:PLOT:PEN|COLor, 14-9
HEADer, 2-7, 8-8
*IDN, 6-12
*IST, 4-28
LER, 7-12
*LMC, 6-13
LONGform, 2-7, 8-11
*LRN, 6-14
LTER, 7-13
Measure Subsystem, 15-1
MEASure:ALL, 15-13
MEASure:COMParE, 15-14
MEASure:CURSor, 15-16
MEASure:DEFine, 15-17
MEASure:DELay, 15-19
MEASure:DESTination, 15-20
MEASure:DUTYcycle, 15-22
MEASure:ESTArt, 15-23
MEASure:ESTOp, 15-25
MEASure:FALLtime, 15-27
MEASure:FREQuency, 15-28
MEASure:LIMittest, 15-29
MEASure:LOWer, 15-30
MEASure:MODE, 15-31
MEASure:NWIDth, 15-32
MEASure:OVERshoot, 15-33
MEASure:PERiod, 15-34
MEASure:POSTfailure, 15-35
MEASure:PREShoot, 15-36
MEASure:PWIDth, 15-37
MEASure:RESults, 15-38
MEASure:RISetime, 15-39
MEASure:SCRatch, 15-40

Command (continued)

MEASure:SOURce, 15-41
MEASure:STATistics, 15-42
MEASure:TDELta, 15-43
MEASure:TMAX, 15-44
MEASure:TMIN, 15-45
MEASure:TSTArt, 15-46
MEASure:TSTOp, 15-47
MEASure:TVOLt, 15-48
MEASure:UNITs, 15-50
MEASure:UPPer, 15-51
MEASure:VACRms, 15-52
MEASure:VAMPlitude, 15-53
MEASure:VAVerage, 15-54
MEASure:VBASe, 15-55
MEASure:VDCRms, 15-56
MEASure:VDELta, 15-57
MEASure:VFIFTy, 15-58
MEASure:VMAX, 15-59
MEASure:VMIN, 15-60
MEASure:VPP, 15-61
MEASure:VRELative, 15-62
MEASure:VRMS, 15-64
MEASure:VSTArt, 15-65
MEASure:VSTOp, 15-66
MEASure:VTIME, 15-67
MEASure:VTOP, 15-68
MEASure:WCOMpare, 15-69
MEASure:WCOMpare:ALLowance, 15-70
MEASure:WCOMpare:COMParE, 15-71
MEASure:WCOMpare:DESTination, 15-72
MEASure:POSTfailure, 15-74
MEASure:WCOMpare:WTEST, 15-75
MENU, 7-14
MERGe, 7-15
*OPC, 6-15
*OPT, 6-16
parsing, 27-1
PLOT, 7-16
*PMC, 6-17
PRINt, 7-17

Command (continued)

*RCL, 6-18
 Root Level Commands, 4-10, 7-1
 *RST, 6-19
 RUN, 7-18
 *SAV, 6-22
 SERial, 7-19
 *SRE, 6-23
 STATus, 7-20
 *STB, 6-25
 STOP, 7-21
 STORe, 7-22
 Subsystem Commands, 4-10
 System Subsystem, 8-1
 SYSTem:DATE, 8-3
 SYSTem:DSP, 8-4
 SYSTem:ERRor, 8-5
 SYSTem:HEADer, 8-8
 SYSTem:KEY, 8-9
 SYSTem:LONGform, 8-11
 SYSTem:PIMacro, 8-12
 SYSTem:SETup, 8-13
 SYSTem:TIME, 8-15
 TER, 7-23
 Timebase Subsystem, 16-1
 TIMEbase:DELay, 16-3
 TIMEbase:MODE, 16-4
 TIMEbase:RANGe, 16-5
 TIMEbase:REFerence, 16-6
 TIMEbase:SAMPle, 16-7
 *TRG, 6-27
 Trigger Mode, 17-6
 Trigger Subsystem, 17-1
 TRIGger:CENTerEd, 17-14
 TRIGGER:CONDITiON, 17-9, 17-10, 17-16
 TRIGger:COUPLing, 17-18
 TRIGGER:DELAY, 17-11, 17-19
 TRIGger:DELay:SLOPe, 17-20
 TRIGger:DELay:SOURce, 17-21
 TRIGGER:FIELD, 17-12, 17-22
 TRIGger:GLITch, 17-23

Command (continued)

TRIGger:GLITch:CENTerEd, 17-24
 TRIGger:GLITch:HOLDoff, 17-25
 TRIGger:GLITch:LEVel, 17-26
 TRIGger:GLITch:SOURce, 17-27
 TRIGger:GLITch:WIDTh, 17-28
 TRIGGER:HOLDOFF, 17-9, 17-26
 TRIGGER:LEVEL, 17-6, 17-8 / 17-9, 17-30
 TRIGger:LINE, 17-31
 TRIGGER:LOGIC, 17-10, 17-32
 TRIGger:MODE, 17-33
 TRIGGER:OCCURRENCE, 17-14, 17-13, 17-34
 TRIGGER:OCCURRENCE:SLOPE, 17-11,
 17-14, 17-35
 TRIGGER:OCCURRENCE:SOURCE, 17-11,
 17-36
 TRIGGER:PATH, 17-9, 17-10, 17-37
 TRIGGER:POLARITY, 17-12, 17-38
 TRIGger:PROBe, 17-39
 TRIGGER:QUALIFY, 17-11, 17-14, 17-40
 TRIGger:SENSitivity, 17-41
 TRIGger:SETup, 17-42
 TRIGGER:SLOPE, 17-8, 17-43
 TRIGGER:SOURCE, 17-8, 17-12, 17-44
 TRIGGER:STANDARD, 17-12, 17-45
 TRIGGER:TIME, 17-11
 *TST, 6-28
 VIEW, 7-24
 *WAI, 6-29
 Waveform Subsystem, 18-1
 WAVeform:COUNt, 4-24
 WAVeform:DATA, 18-11
 WAVeform:FORMat, 18-13
 WAVeform:POINts, 18-14
 WAVeform:PREamble, 18-15
 WAVeform:SOURce, 18-17
 WAVeform:TYPE, 18-18
 WAVeform:XINCrement, 18-19
 WAVeform:XORIGin, 18-20
 WAVeform:XREFerence, 18-21
 WAVeform:YINCrement, 18-22

- Command (continued)
 - WAVeform:YORigin, 18-23
 - WAVeform:YREfERENCE, 18-24
- Command and Data Concepts, 3-1
- command error, 27-4
- Command structure, 2-3
- command tree, 4-1, 4-10
- Command Types, 4-10
- Common command header, 1-6
- common commands, 4-10, 6-1, 27-27
 - *CLS Command, 6-4
 - *DMCCommand, 6-5
 - *EMC Command/Query, 6-6
 - *ESE Command/Query, 6-7
 - *ESR Query, 6-9
 - *GMC Query, 6-11
 - *IDN Query, 6-12
 - *LMC Query, 6-13
 - *LRN Query, 6-14
 - *OPC Command/Query, 6-15
 - *OPT Query, 6-16
 - *PMC Command, 6-17
 - *RCL Command, 6-18
 - *RST Command, 6-19
 - *SAV Command, 6-22
 - *SRE Command/Query, 6-23
 - *STB , 6-25
- Syntax Diagram, 6-1
- *TRG Command, 6-27
- *TST, 6-28
- *WAI Command, 6-29
- Communication, 1-2
- COMPare Command/Query, 15-14
- COMPLete Command/Query, 9-6
- Compound command header, 1-6
- compound header, 4-14
- compound query, 27-3
- COMPRESSED format, 18-9
- CONDition Command/Query, 17-16
- CONNect Command/Query, 12-6
- Controllers, 1-2

- COUNt Command/Query, 9-7
- COUPLing Command/Query, 11-4, 17-18
- Cross-Reference
 - Front Panel to Command, 4-2
- CURSor Query, 15-16

D

- Data Acquisition Types, 18-4
- DATA Command/Query, 12-7, 18-11
- DATA:ASCii Query, 10-2
- DC RMS, 15-56
- DDE - device specific error, 24-3
- deadlock, 27-5
- decimal numeric program data, 27-14
- DEFine Command/Query, 15-17
- definite length arbitrary block response data, 27-23
- Definite-length block response data, 2-11
- DELay Command/Query, 15-19, 16-3, 17-19
- Delay Trigger Mode, 17-11, 17-15
- DELay:SLOPe Command/Query, 17-20

- DELay:SOURce Command/Query, 17-21
- DESTination Command/Query, 15-20
- Device address, 1-4
 - HP-IB, 3-3
- device clear command, 27-3
- Device Listening Syntax, 27-8
- Device Talking Syntax, 27-19
- device-specific error, 27-4
- DIFF Command, 13-6
- differentiate, 13-6
- Digitize, 5-11
- DIGitize Command, 2-4, 7-9, 9-1, 18-4
- display mode, 9-3

Display Subsystem, 12-1
 COLUmN Command/Query, 12-5
 CONNect Command/Query, 12-6
 DATA Command/Query, 12-7
 FORMat Command/Query, 12-9
 GRATicule Command/Query, 12-10
 INVerse Command/Query, 12-11
 LINE Command, 12-12
 MARKer Command/Query, 12-23
 MASK Command/Query, 12-13
 PERsistence Command/Query, 12-15
 ROW Command/Query, 12-17
 SCREen Command/Query, 12-18
 SETup Query, 12-19
 SOURce Command/Query, 12-20
 STRing command, 12-21
 Syntax Diagram, 12-2
 TEXT Command, 12-22
 TMARker Command/Query, 12-23
 VMARker Command/Query, 12-23
 DISPlay:SCREen, 12-18
 DSP Command/Query, 8-4
 Duplicate mnemonics, 1-7
 duty cycle, 26-5
 DUTYcycle Command/Query, 15-22

E

ECL
 undo, 6-18
 ECL Command, 11-6
 Edge Definition, 26-2
 Embedded strings, 1-2, 1-4, 1-11
 Enter statement, 1-2
 Entered, 17-9
 Envelope Mode, 9-5
 ENVELOPE Type, 18-4
 EOI, 1-12
 ERASe Command, 7-11
 ERASE PMEMORY0, 7-11

error number, 8-5
 ERRor query, 8-5
 error queue, 6-4
 ESB - event status bit, 24-3
 *ESE Command
 Command/Query, 6-7
 *ESR Query, 6-9
 ESR, event summary, 6-9
 ESTArt Command/Query, 15-23
 ESTOp Command/Query, 15-25
 Event Status Register, 6-7
 Example Program, 2-3, 5-1
 Digitize, 5-6, 5-11, 5-21
 Initialize, 5-4, 5-9, 5-16, 5-19, 5-26
 EXE - execution error, 24-3
 execution error, 27-4
 Exited, 17-9
 exponential - NR3, 27-23
 Exponents, 1-11

F

fall time, 26-5
 fall time measurement, 15-11
 fall time measurements, 26-1
 FALLtime Command/Query, 15-27
 FIELd Command/Query, 17-22
 FORMat Command/Query, 12-9, 18-13
 Fractional values, 1-11
 frequency, 26-5
 FREQuency Command/Query, 15-28
 frequency measurement, 15-11
 Front Panel to Command
 Cross-Reference, 4-2
 Function Subsystem, 13-1
 ADD Command, 13-5
 DIFF Command, 13-6
 DISPlay Command/Query, 13-7
 FFT Command, 13-8
 FREQuency Command/Query, 13-9

Function Subsystem (continued)

- INTEgrate Command, 13-10
 - INVert Command, 13-11
 - LEVel Command/Query, 13-12
 - MAGNify Command/Query, 13-13
 - MULTiply Command, 13-14
 - OFFSet Command/Query, 13-15
 - ONLY Command, 13-16
 - PEAK Command, 13-17
 - POINts Command/Query, 13-18
 - RANGE Command/Query, 13-21
 - SETup Query, 13-22
 - SPAN Command/Query, 13-23
 - SUBTract Command, 13-24
 - Syntax Diagram, 13-2
 - VERSus Command, 13-25
- ## Functional Elements
- Input Buffer, 27-1
 - Output Queue, 27-2
 - Parser, 27-2

G

- GET command, 27-3
- GLITch Subsystem, 17-23
 - GLITch:CENTERed, 17-24
 - GLITch:HOLDoff, 17-25
 - GLITch:LEVel, 17-26
 - GLITch:SOURce, 17-27
 - GLITch:WIDTh, 17-28
- Glitch Trigger, 17-14
- *GMC Query, 6-11
- GRATicule Command/Query, 12-10
- group execute trigger (GET), 27-3
- GT (Greater Than), 17-9

**HP 54505B/54506B/54510B/54512B
Programming Reference**

H

- Hardcopy Plot, 7-16
- Hardcopy Print, 7-17
- Hardcopy Subsystem, 14-1
 - LENGth Command/Query, 14-4
 - MODE Command/Query, 14-5
 - PAGE Command/Query, 14-6
 - PLOT:AREA Command/Query, 14-7
 - PLOT:INITialize Command/Query, 14-8
 - PLOT:PEN|COLor Command/Query, 14-9
 - Syntax Diagram, 14-1
- HEADer command, 2-7
- HEADer Command/Query, 8-7
- Headers, 1-4, 1-5, 4-1
- HFReject Command/Query, 11-7
- HOLDoff Command/Query, 17-29
- Host language, 1-4
- HP-IB, 3-2

I

- *IDN Query, 6-12, 7-19
- IEEE 488.1, 27-1
- IEEE 488.2, 4-10, 27-5
 - Standard, 1-1, 27-1
 - Syntax, 27-5
- Infinity Representation, 4-16
- Initialization, 2-1
- input
 - vertical, 7-5
- Input Buffer, 27-1, 27-3, 27-5
- Instruction headers, 1-4
- Instruction syntax, 1-4
- Instructions, 1-4
- instrument
 - serial number, 6-12
- Instrument address
 - HP-IB, 3-3

Index-7

integer - NR1, 27-23
integrate, 13-10
INTEgrate Command, 13-10
Interface Capabilities, 3-1
Interface select code
 HP-IB, 3-2
Interrupted Condition, 27-4
interrupted error, 27-26
INVerse Command/Query, 12-11
INVert Command, 13-11

K

key codes, 8-9
KEY Command/Query, 8-9

L

LCL - local, 24-3
leading colon, 4-14
 X1Position, 23-6
learn string, 6-14
LENGth Command/Query, 14-4
LER Query, 7-12
level, trigger, 7-5
LEVel Command/Query, 17-30
LFReject Command/Query, 11-8
Limit Test Event Register, 7-13
LIMittest Command, 15-29
LINE Command, 12-12
LINE Command/Query, 17-31
linefeed, 4-14
linefeed (CRLF), 4-12
*LMC Query, 6-13
Local, 3-3, 8-9
LOGic Command/Query, 17-32
Long form, 1-9
LONGform command, 2-7

LONGform Command/Query, 8-11
LOWer Command/Query, 15-30
Lowercase, 1-9
*LRN Query, 6-14
LT (Less Than), 17-9
LTER Query, 7-13
LTF - limit test failure, 24-3

M

Making Measurements, 15-11, 26-1
Marker Subsystem, 23-1
 DISPlay Command/Query, 23-3
 SETup Query, 23-4
 Syntax Diagram, 23-1
 XDELta Query, 23-5
 X1Position Command/Query, 23-6
 X2Position Command/Query, 23-7
 X1Y1source Command/Query, 23-8
 X2Y2source Command/Query, 23-9
 YDELta Query, 23-10
 Y1Position Command/Query, 23-11
 Y2Position Command/Query, 23-12
MASK Command/Query, 12-13
MAV - message available, 24-3
Measure Subsystem, 15-1
 ALL Query, 15-13
 COMPare Command/Query, 15-14
 CURSor Query, 15-16
 DEFine Command/Query, 15-17
 DELay Command/Query, 15-19
 DESTination Command/Query, 15-20
 DUTYcycle Command/Query, 15-22
 ESTArt Command/Query, 15-23
 ESTOp Command/Query, 15-25
 FALLtime Command/Query, 15-27
 FREQuency Command/Query, 15-28
 LIMittest Command, 15-29
 LOWer Command/Query, 15-30
 MODE Command/Query, 15-31

Measure Subsystem (continued)

- NWIDth Command/Query, 15-32
- OVERshoot Command/Query, 15-33
- PERiod Command/Query, 15-34
- POSTfailure Command/Query, 15-35
- PRECision, nonimplemented, 4-28
- PREShoot Command/Query, 15-36
- PWIDth Command/Query, 15-37
- RESults Query, 15-38
- RISetime Command/Query, 15-39
- SCRatch Command, 15-40
- SOURce Command/Query, 15-41
- STATistics Command/Query, 15-42
- Syntax Diagram, 15-2
- TDELta Query, 15-43
- TMAX Query, 15-44
- TMIN Query, 15-45
- TSTArt Command/Query, 15-46
- TSTOp Command/Query, 15-47
- TVOLt Query, 15-48
- UNITs Command/Query, 15-50
- UPPer Command/Query, 15-51
- VACRms Command/Query, 15-52
- VAMPLitude Command/Query, 15-53
- VAVerage Command/Query, 15-54
- VBASe Command/Query, 15-55
- VDCRms Command/Query, 15-56
- VDELta Query, 15-57
- VFIFTy Command, 15-58
- VMAX Command/Query, 15-59
- VMIN Command/Query, 15-60
- VPP Command/Query, 15-61
- VRELative Command/Query, 15-62
- VRMS Command/Query, 15-64
- VSTArt Command/Query, 15-65
- VSTOp Command/Query, 15-66
- VTIME Query, 15-67
- VTOP Command/Query, 15-68

Measure Subsystem (continued)

- WCOMpare, 15-69
- WCOMpare:ALLowance, 15-70
- WCOMpare:COMpare, 15-71
- WCOMpare:DESTination, 15-72
- WCOMpare:POSTfailure, 15-74
- WCOMpare:WTEST, 15-75
- Measurement Error, 15-10
- Measurement Setup, 5-7, 15-10, 26-1
- MENU Command/Query, 7-14
- MERGe Command, 7-15
- message exchange protocols, 27-1
 - width, 26-4
- MMEMory Subsystem, 21-1
- mnemonic, 4-1
- Mode
 - Delay Trigger, 17-11
 - Glitch Trigger, 17-14
 - Pattern Trigger, 17-9
- Mode (continued)
 - State Trigger, 17-10
 - TV Trigger, 17-12
- MODE Command/Query, 14-5, 15-31, 16-4, 17-33
- MSG - message, 24-3
- MSS (Master Summary Status), 6-25
- MSS - master summary status, 24-3
- Multiple numeric variables, 2-12
- Multiple Memory Subsystem, 21-1
 - DISPlay Command/Query, 21-2
 - FNUMber Command/Query, 21-3
 - SOURce Command/Query, 21-4
 - STORe Command, 21-5
 - Syntax Diagram, 21-1
- Multiple program commands, 1-12
- Multiple program data, 1-10
- Multiple queries, 2-12, 27-3
- Multiple subsystems, 1-12
- MULTiply Command, 13-14

N

- width, 26-4
- NL, 1-12, 4-10, 4-14, 27-6
- Noise Reject, 17-41
- Nonimplemented Command, 4-16
- Normal Persistence mode, 9-3
- NORMAL Type, 18-4
- Notation Conventions and Definitions, 4-17
- nr1 numeric response data, 27-23
- nr3 numeric response data, 27-23
- number of averages, 9-3
- Numeric data, 1-11
- Numeric program data, 1-11
- Numeric variables, 2-10
- NWIDTH Command/Query, 15-32

O

- OCCurrence Command/Query, 17-34
- OCCurrence:SLOPe Command/Query, 17-35
- OCCurrence:Source Command/Query, 17-36
- offset
 - vertical, 7-5
- OFFSet Command/Query, 11-9, 13-15
- ONLY Command, 13-16
- *OPC, 6-15
- OPC - operation complete, 24-3
- Operation Complete, 24-4
- *OPT Query, 6-16
- Order of commands, 5-2
- Output command, 1-3
- output queue, 6-15, 27-2, 27-5
- OUTPUT statement, 1-2
- Overlapped Commands, 4-16
- OVERshoot Command/Query, 15-33

P

- PAGE Command/Query, 14-6
- Parallel Poll, 24-5
- Parameters, 1-5
- Parser, 2-1, 27-2, 27-9
- parsing, 27-1 / 27-2
- PATH Command/Query, 17-37
- Pattern Trigger Mode, 17-9, 17-16
- period, 26-5
- PERiod Command/Query, 15-34
- period measurement, 15-11
- PERSistence Command/Query, 12-15
- Pixel Memory Subsystem, 20-1
 - CLEAr Command, 20-2
 - DISPlay Command/Query, 20-3
 - MERGe Command, 20-4
 - SETup Query, 20-5
 - Syntax Diagram, 20-1
- PLOT Query, 7-16
- PLOT:AREA Command/Query, 14-7
- PLOT:INITialize Command/Query, 14-8
- PLOT:PEN|COLor Command/Query, 14-9
 - + width, 26-4
- PMEMemory Subsystem, 20-1
- POINts Command/Query, 9-8
- POINts Query, 18-14
- POLarity Command/Query, 17-38
- PON - power on, 24-3
- POSTfailure Command/Query, 15-35
- PREamble Command/Query, 18-15
- PREShoot Command/Query, 15-36
- PRINt Query, 7-17
- PROBe Command/Query, 11-10, 17-39
- Program data, 1-5, 1-10, 27-14
- Program example, 2-3, 5-1
 - Digitize, 5-6, 5-11, 5-21
 - Initialize, 5-4, 5-9, 5-16, 5-19, 5-26
- program message, 4-14, 27-2 / 27-3, 27-6, 27-9
- Program message syntax, 1-3

Program message terminator, 1-12, 4-12, 27-2
program message unit, 27-6, 27-10
program message unit separator, 4-12, 27-11
Program syntax, 1-3
Protocols, 27-1
pulse width measurement, 15-11
PWIDth Command/Query, 15-37

Q

QUALify Command/Query, 17-40
Queries
 Nonimplemented, 4-28
Query, 1-4, 1-8, 2-7
Query command, 1-8
query error, 27-3 / 27-4
query message, 27-2 / 27-3
query message unit, 27-21
query program header, 27-11
Query response, 2-6
query responses, 4-16
Question mark, 1-8
QYE - query error, 24-3

R

RANGE, 17-9
RANGe Command/Query, 11-11, 13-21, 16-5
Rawdata Mode, 9-5, 18-7, 18-15
RAWDATA Type, 18-4
*RCL Command, 6-18
Realtime, 16-7
Recall
 undo, 6-18
RECOrd, nonimplemented, 4-28
REFerence Command/Query, 16-6
Repetitive, 16-7
Response data, 2-11, 27-23

response data separator, 27-25
Response Generation, 4-16
Response header, 27-21
response header separator, 27-26
response message, 27-2 / 27-3, 27-21
response message terminator, 27-26
response message unit, 27-21
response message unit separator, 27-26
response messages, 27-2
RESults Query, 15-38
rise time, 26-5
rise time measurement, 15-11
rise time measurements, 26-1
RISetime Command/Query, 15-39
Root Level commands, 4-10, 7-1
 AUToscale Command, 7-5
 BEEPer Command/Query, 7-6
 BLANk Command, 7-7
 BNC Command/Query, 7-8
 DIGitize Command, 7-9
 ERASe Command, 7-11
 LER Query, 7-12
 LTER Query, 7-13
 MENU Command/Query, 7-14
 MERGe Command, 7-15
 PLOT Query, 7-16
 PRINt Query, 7-17
 RUN Command, 7-18
 SERial Command, 7-19
 STATus Query, 7-20
 STOP Command, 7-21
 STORe Command, 7-22
 Syntax Diagram, 7-1
 TER Query, 7-23
 VIEW Command, 7-24
ROW Command/Query, 12-17
ROC - request control, 24-3
RQS - request service, 24-3
*RST Command, 6-19
RUN Command, 7-18

S

- SAMPlE Command/Query, 16-7
- *SAV Command, 6-22
- save register, 6-22
- save/recall register, 6-18
- SCRatch Command, 15-40
- SCREen Command/Query, 12-18
- seconds/division, 16-5
- semicolon, 27-6
- sensitivity
 - vertical, 7-5
- SENSitivity Command/Query, 17-41
- Separator, 1-5
- Sequential commands, 4-16
- Sequential single-shot, 22-1
- Sequential Subsystem, 22-1
 - DISPlay Command/Query, 22-3
 - EXCLude Command/Query, 22-4
 - INCLude Command/Query, 22-6
 - NPOints Command/Query, 22-8
 - NSEGMents Command/Query, 22-9
 - SETup Query, 22-11
 - SNUMber Command/Query, 22-12
 - SOURce Command/Query, 22-13
 - Syntax Diagram, 22-1
- SERial Command, 7-19
- serial number, 6-12, 7-19
- Serial Poll, 24-5
- Service, 6-23
- SETup Command/Query, 8-13
- 707 (device address), 2-9
- Short form, 1-9
- Simple command header, 1-5
- SINGLE time base mode, 16-4
- SLOPe Command/Query, 17-35
- SOURce Command/Query, 12-20, 15-41, 17-36, 18-17
- space, 27-6
- Spaces, 1-5
- *SRE Command/Query, 6-23
- standard + width, 26-4
- standard - width, 26-4
- STANdard Command/Query, 17-45
- Standard Event Status Enable Register, 6-7
- Standard Event Status Register, 6-7, 6-10
- State Trigger Mode, 17-10, 17-16
- STATistics Command/Query, 15-42
- Status, 2-12
- Status Byte, 6-23, 24-4
- status data structures, 6-4
- STATus Query, 7-20
- status register, 6-3
- Status registers, 2-12
- Status Reporting, 24-1
- *STB Command, 6-25
- STOP Command, 7-21
- STORE Command, 7-22
- STRing command, 12-21
- string program data, 27-14
- string response data, 27-23
- String variables, 2-9
- Subsystem
 - Acquire, 9-1
 - Calibrate, 10-1
 - Channel, 11-1
 - Display, 12-1
 - Function, 13-1
 - HARDcopy, 14-1
 - Marker, 23-1
 - Measure, 15-1
 - Multiple Memory, 21-1
 - Pixel Memory, 20-1
 - Sequential, 22-1
 - System, 8-1
 - Timebase, 16-1
 - Trigger, 17-1
 - Waveform, 18-1
 - Waveform Memory, 19-1
- Subsystem commands, 4-10
- Subsystem:Channel, 11-1

- SUBTRACT Command, 13-24
- suffix multipliers, 27-16
- suffix program data, 27-14
- suffix units, 27-16
- sweep speed, 7-5
- Syntax Diagram
 - Acquire Subsystem, 9-1
 - Calibrate Subsystem, 10-1
 - Channel Subsystem, 11-2
 - Common Commands, 6-1
 - Display Subsystem, 12-2
 - Function Subsystem, 13-2
 - Hardcopy Subsystem, 14-1
 - Marker Subsystem, 23-1
 - Measure Subsystem, 15-2
 - Multiple Memory Subsystem, 21-1
 - Pixel Memory Subsystem, 20-1
 - Root Level Commands, 7-1
 - Sequential Subsystem, 22-2
 - System Subsystem, 8-1
 - Waveform Subsystem, 18-2
 - Waveform Memory Subsystem, 19-2
- Syntax Diagram:Trigger Subsystem, 17-2
- syntax diagrams, 27-5
- syntax error, 27-4
- Syntax Overview, 27-5
- System Subsystem, 8-1
 - DATE Command/Query, 8-3
 - DSP Command/Query, 8-4
 - ERROR Query, 8-5
 - HEADER Command/Query, 8-8
 - KEY Command/Query, 8-9
 - LONGform Command/Query, 8-11
 - PIMacro Command, 8-12
 - SETup Command/Query, 8-13
 - Syntax Diagram, 8-2
 - TIME Command/Query, 8-15

T

- Talking to the instrument, 1-2
- TDELta Query, 15-43
- TER Query, 7-23
- Terminator, 1-12
- TEXT Command, 12-22
- Timebase Subsystem, 16-1
 - DELay Command/Query, 16-3
 - MODE Command/Query, 16-4
 - RANGE Command/Query, 16-5
 - REFERence Command/Query, 16-6
 - SAMPle Command/Query, 16-7
 - SETup Query, 16-8
- TMARKer Command/Query, 12-23
- TMAX Query, 15-44
- TMIN Query, 15-45
- TNULl Command/Query, 10-4
- *TRG Command, 6-27
- TRG - trigger, 24-3
- Trigger Bit, 24-4
- Trigger Condition command, 17-9 / 17-10
- Trigger Delay command, 17-11
- Trigger Delay Source command, 17-11
- Trigger Event, 17-11
- Trigger Event command, 17-11
- Trigger Field command, 17-12
- Trigger Holdoff, 17-9
- trigger level, 7-5
- Trigger Level command, 17-6, 17-8
- Trigger Logic command, 17-9 / 17-10
- Trigger Mode, 17-6
 - EDGE, 17-6, 17-8
 - TV, 17-6
 - Valid Commands, 17-7
- Trigger Occurrence command, 17-11, 17-13
- Trigger Occurrence Slope command, 17-13
- Trigger Path command, 17-9 / 17-10
- Trigger Polarity command, 17-12
- Trigger Qualify command, 17-11, 17-13

Trigger Slope command, 17-8
 Trigger Source command, 17-8, 17-11
 Trigger Standard command, 17-12
 Trigger Subsystem, 17-1
 CENTERed Command, 17-15
 CONDition Command/Query, 17-16
 COUPLing Command/Query, 17-18
 DELay Command/Query, 17-19
 DELay:SLOPe Command/Query, 17-20
 DELay SOURce Command/Query, 17-21
 FIELd Command/Query, 17-22
 GLITCh, 17-23
 GLITCh:CENTERed, 17-24
 GLITCh:HOLDoff, 17-25
 GLITCh:LEVel, 17-26
 GLITCh:SOURce, 17-27
 GLITCh:WIDTh, 17-28
 HOLDoff Command/Query, 17-29
 LEVel Command/Query, 17-30
 LINE Command/Query, 17-31
 LOGic Command/Query, 17-32
 MODE Command/Query, 17-33
 OCCurrence Command/Query, 17-34
 OCCurrence:SLOPe Command/Query, 17-35
 OCCurrence:SOURce Command/Query, 17-36
 PATH Command/Query, 17-37
 POLarity Command/Query, 17-38
 PROBe Command/Query, 17-39
 QUALify Command/Query, 17-40
 SENSitivity Command/Query, 17-41
 SETup Query, 17-42
 SLOPe Command/Query, 17-43
 SOURce Command/Query, 17-44
 STANdard Command/Query, 17-45
 Syntax Diagram, 17-2
 Trigger Time command, 17-11
 TRIGGERED time base mode, 16-4
 True RMS, 15-56
 Truncation Rules, 4-1
 *TST Query, 6-28
 TSTArt Command/Query, 15-46

TSTOp Command/Query, 15-47
 TTL
 undo, 6-18
 TTL Command, 11-13
 TV Trigger Mode, 17-6, 17-12, 17-16
 TVOLT Query, 15-48
 TYPE Command/Query, 9-10
 TYPE Query, 18-18

U

UNITs Command/Query, 15-50
 Unterminated Condition, 27-4
 UPPer Command/Query, 15-51
 upper/lower case equivalence, 27-8
 Uppercase, 1-9
 URQ - user request, 24-3
 URQ, user request, 6-7
 User-Defined Measurements, 15-10

V

VACRms Command/Query, 15-52
 Vamp, 26-5
 VAMPLitude Command/Query, 15-53
 VAverage Command/Query, 15-54
 Vavg, 26-6
 Vbase, 26-5
 VBASE Command/Query, 15-55
 VDCRms Command/Query, 15-56
 VDELta Query, 15-57
 VERSus Command, 13-25
 vertical input, 7-5
 vertical offset, 7-5
 vertical sensitivity, 7-5
 VFIFty Command, 15-58
 VIEW command, 7-7, 7-24
 VMARker Command/Query, 12-23

Vmax, 26-5
 VMAX Command/Query, 15-59
 Vmin, 26-5
 VMIN Command/Query, 15-60
 Voltage measurements, 15-13
 Vp-p, 26-5
 VPP Command/Query, 15-61
 VRELative Command/Query, 15-62
 Vrms, 26-6
 VRMS Command/Query, 15-64
 VSTart Command/Query, 15-65
 VSTop Command/Query, 15-66
 VTIme Query, 15-67
 Vtop, 26-5
 VTOP Command/Query, 15-68

W

*WAI Command, 6-29
 Waveform Data Conversion, 18-8
 Waveform Compare Flowchart, 15-69
 Waveform Subsystem, 18-2
 COUNT, nonimplemented, 4-22
 DATA Command/Query, 18-11
 FORMAT Command/Query, 18-13
 POINTs Query, 18-14
 PREAmble Command/Query, 18-15
 SOURce Command/Query, 18-17
 Syntax Diagram, 18-2
 TYPE Query, 18-18
 XINCrement Query, 18-19
 XORigin Query, 18-20
 XREFerence Query, 18-21
 YINCrement Query, 18-22
 YORigin Query, 18-23
 YREFerence Query, 18-24
 Waveform Memory Subsystem, 19-1
 DISPlay Command/Query, 19-3
 GET Command, 19-4
 PROTeCt Command/Query, 19-5

**HP 54505B/54506B/54510B/54512B
 Programming Reference**

SETup Query, 19-6
 Syntax Diagram, 19-2
 XOFFset Command/Query, 19-7
 XRANge Command/Query, 19-8
 YOFFset Query, 19-9
 YRANge Query, 19-10
 White space, 1-5, 27-8
 – width, 26-4
 + width, 26-4
 WMEMory Subsystem, 19-1
 WORD format, 18-9

X

XDELta Query, 23-5
 XINCrement Query, 18-19
 xmarkers, 23-6, 23-7
 XORigin Query, 18-20
 XREFerence Query, 18-21

Y

YDELta Query, 23-10
 YINCrement Query, 18-22
 ymarkers, 23-11, 23-12
 YORigin Query, 18-23
 YREFerence Query, 18-24