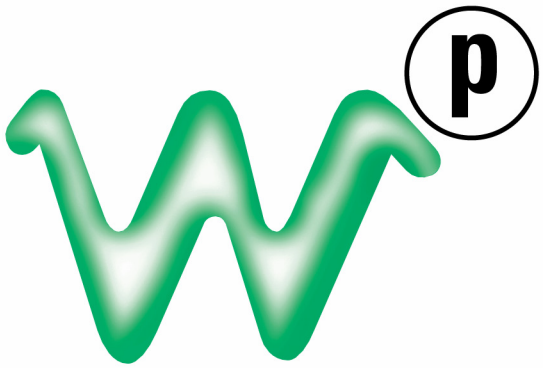


wavepro™

REMOTE CONTROL MANUAL

JANUARY 2002

LeCroy Corporation
700 Chestnut Ridge Road
Chestnut Ridge, NY 10977–6499
Tel: (845) 578 6020, Fax: (845) 578 5985

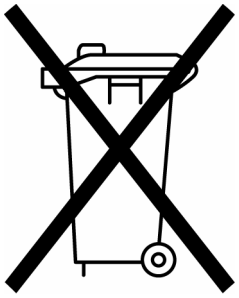
Internet: www.lecroy.com

© 2002 by LeCroy Corporation. All rights reserved. Information in this publication supersedes all earlier versions. Specifications subject to change.

LeCroy, ProBus and SMART Trigger are registered trademarks, and ActiveDSO, ScopeExplorer, WaveAnalyzer, and WavePro are trademarks, of LeCroy Corporation. Centronics is a registered trademark of Data Computer Corp. Epson is a registered trademark of Epson America Inc. MathCad is a registered trademark of MATHSOFT Inc. MATLAB is a registered trademark of The MathWorks, Inc. Microsoft, MS and Microsoft Access are registered trademarks, and Windows and NT trademarks, of Microsoft Corporation. PowerPC is a registered trademark of IBM Microelectronics. DeskJet, ThinkJet, QuietJet, LaserJet, PaintJet, HP 7470 and HP 7550 are registered trademarks of Hewlett-Packard Company.

Manufactured under an ISO 9000 Registered Quality Management System

Visit www.lecroy.com to view the certificate.



This electronic product is subject to disposal and recycling regulations that vary by country and region. Many countries prohibit the disposal of waste electronic equipment in standard waste receptacles.

For more information about proper disposal and recycling of your LeCroy product, please visit www.lecroy.com/recycle.

WP-RCM-E Rev C

INTRODUCTION 1

PART ONE: ABOUT REMOTE CONTROL 3

CHAPTER ONE: *OVERVIEW* 5

Operate WavePro by Remote Control 5
STANDARDS 6
PROGRAM MESSAGES 6
COMMANDS AND QUERIES 7
HEADERS 8
HEADER PATHS 8
DATA 9
CHARACTER DATA 9
NUMERIC DATA 9
STRING DATA 10
BLOCK DATA 10
RESPONSE MESSAGES 10
USE SCOPEEXPLORER™ 11

CHAPTER TWO: *CONTROL BY GPIB* 13

Talk, Listen or Control 13
INTERFACE 13
ADDRESS 14
GPIB SIGNALS 14
I/O BUFFERS 14
USE IEEE 488.1 STANDARD MESSAGES 15
DEVICE CLEAR 15
GROUP EXECUTE TRIGGER 15
REMOTE ENABLE 15
INTERFACE CLEAR 16
CONFIGURE THE GPIB-DRIVER SOFTWARE 16
MAKE SIMPLE TRANSFERS 17
USE ADDITIONAL DRIVER CALLS 19
MAKE SERVICE REQUESTS 19
Take Instrument Polls 21
DO CONTINUOUS POLLING 21
TAKE A SERIAL POLL 21
DO A PARALLEL POLL 22
PERFORM AN `*IST` POLL 24

Drive Hard-copy Devices on the GPIB **25**
READ DATA BY CONTROLLER 25
SEND DATA TO BOTH 25
TALK DIRECTLY TO PRINTER 26

CHAPTER THREE: *CONTROL BY RS-232* **29**

Communicate through the RS-232-C Port **29**
HANDSHAKE CONTROL 29
EDITING FEATURES 30
MESSAGE TERMINATORS 30
SRQ MESSAGE 31
LONG LINE SPLITTING 31
REMARKS 32
Simulate GPIB Messages **33**

CHAPTER FOUR: *UNDERSTAND AND MANAGE WAVEFORMS* **35**

Know Your Waveform **35**
LOGICAL DATA BLOCKS 35
INSPECT WAVEFORM CONTENTS 36
USE THE WAVEFORM QUERY 37
INTERPRET VERTICAL DATA 39
CALCULATE A DATA POINT'S HORIZONTAL POSITION 40
USE THE WAVEFORM COMMAND 42
Transfer Waveforms at High Speed **43**

CHAPTER FIVE: *CHECK WAVEFORM STATUS* **45**

Use Status Registers **45**
OVERVIEW 45
STATUS BYTE REGISTER (STB) 47
STANDARD EVENT STATUS REGISTER (ESR) 47
STANDARD EVENT STATUS ENABLE REGISTER (ESE) 48
SERVICE REQUEST ENABLE REGISTER (SRE) 48
PARALLEL POLL ENABLE REGISTER (PRE) 48
INTERNAL STATE CHANGE STATUS REGISTER (INR) 48
INTERNAL STATE CHANGE ENABLE REGISTER (INE) 49
COMMAND ERROR STATUS REGISTER (CMR) 49
DEVICE DEPENDENT ERROR STATUS REGISTER (DDR) 49
EXECUTION ERROR STATUS REGISTER (EXR) 49
USER REQUEST STATUS REGISTER (URR) 49

PART TWO: COMMANDS 51

Use WavePro Commands and Queries 53
COMMAND NOTATION 53
Table of Commands and Queries – By Short Form 55
Table of Commands and Queries – By Subsystem 59

APPENDIX I, *GPIB PROGRAM EXAMPLES* 275

Example 1 275
USE THE INTERACTIVE GPIB PROGRAM "IBIC" 275
Example 2 276
USE THE GPIB PROGRAM FOR IBM PC (HIGH-LEVEL FUNCTION CALLS) 276
Example 3 280
USE THE GPIB PROGRAM FOR IBM PC (LOW-LEVEL FUNCTION CALLS) 280

APPENDIX II, *WAVEFORM TEMPLATE* 283

Waveform Template 283

***INDEX* 295**

BLANK PAGE

About this Manual

This manual explains how to remotely control the oscilloscope, using commands keyed into the external controller. Normally, this controller is a computer, although it could be a simple terminal.

The manual includes a complete list of the commands you'll need to perform most Waverunner operations (you can find commands for a few special, optional functions in the software option's dedicated manual). The manual has two main parts:

Part One, "About Remote Control," covers the principles of remote control, and offers practical examples.

Part Two, "Commands," describes each of the remote control commands and queries for Waverunner operations. It starts with two special indexes that list the commands by short name and by category. Use these to find the command or query you wish to use.

See also the table of contents and the index at the back of the manual.

As an additional guide, each chapter is prefaced by a summary of its contents.

Watch for these icons and the information they signal:

TIPs offer additional hints on how to get the most out of Waverunner actions or features.

NOTEs bring to your attention important information you should know.

See also Chapter 12, "Use the *WavePro* DSO with a PC," in the *Operator's Manual*.

BLANK PAGE

PART ONE

ABOUT REMOTE CONTROL

Part One explains how the *WavePro* DSO operates under remote control. It covers GPIB and RS-232-C interfaces, the transfer and formatting of waveforms, and the use of status bytes in reporting errors.

CHAPTER ONE: *Overview*

In this chapter, see how to

Construct program messages

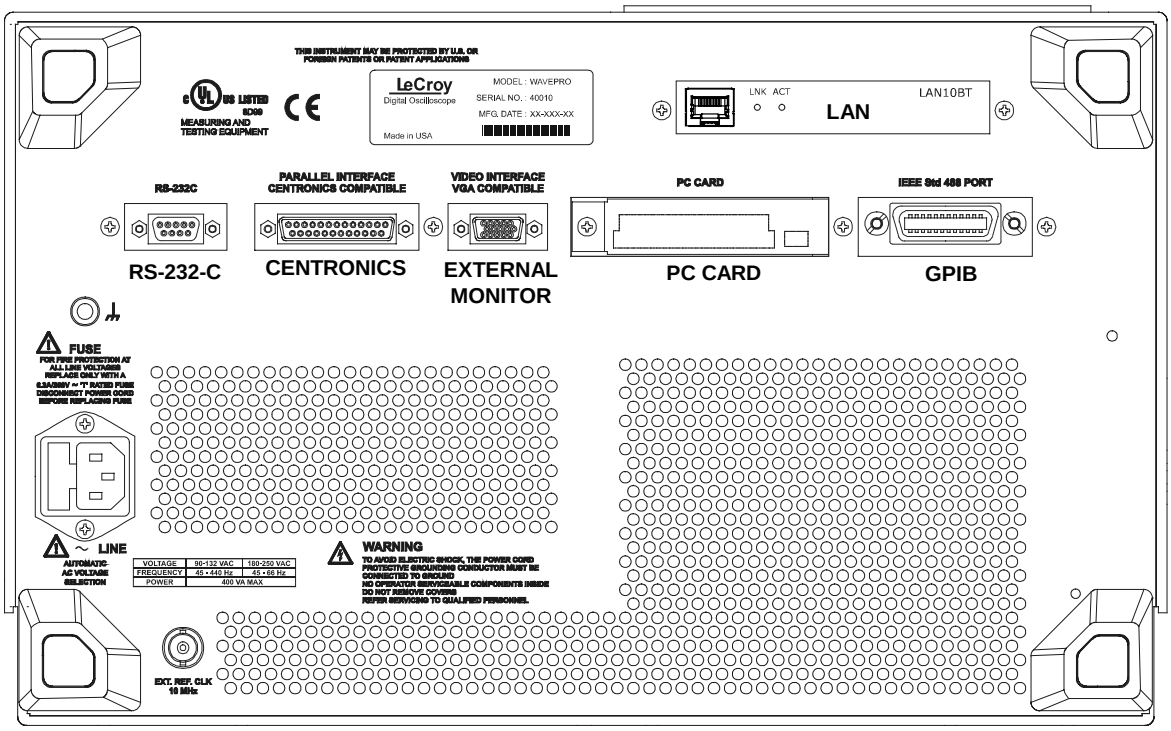
Use commands and queries

Include data, and make data strings

Use ScopeExplorer for remote control

Operate Your WavePro DSO by Remote Control

You can fully control your WavePro oscilloscope remotely by using either the GPIB (General Purpose Interface Bus) port or the RS-232-C communication port on the scope rear panel, shown below. The only actions for which you must use the front panel controls are the powering up of the scope and the setting of remote addresses. Use LeCroy's ScopeExplorer software as the ideal interface between scope and PC (see page 11).



WavePro back panel, including the GPIB and RS-232-C ports used in remote control.

TIP: Use WavePro Remote Control Assistant to monitor all your remote control operations. See the `COMM_HELP` command in Part Two of this manual, and Chapter 12 of the* Operator's Manual*, "Use WavePro with PC."

STANDARDS

LeCroy remote control commands conform to the GPIB IEEE 488.2* standard. This may be considered an extension of the IEEE 488.1 standard, which deals mainly with electrical and mechanical issues. The IEEE 488.2 recommendations have also been adopted for RS-232-C communications wherever appropriate.

PROGRAM MESSAGES

You control the oscilloscope remotely using program messages that consist of one or several commands or queries. The program messages you send from the external controller to the WavePro oscilloscope must conform to precise format structures. The oscilloscope will execute all program messages sent in the correct form, but will ignore those with errors.

You can use upper- or lower-case characters, or both, in program messages.

Warning or error messages are normally not reported unless the controller explicitly examines the relevant status register, or if the status-enable registers have been set so that the controller can be interrupted when an error occurs. If you connect an external monitor to the WavePro's RS-232-C port, however, you will be able to observe all your remote control transactions, including error messages, as they happen. See the command `COMM_HELP` in Part Two, "Commands."

Program messages are separated by semicolons **;** and end in a terminator:
<command/query>;.....;<command/query> <terminator>.

The oscilloscope will not decode an incoming program message before receiving its terminator. The exception is when the program message is longer than the 256 byte input buffer; then the oscilloscope will start analyzing the message when the buffer is full. Commands and queries are executed in the order in which they are transmitted.

In GPIB mode, the following are valid terminators:

<NL>	New-line character (i.e. the ASCII new-line character, whose decimal value is 10).
<NL><EOI>	New-line character with a simultaneous <EOI> signal.
<EOI><EOI>	Signal together with the last character of the program message.

The <NL> <EOI> terminator is always used in response messages sent by the oscilloscope to the controller.

In RS-232-C communications, you can define the terminator with the command `COMM_RS232`. The default value is <CR>, which is the ASCII carriage return character, whose decimal value is 13.

NOTE: The <EOI> signal is a dedicated GPIB interface line, which can be set with a special call to the GPIB interface driver. Refer to the GPIB interface manufacturer's manual and support programs.

*ANSI/IEEE Std. 488.2–1987, IEEE Standard Codes, Formats, Protocols, and Common Commands. The Institute of Electrical and Electronics Engineers Inc., 345 East 47th Street, New York, NY 10017 USA.

COMMANDS AND QUERIES

Program messages are made up of one or more commands or queries. While the command directs the oscilloscope to change its state (for example, its timebase or vertical sensitivity) the query asks the oscilloscope about that state. Very often, you will use the same mnemonic for a command and a query, the query being identified by a **?** after the last character.

For example, to change the timebase to 2 ms/ div, send this command to the oscilloscope:
`TIME_DIV 2 M`

Or, to ask the oscilloscope about its timebase, send this query:
`TIME_DIV?`

A query causes the oscilloscope to send a response message. The control program should read this message with a 'read' instruction to the GPIB or RS-232-C interface of the controller.

The response message to the above query might be:
`TIME_DIV 10 NS`

The portion of the query preceding the question mark is repeated as part of the response message. If desired, this text can be suppressed with the command `COMM_HEADER`.

Depending on the state of the oscilloscope and the computation to be done, several seconds may pass before a response is received. Command interpretation does not have priority over other oscilloscope activities.

The general form of a command or a query consists of a command header, <header>, optionally followed by one or several parameters, <data>, separated by commas:

<header>[?] <data>**,** ...**,** <data>

The notation [?] shows that the question mark is optional (turning the command into a query).

There is a space between the header and the first parameter.

There are commas between parameters.

TIP: Set the controller I/O timeout conditions to three or more seconds to give the scope time to respond. An incorrect query will not get a response; and, if Remote Control Assistant is enabled, a beep will sound.

The following are examples of how program messages are made up of commands and queries...

`GRID DUAL`: This program message consists of a single command that instructs the oscilloscope to display a dual grid.

The terminator is not shown, as it is usually automatically added by the interface driver routine writing to GPIB or RS232.

`DZOM ON; DISPLAY OFF; DATE?`:This program message consists of two commands, followed by a query. They instruct the oscilloscope to turn on the multi-zoom mode, turn off the display, and then ask for the current date. Again, the terminator is not shown.

`DATE 15,JAN,1993,13,21,16`: This command instructs the oscilloscope to set its date and time to 15 JAN 1993, 13:21:16. The command header `DATE` indicates the action, the 6 data values specify it in detail.

HEADERS

The header is the mnemonic form of the operation to be performed by the oscilloscope. Most command and query headers have a long form, which allows them to be read more easily, and a short form for better transfer and decoding speed. The two are fully equivalent and you can use them interchangeably. For example, `TRIG_MODE AUTO` and `TRMD AUTO` are two separate but equivalent commands for switching to the automatic trigger mode.

Some command or query mnemonics are imposed by the IEEE 488.2 standard. They are standardized so that different oscilloscopes will present the same programming interface for similar functions. All these mnemonics begin with an asterisk `*`. For example, the command `*RST` is the IEEE 488.2 imposed mnemonic for resetting the oscilloscope, whereas `*TST?` instructs the oscilloscope to perform an internal self-test and report the outcome.

HEADER PATHS

Certain commands or queries apply to a sub-section of the oscilloscope; for example, a single input channel or a trace on the display. In such cases, you must prefix the header by a path name that indicates the channel or trace to which the command applies. The header path normally consists of a two-letter path name followed by a colon `:` immediately preceding the command header. One of the waveform traces can usually be specified in the header path:

HEADER PATH NAME	WAVEFORM TRACE
`C1`, `C2`	Channels 1 and 2
`C3`, `C4`	Channels 3 and 4 (on four-channel models)
`M1`, `M2`, `M3`, `M4`	Memories 1, 2, and 3 and 4
`TA`, `TB`, `TC`, `TD`	Traces A, B, C and D
`EX`, `EX10`, `EX5`	External trigger
`LINE`	LINE source for trigger

Example: `C1:OFST -300 MV` Command to set the offset of Channel 1 to –300 mV.

You need only specify a header path once. Subsequent commands with header destinations not indicated are assumed to refer to the last defined path. For example, the queries `C2:VDIV?; C2:OFST?` ask: What is the vertical sensitivity and the offset of channel 2? While the queries `C2:VDIV?; OFST?` ask exactly the same question without repeating the path.

DATA

Whenever a command or query uses additional data values, the values are expressed as ASCII characters. There is a single exception: the transfer of waveforms with the command/query WAVEFORM, where the waveform can be expressed as a sequence of binary data values. See Chapter 4, "Waveform Structure." ASCII data can have the form of character, numeric, string, or block data.

CHARACTER DATA

These are simple words or abbreviations to indicate a specific action.

Example: `DUAL_ZOOM ON`

In this example, the data value `ON` commands the dual zoom mode to be turned on (the data value `OFF` will have the opposite effect).

However, this can become more complex. In some commands, where you can specify as many as a dozen different parameters, or where not all the parameters are applicable at the same time, the format requires pairs of data values. The first value names the parameter to be modified, while the second gives its value. Only those parameter pairs changed need to be indicated.

Example: `HARDCOPY_SETUP DEV,EPSON,PORT,GPIB`

In this example, two pairs of parameters have been used. The first specifies the device as an `EPSON` (or compatible) printer, while the second indicates the GPIB port. While the command `HARDCOPY_SETUP` allows many more parameters, either they are not relevant for printers or they are left unchanged.

NUMERIC DATA

The numeric data type is used to enter quantitative information. Numbers can be entered as integers or fractions, or in exponential representation:

`TA:VPOS -5`	Move the displayed trace of Trace A downwards by five divisions.
`C2:OFST 3.56`	Set the DC offset of Channel 2 to 3.56 V.
`TDIV 5.0E-6`	Adjust the timebase to 5 µsec/div.

Example: There are many ways of setting the timebase of the oscilloscope to 5 µsec/div:

`TDIV 5E-6`	Exponential notation, without any suffix.
`TDIV 5 US`	Suffix multiplier `U` for 1E−6, with the (optional) suffix `S` for seconds.

or

```
TDIV 5000 NS
TDIV 5000E-3 US
```

You can follow numeric values with multipliers and units to modify the value of the numerical expression. The following mnemonics are recognized:

MULTIPLIER	EXP. NOTE.	SUFFIX	MULTIPLIER	EXP. NOTE.	SUFFIX
EX	1E18	Exa-	PE	1E15	Peta-
T	1E12	Tera-	G	1E9	Giga-
MA	1E6	Mega-	K	1E3	kilo-
M	1E–3	milli-	U	1E–6	micro-
N	1E–9	nano-	PI	1E–12	pico-
F	1E–15	femto-	A	1E–18	atto-

STRING DATA

This data type enables you to transfer a (long) string of characters as a single parameter. Simply enclose any sequence of ASCII characters between single or double quotation marks:

```
MESSAGE 'Connect probe to point J3'
```

The oscilloscope displays this message in the Message field above the grid.

BLOCK DATA

These are binary data values coded in hexadecimal ASCII: four-bit nibbles translated into the digits 0 through 9 or A through F, and transmitted as ASCII characters. They are used only for the transfer of waveforms from WavePro to controller (`WAVEFORM`) and for WavePro panel setups (`PANEL_SETUP`).

RESPONSE MESSAGES

The oscilloscope sends a response message to the controller in answer to a query. The format of such messages is the same as that of program messages: individual responses in the format of commands, separated by semicolons **;** and ending in terminators. These messages can be sent back to the oscilloscope in the form in which they were received, to be accepted as valid commands. In GPIB response messages, the <NL> <EOI> terminator is always used.

Example: The controller sends the program message:
`TIME_DIV?;TRIG_MODE NORM;C1:COUPLING?` (terminator not shown).

The oscilloscope might respond to this with:
`TIME_DIV 50 NS;C1:COUPLING D50` (terminator not shown).

The response message refers only to the queries: `TRIG_MODE` is left out. If this response is sent back to the oscilloscope, it is a valid program message for setting its timebase to 50 ns/ div and the input coupling of Channel 1 to 50 Ω.

Whenever you expect a response from the oscilloscope, you must have the control program instruct the GPIB or RS-232-C interface to read from the oscilloscope. If the controller sends another program message without reading the response to the previous one, the response message in the output buffer of the oscilloscope will be

discarded. The oscilloscope keeps to stricter rules for response messages than for acceptance of program messages. While you can send program messages from the controller in upper- or lower-case characters, response messages are always returned in upper-case. Program messages may contain extraneous spaces or tabs (white space), but response messages will not. And while program messages may contain a mixture of short and long command or query headers, response messages always use short headers by default.

However, you can use the command `COMM_HEADER` to force the oscilloscope to use long headers, or none at all. If the response header is omitted, the response transfer time will be minimized. But the response will not be able to be sent back to the oscilloscope. Suffix units are also suppressed in the response.

If you were to set the trigger slope of Channel 1 to negative, the query `C1:TRSL?` might yield the following responses:

`C1:TRIG_SLOPE NEG`	header format: long
`C1:TRSL NEG`	header format: short
`NEG`	header format: off

TIP: Waveforms you obtain from the oscilloscope using the query `WAVEFORM?` are a special kind of response message. Control their exact format by using the `COMM_FORMAT` and `COMM_ORDER` commands.

USE SCOPEEXPLORER

ScopeExplorer is an easy-to-use and practical software tool for interfacing your WavePro oscilloscope with a PC running Windows:

1. Connect the scope to a PC using either the GPIB (you'll need a PC with GPIB card installed) or PC-standard RS-232-C port on the scope's rear panel.
2. Download ScopeExplorer free of charge at http://www.lecroy.com/scopeexplorer. Or inquire at your LeCroy customer service center.
3. Having installed ScopeExplorer, open it as you would any Windows program. Use its on-line help to do the following:

   Use the teletype-like terminal to send standard remote control commands from computer to oscilloscope and to display the WavePro response on the PC.

   Control the scope by means of an interactive, virtual scope front panel.

   Pipe sequences of commands from a file to the scope, then send the scope's responses to another file.

   Transfer pixel-for-pixel copies of your WavePro display to PC, then view them, print them, or both from the computer. With a single press of a button or key, you can copy bitmap waveform images to the Windows Clipboard, ready to paste into any Windows application.

   Capture WavePro front panel setups, and, using a long filename, store them on the computer. You can then transfer them back into the scope to reproduce an identical setup.

   Transfer your waveforms to PC, and store them in either the compact LeCroy Binary format, or an ASCII version compatible with PC-based analysis products.

CHAPTER TWO: *Control by GPIB*

In this chapter, see how to

- ***Address your WavePro scope for GPIB***
- ***Configure GPIB software***
- ***Enable remote or local control***
- ***Make transfers of data***
- ***Make service requests***
- ***Poll WavePro***
- ***Drive hard-copy devices***

Talk, Listen, or Control

You can remotely control your WavePro oscilloscope, using the General Purpose Interface Bus (GPIB). GPIB is similar to a standard computer bus. But while the computer interconnects circuit cards by means of a backplane bus, the GPIB interconnects independent devices (oscilloscopes and computers, for example) by means of a cable bus. GPIB also carries both program and interface messages.

Program messages, often called device dependent messages, contain programming instructions, measurement results, oscilloscope status and waveform data.

Interface messages manage the bus itself. They perform functions such as initialization, addressing and "unaddressing" of devices, and the setting of remote and local modes.

On the one hand, devices connected by GPIB to your WavePro oscilloscope can be listeners, talkers, or controllers. A talker sends program messages to one or more listeners, while a controller manages the flow of information on the bus by sending interface messages to the devices. The host computer must be able to play all three roles. For details of how the controller configures the GPIB for specific functions, refer to the GPIB interface manufacturer's manual.

On the other hand, the WavePro can be a talker or listener, but NOT a controller.

INTERFACE

WavePro interface capabilities include the following IEEE 488.1 definitions:

AH1	Complete Acceptor Handshake	**DC1**	Complete Device Clear Function
SH1	Complete Source Handshake	**DT1**	Complete Device Trigger
L4	Partial Listener Function	**PP1**	Parallel Polling: remote configurable
T5	Complete Talker Function	**C0**	No Controller Functions
SR1	Complete Service Request Function	**E2**	Tri-state Drivers
RL1	Complete Remote/Local Function		

ADDRESS

Every device on the GPIB has an address. To address WavePro, set the remote control port to **GPIB** by means of the scope's front panel UTILITIES button and on-screen menus. If you select "RS-232" in the same way, the oscilloscope will execute over the GPIB solely "talk-only" operations, such as driving a printer. Setting WavePro to "RS-232" enables the oscilloscope to be controlled through the RS-232-C port. See Chapter 12 of the *Operator's Manual* for how to do this.

If you address WavePro to talk, it will remain in that state until it receives a universal untalk command (UNT), its own listen address (MLA), or another oscilloscope's talk address.

If you address WavePro to listen, it will remain configured to listen until a universal unlisten command (UNL), or its own talker address (MTA), is received.

GPIB SIGNALS

The GPIB system consists of 16 signal lines and eight ground or shield lines. The signal lines are divided into three groups:

Data Lines: These eight lines, usually called DIO1 through DIO8, carry both program and interface messages. Most of the messages use the 7-bit ASCII code, in which case DIO8 is unused.

Handshake Lines: These three lines control the transfer of message bytes between devices. The process is called a three-wire interlocked handshake, and it guarantees that the message bytes on the data lines are sent and received without transmission error.

Interface Management Lines: These five lines manage the flow of information across the interface:

ATN (ATteNtion): The controller drives the ATN line true when it uses the data lines to send interface messages such as talk and listen addresses or a device clear (DCL) message. When ATN is false, the bus is in data mode for the transfer of program messages from talkers to listeners.

IFC (InterFace Clear): The controller sets the IFC line true to initialize the bus.

REN (Remote ENable): The controller uses this line to place devices in remote or local program mode.

SRQ (Service ReQuest): Any device can drive the SRQ line true to asynchronously request service from the controller. This is the equivalent of a single interrupt line on a computer bus.

EOI (End Or Identify): This line has two purposes: The talker uses it to mark the end of a message string. The controller uses it to tell devices to identify their response in a parallel poll (discussed later in this section).

I/O BUFFERS

The oscilloscope has 256-byte input and output buffers. An incoming program message is not decoded before a message terminator has been received. However, if the input buffer becomes full (because the program message is longer than the buffer), the oscilloscope starts analyzing the message. In this case, data transmission is temporarily halted, and the controller may generate a timeout if the limit was set too low.

USE IEEE 488.1 STANDARD MESSAGES

The IEEE 488.1 standard specifies not only the mechanical and electrical aspects of the GPIB, but also the low-level transfer protocol. For instance, it defines how a controller addresses devices, turns them into talkers or listeners, resets them, or puts them in the remote state. Such interface messages are executed with the interface management lines of the GPIB, usually with ATN true.

All these messages except GET are executed immediately upon receipt.

The command list in Part Two of this manual does not contain a command for clearing the input or output buffers, nor for setting the oscilloscope to the remote state.

NOTE: In addition to the IEEE 488.1 interface message standards, the IEEE 488.2 standard specifies certain standardized program messages, i.e., command headers. They are identified with a leading asterisk * and are listed in the System Commands section.

This is because such commands are already specified as IEEE 488.1 standard messages. Refer to the GPIB interface manual of the host controller as well as to its support programs, which should contain special calls for the execution of these messages.

The following description covers those IEEE 488.1 standard messages that go beyond mere reconfiguration of the bus and that have an effect on WavePro operation.

DEVICE CLEAR

In response to a universal Device CLear (DCL) or a Selected Device Clear message (SDC), WavePro clears the input or output buffers, cancels the interpretation of the current command (if any) and clears pending commands. However, status registers and status-enable registers are *not* cleared. Although DCL will have an immediate effect, it can take several seconds to execute if the oscilloscope is busy.

GROUP EXECUTE TRIGGER

The Group Execute Trigger message (GET) causes WavePro to arm the trigger system, and is functionally identical to the *TRG command.

REMOTE ENABLE

This interface message is executed when the controller holds the Remote ENable control line (REN) true, allowing you to configure the oscilloscope as a listener. All the front panel controls except the menu buttons are disabled. The menu indications on the right-hand side of the screen no longer appear, since menus cannot now be operated manually. Instead, the text REMOTE ENABLE appears at the top of the menu field to indicate that the oscilloscope is set in the remote mode. Whenever the controller returns the REN line to false, all oscilloscopes on the bus return to GO TO LOCAL.

When you press the GO TO LOCAL menu button, the scope returns to front panel control, unless you have placed the oscilloscope in Local LOckout (LLO) mode (see below).

The Go To Local message (GTL) causes the oscilloscope to return to local mode. All front panel controls become active and the normal menus reappear. Thereafter, whenever the oscilloscope is addressed as a listener it will be immediately reset to the remote state, except when the LLO command has been sent.

When you activate Local Lockout, the scope can only be returned to its local state by returning the LLO to false. Whenever you return the oscilloscope to the remote state the local lockout mode will immediately become effective again.

The LLO message causes the GO TO LOCAL menu to disappear. You can send this message in local or remote mode. But it only becomes effective once you have set the oscilloscope in remote mode.

INTERFACE CLEAR

The InterFace Clear message (IFC) initializes the GPIB but has no effect on the operation of the WavePro.

NOTE: To illustrate the GPIB programming concepts, a number of examples written in BASICA are included here. It is assumed that the controller is IBM-PC compatible, running under DOS, and that it is equipped with a National Instruments GPIB interface card. Nevertheless, GPIB programming with other languages such as C or Pascal is quite similar. If you're using another type of computer or GPIB interface, refer to the interface manual for installation procedures and subroutine calls.

CONFIGURE THE GPIB DRIVER SOFTWARE

1. Verify that the GPIB interface is properly installed in the computer. If it is not, follow the interface manufacturer's installation instructions. In the case of the National Instruments interface, it is possible to modify the base I/O address of the board, the DMA channel number, and the interrupt line setting using switches and jumpers. In the program examples below, default positions are assumed.
2. Connect WavePro to the computer with a GPIB interface cable.
3. Set the GPIB address to the required value. The program examples assume a setting of **4**.

The host computer requires an interface driver that handles the transactions between the operator's programs and the interface board.

In the case of the National Instruments interface, the installation procedure will:

a. Copy the GPIB handler GPIB.COM into the boot directory.

b. Modify the DOS system configuration file CONFIG.SYS to declare the presence of the GPIB handler.

c. Create a sub-directory called GPIB-PC, and install in GPIB-PC a number of files and programs useful for testing and reconfiguring the system and for writing user programs.

The following files in the sub-directory GPIB-PC are particularly useful:

IBIC.EXE allows interactive control of the GPIB by means of functions entered at the keyboard. Use of this program is highly recommended to anyone unfamiliar with GPIB programming or with WavePro's remote commands.

DECL.BAS is a declaration file that contains code to be included at the beginning of any BASICA application program. Simple application programs can be quickly written by appending the operator's instructions to DECL.BAS and executing the complete file.

IBCONF.EXE is an interactive program that allows inspection or modification of the current settings of the GPIB handler. To run IBCONF.EXE, refer to the National Instruments manual.

NOTE: In the program examples in this section, it is assumed that the National Instruments GPIB driver GPIB.COM is in its default state, i.e. that the user has not modified it with IBCONF.EXE. This means that the interface board can be referred to by the symbolic name 'GPIB0' and that devices on the GPIB bus with addresses between 1 and 16 can be called by the symbolic names 'DEV1' to 'DEV16'. If you have a National Instruments PC2 interface card rather than PC2A, you must run IBCONF to declare the presence of this card rather than the default PC2A.

MAKE SIMPLE TRANSFERS

For a large number of remote control operations, it is sufficient to use just three different subroutines (IBFIND, IBRD and IBWRT) provided by National Instruments. The following complete program reads the timebase setting of WavePro and displays it on the terminal:

```
1–99    <DECL.BAS>
100     DEV$="DEV4"
110     CALL IBFIND(DEV$,SCOPE%)
120     CMD$="TDIV?"
130     CALL IBWRT(SCOPE%,CMD$)
140     CALL IBRD(SCOPE%,RD$)
150     PRINT RD$
160     END
```

Lines 1–99 are a copy of the file DECL.BAS supplied by National Instruments. The first six lines are required for the initialization of the GPIB handler. The other lines are declarations which may be useful for larger programs, but are not really required code. The sample program above only uses the strings `CMD$` and `RD$`, which are declared in DECL.BAS as arrays of 255 characters.

Lines 100 and 110 open the device `DEV4` and associate with it the descriptor `SCOPE%`. All I/O calls after that will refer to `SCOPE%`. The default configuration of the GPIB handler recognizes `DEV4` and associates with it a device with the GPIB address 4.

Lines 120 and 130 prepare the command string `TDIV?` and transfer it to the oscilloscope. The command instructs the oscilloscope to respond with the current setting of the timebase.

Lines 140 and 150 read the response of the oscilloscope and place it into the character string RD$.

Line 170 displays the response on the terminal.

> **NOTE:** DECL.BAS requires access to the file BIB.M during the GPIB initialization. BIB.M is one of the files supplied by National Instruments, and it must exist in the directory currently in use.

The first two lines of DECL.BAS both contain a string XXXXX, which must be replaced by the number of bytes that determine the maximum workspace for BASICA (computed by subtracting the size of BIB.M from the space currently available in BASICA). For example, if the size of BIB.M is 1200 bytes, and when BASICA is loaded it reports "60200 bytes free," you should replace "XXXXX" by the value 59 000 or less.

When running this sample program, WavePro will automatically be set to the remote state when IBWRT is executed, and will remain in that state. Pressing the LOCAL menu button will return WavePro to local mode if the GPIB handler was modified to inhibit Local LOckout (LLO). Here is a slightly modified version of the sample program that checks if any error occurred during GPIB operation:

```
1-99     <DECL.BAS>
100      DEV$="DEV4"
110      CALL IBFIND(DEV$,SCOPE%)
120      CMD$="TDIV?"
130      CALL IBWRT(SCOPE%,CMD$)
140      IF ISTA% < 0 THEN GOTO 200
150      CALL IBRD(SCOPE%,RD$)
160      IF ISTA% < 0 THEN GOTO 250
170      PRINT RD$
180      IBLOC(SCOPE%)
190      END
200      PRINT "WRITE ERROR =";IBERR%
210      END
250      PRINT "READ ERROR =";IBERR%
260      END
```

The GPIB status word ISTA%, the GPIB error variable IBERR% and the count variable IBCNT% are defined by the GPIB handler and are updated with every GPIB function call. Refer to the National Instruments manual for details. The sample program above would report if the GPIB address of the oscilloscope was set to a value other then 4. Line 180 resets the oscilloscope to local with a call to the GPIB routine IBLOC.

USE ADDITIONAL DRIVER CALLS

IBLOC is used to execute the IEEE 488.1 standard message Go To Local (GTL), i.e. it returns the oscilloscope to the local state. The programming example above illustrates its use.

IBCLR executes the IEEE 488.1 standard message Selected Device Clear (SDC).

IBRDF and **IBWRTF**, respectively, allow data to be read from GPIB to a file, and written from a file to GPIB. Transferring data directly to or from a storage device does not limit the size of the data block, but may be slower than transferring to the computer memory.

IBRDI and IBWRTI allow data to be read from GPIB to an integer array, and written from integer array to GPIB. Since the integer array allows storage of up to 64 kilobytes (in BASIC), IBRDI and IBWRTI should be used for the transfer of large data blocks to the computer memory, rather than IBRD or IBWRT, which are limited to 256 bytes by the BASIC string length. Note that IBRDI and IBWRTI only exist for BASIC, since for more modern programming languages, such as C, the functions called IBRD and IBWRT are far less limited in data block size.

IBTMO can be used to change the timeout value during program execution. The default value of the GPIB driver is 10 seconds — for example, if the oscilloscope does not respond to an IBRD call, IBRD will return with an error after the specified time.

IBTRG executes the IEEE 488.1 standard message Group Execute Trigger (GET), which causes WavePro to arm the trigger system.

National Instruments supplies a number of additional function calls. In particular, it is possible to use the so-called board level calls, which allow a very detailed control of the GPIB.

NOTE: The SRQ bit is latched until the controller reads the STatus Byte Register (STB). The action of reading the STB with the command `*STB?` clears the register contents except the MAV bit (bit 4) until a new event occurs. Service requesting can be disabled by clearing the SRE register with the `*SRE 0` command.

MAKE SERVICE REQUESTS

When a WavePro is used in a remote application, events often occur asynchronously, i.e. at times that are unpredictable for the host computer. The most common example of this is a trigger wait after the oscilloscope is armed: the controller must wait until the acquisition is finished before it can read the acquired waveform. The simplest way of checking if a certain event has occurred is by either continuously or periodically reading the status bit associated with it until the required transition is detected. Continuous status bit polling is described in more detail below. For a complete explanation of status bits, refer to Chapter 5.

Perhaps a more efficient way of detecting events occurring in the oscilloscope is the use of the Service ReQuest (SRQ). This GPIB interrupt line can be used to interrupt program execution in the controller. The controller can then execute other programs while waiting for the oscilloscope. Unfortunately, not all interface manufacturers support the programming of interrupt service routines. In particular, National Instruments supports only the SRQ bit within the ISTA% status word. This requires you to continuously or periodically

check this word, either explicitly or with the function call IBWAIT. In the absence of real interrupt service routines, the use of SRQ may not be very advantageous.

In the default state, after power-on, the Service ReQuest is disabled. You enable SRQ by setting the Service Request Enable register with the command "*SRE" and by specifying which event should generate an SRQ. WavePro will interrupt the controller as soon as the selected event(s) occur by asserting the SRQ interface line. If several devices are connected to the GPIB, you may be required to identify which oscilloscope caused the interrupt by serial polling the various devices.

Example: To assert SRQ in response to the events "new signal acquired" or "return-to-local" (pressing the soft key/menu button for GO TO LOCAL).These events are tracked by the INR register, which is reflected in the SRE register as the INB summary bit in position 0. Since bit position 0 has the value 1, the command `*SRE 1` enables the generation of SRQ whenever the INB summary bit is set.

In addition, the events of the INR register that may be summarized in the INB bit must be specified. The event "new signal acquired" corresponds to INE bit 0 (value 1) while the event "return-to-local" is assigned to INE bit 2 (value 4). The total sum is 1 + 4 = 5. Thus the command `INE 5` is needed:

```
CMD$="INE 5;*SRE 1"
CALL IBWRT(SCOPE%,CMD$)
```

Example: To assert SRQ when soft key 4 (fourth menu button from top of screen) is pressed. The event "soft key 4 pressed" is tracked by the URR register. Since the URR register is not directly reflected in STB but only in the ESR register (URR, bit position 6), the ESE enable register must be set first with the command `*ESE 64` to allow the URQ setting to be reported in STB. An SRQ request will now be generated provided that the ESB summary bit (bit position 5) in the SRE enable register is set (`*SRE 32`):

```
CMD$="*ESE 64;*SRE 32"
CALL IBWRT(SCOPE%,CMD$)
```

NOTE: The term "soft-key," used here in reference to remote operations, is synonymous with "menu button," used in the accompanying Operator's Manual to mean front panel operations. Both terms refer to the column of seven buttons running parallel to the screen on the WavePro front panel and the menu functions they control.

Take Instrument Polls

You can regularly monitor state transitions within the oscilloscope by polling selected internal status registers. There are four basic polling methods you can use to detect the occurrence of a given event: continuous, serial, parallel, and `*IST`. By far the simplest of these is continuous polling. The others are appropriate only when interrupt-service routines (servicing the SRQ line) are supported, or multiple devices on GPIB require constant monitoring. To emphasize the differences between the methods, described below, the same example (determining whether a new acquisition has taken place) is used in each case.

DO CONTINUOUS POLLING

A status register is continuously monitored until a transition is observed. This is the most straightforward method for detecting state changes, but may not be practical in certain situations, especially with multiple device configurations.

In the following example, the event "new signal acquired" is observed by continuously polling the INternal state change Register (INR) until the corresponding bit (in this case bit 0, i.e. value 1) is non-zero, indicating a new waveform has been acquired. Reading INR clears this at the same time, so that there is no need for an additional clearing action after a non-zero value has been detected. The command `CHDR OFF` instructs the oscilloscope to omit any command headers when responding to a query, simplifying the decoding of the response. The oscilloscope will then send "1" instead of "INR 1":

```
CMD$="CHDR OFF"
CALL IBWRT(SCOPE%,CMD$)
MASK% = 1'New Signal Bit has value 1'
LOOP% = 1
WHILE LOOP%
CMD$="INR?"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$)
NEWSIG% = VAL(RD$) AND MASK%
IF NEWSIG% = MASK% THEN LOOP% = 0
WEND
```

TAKE A SERIAL POLL

Serial polling takes place once the SRQ interrupt line has been asserted, and is only advantageous when you are using several oscilloscopes at once. The controller finds which oscilloscope has generated the interrupt by inspecting the SRQ bit in the STB register of each. Because the service request is based on an interrupt

mechanism, serial polling offers a reasonable compromise in terms of servicing speed in multiple-device configurations.

In the following example, the command `INE 1` enables the event "new signal acquired" to be reported in the INR to the INB bit of the status byte STB. The command `*SRE 1` enables the INB of the status byte to generate an SRQ whenever it is set. The function call IBWAIT instructs the computer to wait until one of three conditions occurs: &H8000 in the mask (MASK%) corresponds to a GPIB error, &H4000 to a timeout error, and &H0800 to the detection of RQS (ReQuest for Service) generated by the SRQ bit.

Whenever IBWAIT detects RQS, it automatically performs a serial poll to find out which oscilloscope generated the interrupt. It will only exit if there was a timeout or if the oscilloscope (`SCOPE%`) generated SRQ. The additional function call `IBRSP` fetches the value of the status byte, which may be further interpreted. For this to work properly, the value of "Disable Auto Serial Polling" must be set to "off" in the GPIB handler (use `IBCONF.EXE` to check).

```
CMD$="*CLS; INE 1; *SRE 1"
CALL IBWRT(SCOPE%,CMD$)
MASK% = &HC800
CALL IBWAIT(SCOPE%,MASK%)
IF (IBSTA% AND &HC000) <> 0 THEN PRINT "GPIB or Timeout Error" : STOP
CALL IBRSP(SCOPE%,SPR%)
PRINT "Status Byte =.", SPR%
```

Board-level function calls can deal simultaneously with several oscilloscopes attached to the same interface board. Refer to the National Instruments manual.

NOTE: After the serial poll is completed, the RQS bit in the STB status register is cleared. Note that the other STB register bits remain set until they are cleared by means of a "*CLS" command or the oscilloscope is reset. If these bits are not cleared, they cannot generate another interrupt.

DO A PARALLEL POLL

Like serial polling, this is only useful with several oscilloscopes. The controller simultaneously reads the Individual STatus bit (IST) of all oscilloscopes to determine which one needs service. This method allows up to eight different oscilloscopes to be polled at the same time.

When a parallel poll is initiated, each oscilloscope returns a status bit over one of the DIO data lines. Devices may respond either individually, using a separate DIO line, or collectively on a single data line. Data-line assignments are made by the controller using a Parallel Poll Configure (PPC) sequence.

In the following example, the command `INE 1` enables the event "new signal acquired" in the INR to be reported to the INB bit of the status byte STB. The PaRallel poll Enable register (PRE) determines which events will be summarized in the IST status bit. The command `*PRE 1` enables the INB bit to set the IST bit

whenever it is itself set. Once parallel polling has been established, the parallel-poll status is examined until a change on data bus line DIO2 takes place.

Stage 1

1. Enable the INE and PRE registers
2. Configure the controller for parallel poll
3. Instruct WavePro to respond on data line 2 (DIO2) with these commands:

```
CMD1$="?_@$"
CALL IBCMD(BRD0%,CMD1$)
CMD$="INE 1;*PRE 1"
CALL IBWRT(BRD0%,CMD$)
CMD4$=CHR$(&H5)+CHR$(&H69)+"?"
CALL IBCMD(BRD0%,CMD4$)
```

Stage 2

4. Parallel poll the oscilloscope until DIO2 is set with these commands:

```
LOOP% = 1
WHILE LOOP%
CALL IBRPP(BRD0%,PPR%)
IF (PPR% AND &H2) = 2 THEN LOOP% = 0
WEND
```

Stage 3

5. Disable parallel polling (hex 15) and clear the parallel poll register with these commands:

```
CMD5$=CHR$(&H15)
CALL IBCMD(BRD0%,CMD5$)
CALL IBCMD(BRD0%,CMD1$)
CMD$="*PRE 0"CALL IBWRT(BRD0%,CMD$):
```

In the above example, board-level GPIB function calls are used. It is assumed that the controller (board) and WavePro (device) are respectively located at addresses 0 and 4.

The listener and talker addresses for the controller and WavePro are:

LOGIC DEVICE	LISTENER ADDRESS	TALKER ADDRESS
External Controller	32 (ASCII<space>)	64 (ASCII @)
WavePro	32+4=36 (ASCII $)	64+4=68 (ASCII D)

PERFORM AN *IST POLL

You can also read the state of the Individual STatus bit (IST) returned in parallel polling by sending the `*IST?` query. To enable this poll mode, you must initialize WavePro as for parallel polling by writing into the PRE register. Since `*IST` emulates parallel polling, apply this method wherever parallel polling is not supported by the controller. In the following example, the command `INE 1` enables the event "new signal acquired" in the INR to be reported to the INB bit of the status byte STB. The command `*PRE 1` enables the INB bit to set the IST bit whenever it is set. The command `CHDR OFF` suppresses the command header in the oscilloscope's response, simplifying the interpretation. The status of the IST bit is then continuously monitored until set by the oscilloscope:

```
CMD$="CHDR OFF; INE 1; *PRE 1"
CALL IBWRT(SCOPE%,CMD$)
LOOP% = 1
WHILE LOOP%
CMD$="*IST?"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$)
IF VAL(RD$) = 1 THEN LOOP% = 0
WEND
```

NOTE: The characters "?" and "_" appearing in the command strings stand for unlisten and untalk respectively. They are used to set the devices to a "known" state. To shorten the size of the program examples, device talking and listening initialization instructions have been grouped into character chains. They are: `CMD1$ = "?_@$" Unlisten, Untalk, PC talker, DSO listener.` ***The remote message code for executing a parallel response in binary form is 01101PPP, where PPP specifies the data line. Because data line 2 is selected, the identification code is 001, which results in the code 01101001 (binary) or &H69 (hex). See Table 38 of the IEEE 488-1978 Standard for further details.***

Drive Hard-copy Devices on the GPIB

You can interface your WavePro oscilloscope with a wide range of hard-copy devices, such as printers and plotters, and copy the screen contents to them. List the devices supported using the command HARDCOPY_SETUP.

With a hard-copy device connected to the GPIB, you can use either of two basic configurations. When only WavePro and a hard-copy device such as a printer are connected, you must configure the oscilloscope as talker-only, and the hard-copy device as listener-only, to ensure proper data transfer. However, when an external controller is connected to the GPIB, you must use this controller to supervise the data transfers. You can then use a variety of schemes to transfer WavePro screen contents.

Configure WavePro as talker-only with its front panel controls. The hard-copy device manufacturer usually specifies an address that forces the oscilloscope into listening mode, and you can select this as well as the other necessary settings using the same menus. See Chapter 6, "Document Your Work," of the *Operator's Manual*.

Use the following schemes for driving hard-copy devices by remote control using GPIB.

READ DATA BY CONTROLLER

The controller reads the data into internal memory, then sends them to the printer. You can arrange this with simple high-level GPIB function calls. The controller stores the full set of printer instructions and afterwards sends them to the graphics device. This method is the most straightforward way to transfer screen contents, but requires a large amount of buffer storage:

```
CMD$ = "SCDP"
CALL IBWRT(SCOPE%,CMD$)
FILE$="PRINT.DAT"
CALL IBRDF(SCOPE%,FILE$)
CALL IBWRTF(PRINTER%,FILE$)
```

SEND DATA TO BOTH

WavePro sends data to both controller and printer. The oscilloscope puts the printer instructions onto the bus. The data is directly put out and saved in scratch memory in the controller. The contents of the scratch file can be deleted later:

Stage 1: Controller talker, WavePro listener.

1. Issue the screen dump command

   ```
   CMD1$="? @$": CALL IBCMD(BRD0%,CMD1$)
   CMD$="SCDP": CALL IBWRT(BRD0%,CMD$)
   ```

Stage 2: WavePro talker, controller and printer listeners.

2. Print data while storing data in scratch file SCRATCH.DAT with the commands

```
CMD2$="? D%": CALL IBCMD(BRD0%,CMD2$)
FILE$="SCRATCH.DAT": CALL IBRDF(BRD0%,FILE$)
```

TALK DIRECTLY TO PRINTER

a. The controller goes into a standby state.

b. WavePro becomes a talker and sends data directly to the printer.

c. The controller goes into standby and resumes GPIB operations once the data have been printed, i.e. when an EOI is detected:

Stage 1: Controller talker, WavePro listener.

1. Issue the screen dump command

```
CMD1$="?_@$": CALL IBCMD(BRD0%,CMD1$)
CMD$="SCDP": CALL IBWRT(BRD0%,CMD$)
```

Stage 2: WavePro talker, printer or plotter listener.

2. Put controller in standby

```
CMD2$="?_D%": CALL IBCMD(BRD0%,CMD2$)
V%=1: CALL IBGTS(BRD0%,V%)
```

In the second and third schemes presented above, board-level GPIB function calls are used. It is assumed that the controller (board), WavePro and the printer are respectively located at addresses 0, 4, and 5.

The listener and talker addresses for the controller, WavePro, and printer are as follows:

LOGIC DEVICE	LISTENER ADDRESS	TALKER ADDRESS
Controller	32 (ASCII<space>)	64 (ASCII @)
WavePro	32+4=36 (ASCII $)	64+4=68 (ASCII D)
Printer	32+5=37 (ASCII %)	64+5=69 (ASCII E)

The characters "?" and "_" appearing in the command strings stand for unlisten and untalk respectively. They are used to set the devices to a "known" state.

To shorten the size of the program examples, device talking and listening initialization instructions have been grouped into character chains. They are:

`CMD1$ = "?_@$"` Unlisten, Untalk, Controller talker, WavePro listener

`CMD2$ = "?_ D"` Unlisten, Untalk, Controller listener, WavePro talker

CHAPTER THREE: *Control by RS-232*

In this chapter, see how to

Control WavePro by RS-232-C

Simulate GPIB messages using RS-232-C

Communicate through the RS-232-C Port

Your WavePro oscilloscope can also be controlled remotely through the RS-232-C port, which supports the transfer of all commands for its operation. Nevertheless, RS-232 waveform transfer is only possible in HEX mode, using the default value for COMM_FORMAT, and with the syntax of the response to WF? identical to that for GPIB.

RS-232-C connector pin assignments for connecting WavePro to an external controller are given in Chapter 12, "Use WavePro with PC," of the WavePro *Operator's Manual*.

The RS-232-C port is full-duplex configured. This means that both sides — WavePro oscilloscope and external controller — can send and receive messages at the same time. However, the oscilloscope stops outputting when it receives a new command.

You should transmit long messages to the oscilloscope while it is in a trigger mode, and not while an acquisition is in progress. This is especially important when sending waveforms or front panel setups.

Characters that cannot be printed in ASCII are here represented by their mnemonics. For example:

<LF>	ASCII line feed character whose decimal value is 10.
<BS>	ASCII backspace character whose decimal value is 8.
CTRL_U	The control key and the U key are pressed simultaneously.

Set RS-232-C behavior according to your needs. In addition to the basic setup on the front panel menu, there are "immediate commands," as well as the special command COMM_RS232 for this. Immediate commands consist of the ASCII ESCape character <ESC> (whose decimal value is 27), followed by another character. These commands are interpreted as soon as the second character has been received.

You can have the serial port echo the received characters. This is useful when the oscilloscope is connected to a terminal. Echoing can be turned on or off by sending the two-character sequence <ESC>] or <ESC>[. Echoing is on by default, but the host must not echo characters received from the oscilloscope.

HANDSHAKE CONTROL

When the oscilloscope intake buffer becomes nearly full, the instrument sends a handshake signal to the host telling it to stop transmitting. When this buffer has enough room to receive more characters, another handshake signal is sent. These signals are either the CTRL-S (or <XOFF>) and CTRL-Q (<XON>) characters, or a signal level on the RTS line. They are selected by sending the two-character sequence <ESC>) for XON/XOFF handshake (the default), or <ESC> for the RTS handshake.

You can control the flow of characters coming from the oscilloscope by either a signal level on the CTS line or the <XON>/<XOFF> pair of characters.

NOTE: The RS-232-C baud rate, parity, character length, and number of stop bits are among the parameters saved or recalled by the front panel SAVE or RECALL buttons, and by the remote commands *SAV, *RCL, or PANEL_SETUP. When recalling by remote, ensure that these parameters are set at the same value on both controller and oscilloscope. Otherwise, the host may no longer be able to communicate with the oscilloscope and a manual reconfiguration would be necessary.

EDITING FEATURES

When the oscilloscope is directly connected to a terminal, the following will make correction of typing errors easier:

<BS> or <DELETE>	Delete the last character.
CTRL_U	Delete the last line.

MESSAGE TERMINATORS

Message terminators are markers that indicate to the receiver that a message has been completed. The Program Message Terminator is a character you could select when you input to the oscilloscope. Choose a character never used for anything else, using the command COMM_RS232 and the keyword EI. The default Program Message Terminator is the ASCII character <CR>, whose decimal value is 13.

The oscilloscope appends a Response Message Terminator to the end of each of its responses. This is a string, similar to a computer prompt, which you also choose. This string must not be empty. The default Response Message Terminator is \n\r, which is the same as <LF><CR>.

Example: COMM_RS232 EI,3
This command informs the oscilloscope that each message it receives will be terminated with the ASCII character <ETX>, whose decimal value is 3.

Example: COMM_RS232 EO,"\r\nEND\r\n"
This command indicates to the oscilloscope that it must append the string "\r\nEND\r\n" to each response.

After you make these settings, a host command will look like this:

```
TDIV?<ETX>
```

And the oscilloscope will respond with:

```
TDIV 1.S
END
```

TIP: After it sends a COMM_RS232 command, the host must wait for the oscilloscope to change its behavior before it can send a command in the new mode. Ensure this by including a query on the line that contains the COMM_RS232 command (for example, COMM_RS232 EI,3;*STB?) and waiting until the response is received.

SRQ MESSAGE

Each time the Master Summary Status (MSS) bit of the STatus Byte (STB) is set, the SRQ message (a string of characters) is sent to the host to indicate that the oscilloscope requests service. The RS-232-C SRQ message has the same meaning as the GPIB SRQ message. If the string is empty, no message will be sent. This is the default setting. Note that no response message terminator is added at the end of the SRQ message.

Example: `COMM_RS232 SRQ, "\r\n\nSRQ\r\n\a"`

When the MSS bit is set, the oscilloscope will send a <CR> followed by two <LF>SRQ, and a <CR> followed by a <LF>. The buzzer will sound.

LONG LINE SPLITTING

Line splitting is a feature provided for hosts that cannot accept lines with more than a certain number of characters. The oscilloscope may be configured to split responses into many lines. This feature is very useful for waveform or front panel setup transfers although it is applicable to all response messages. Two parameters control this feature:

Line Separator:Off
- messages will not be split into lines.

<CR>,<LF> or <CR><LF>
- possible line terminators.

Line Length:the maximum number of characters to a line.

Example: `COMM_RS232 LS,LF,LL,40`

The line separator is the ASCII character <LF>, the line is a maximum of 40 characters long (excluding the line separator).

If the oscilloscope receives the command PNSU?, it may answer with:

```
PNSU#9000001496
AAAA5555000655AA4030005800190000000000001
00000000000000000000000000C1B0100580000
00000000000000000000000000000000000000000
...
```

REMARKS

Long commands sent to the oscilloscope may not be split into lines. If a command sent to the oscilloscope is the response to a previous query, the line-split characters (<LF>, <CR>) must be removed. This also applies to line-split characters inside strings sent to the oscilloscope.

However, hex-ASCII data sent to the oscilloscope may contain line-split characters. If you wish to use line splitting, ensure that neither the input message terminator characters nor the line-split characters occur in the data.

Simulate GPIB Messages

Use these RS-232-C commands to simulate GPIB 488.1 messages:

RS232 COMMAND	GPIB MESSAGE	EFFECT AND EQUIVALENCE
<ESC>C or <ESC>c	Device Clear (DCL)	Clears the input and output buffers. This command has the same meaning as the GPIB DCL or SDC interface messages.
<ESC>R or <ESC>r	Remote Enable (REN)	Places the oscilloscope in remote mode. This command's function is the same as the GPIB command asserting the REN line and setting the oscilloscope to listener.
<ESC>L or <ESC>l	Go to Local (GTL)	Places the oscilloscope in local mode. The command clears local lockout (see below). It has the same function as GPIB's setting the REN line to false.
<ESC>F or <ESC>f	Local Lockout (LLO)	Disables the front panel "LOCAL" button immediately if the oscilloscope is already in remote mode or, if not, when the oscilloscope is next set to remote. This Local Lockout (see Chapter 2, "Control by GPIB") can be cancelled only with the <ESC>L command. <ESC>F has the same meaning as GPIB's LLO interface message.
<ESC>T or <ESC>t	Group Execute Trigger (GET)	Rearms the oscilloscope while it is in the STOP mode, but only while the oscilloscope is in remote mode. This command has the same meaning as the `*TRG` command and GPIB's GET interface message.

CHAPTER FOUR: *Understand and Manage Waveforms*

In this chapter, see how to

Structure Waveforms

Inspect waveform contents

Transfer waveforms rapidly

Know Your Waveform

A waveform can be said to have two main parts. The first is its basic data array: raw data values from the oscilloscope's ADCs (Analog-to-Digital Converters) obtained in the waveform's capture. The second is the description that accompanies this raw data: the vertical and horizontal scale or time of day, for example, necessary for a full understanding of the information contained in the waveform.

You can access this information by remote control using the `INSPECT?` query (see page 36), which interprets it in an easily understood ASCII text form. And you can rapidly transfer the information using the `WAVEFORM?` query (see page 37). Or write it back into the oscilloscope with the `WAVEFORM` command (page 42).

Your WavePro DSO contains a data structure, or template (see Appendix II), which provides a detailed description of how waveform information is organized. Although a sample template is provided with this manual, we suggest you use the `TEMPLATE?` query to access the *WavePro* DSO template in the oscilloscope itself (the template may change as your oscilloscope's firmware is enhanced).

You can also store waveforms in preformatted ASCII output, for popular spreadsheet and math processing packages, using the STORE and STORE_SETUP commands. Also see Chapter 12, "Use *WavePro* DSO with PC," of the *Operator's Manual*.

LOGICAL DATA BLOCKS

Each of your waveforms will normally contain at least a waveform descriptor and data array block. However, other blocks may also be present in more complex waveforms.

Waveform Descriptor block (WAVEDESC): This includes all the information necessary to reconstitute the display of the waveform from the data, including: hardware settings at the time of acquisition, the exact time of the event, kinds of processing performed, your oscilloscope name and serial number, the encoding format used for the data blocks, and miscellaneous constants.

Optional User-provided Text block (USERTEXT): Use the `WFTX` command to put a title or description of a waveform into this block, and the `WFTX?` query for an alternative way to read it. This text block can hold up to 160 characters. Display them as four lines of 40 characters by selecting "**Text & Times**" from the status menu, using *WavePro* DSO front panel controls (see the *Operator's Manual*).

Sequence Acquisition Times block (TRIGTIME): This is needed for sequence acquisitions to record the exact timing information for each segment. It contains the time of each trigger relative to the trigger of the first segment, as well as the time of the first data point of each segment relative to its trigger.

Random Interleaved Sampling times block (RISTIME): This is required for RIS acquisitions to record the exact timing information for each segment.

First Data Array block (SIMPLE or DATA_ARRAY_1): This is the basic integer data of the waveform. It can be raw or corrected ADC data or the integer result of waveform processing.

Second Data Array block (DATA_ARRAY_2): This is a second data array, needed to hold the results of processing functions such as Extrema or FFT math functions:

	EXTREMA	FFT
DATA_ARRAY_1	Roof trace	Real part
DATA_ARRAY_2	Floor trace	Imaginary part

NOTE: The WavePro DSO template also describes an array named DUAL. But this is simply a way to allow the INSPECT? command to examine the two data arrays together.

INSPECT WAVEFORM CONTENTS

Use the INSPECT? query to examine the contents of your waveform. You can use it on both of the main waveform parts. Its most basic form is: INSPECT? "name", the template giving you the name of a descriptor item or data block. The answer is returned as a single string, but may cover many lines. Some typical dialogue follows:

Question `C1:INSPECT? "VERTICAL_OFFSET"`

Response `C1:INSP "VERTICAL_OFFSET: 1.5625e-03"`

Question `C1:INSPECT? "TRIGGER_TIME"`

Response `C1:INSP "TRIGGER_TIME: Date = FEB 17, 1994, Time = 4: 4:29.5580"`

You can also use INSPECT? to provide a readable translation of the full waveform descriptor block using INSPECT? "WAVEDESC". Again, the template will give you the details for interpretation of each of the parameters. Use, too, INSPECT? "SIMPLE" to examine the measured data values of a waveform. For an acquisition with 52 points, for example:

```
INSPECT? "SIMPLE"
C1:INSP "
 0.0005225  0.0006475 -0.00029   -0.000915   2.25001E-0 0.000835
 0.0001475 -0.0013525 -0.00204   -4E-05      0.0011475  0.0011475
-0.000915  -0.00179   -0.0002275  0.0011475  0.001085  -0.00079
-0.00179   -0.0002275  0.00071    0.00096   -0.0003525 -0.00104
 0.0002725  0.0007725  0.00071   -0.0003525 -0.00129   -0.0002275
 0.0005225  0.00046   -0.00104   -0.00154    0.0005225  0.0012725
 0.001335  -0.0009775 -0.001915  -0.000165   0.0012725  0.00096
-0.000665  -0.001665  -0.0001025  0.0010225  0.00096   -0.0003525
-0.000915   8.50001E-0 0.000835   0.0005225
"
```

The numbers in the table above are the fully converted measurements in volts. When the data block contains thousands of items the string will contain a great many lines.

Depending on the application, you may prefer the data in its raw form, with either a BYTE (8 bits) or a WORD (16 bits) for each data value. In that case, use the relations `INSPECT? "SIMPLE",BYTE` with `WAVEFORM?`. The examination of data values for waveforms with two data arrays can be performed as follows:

`INSPECT? "DUAL"`	to get pairs of data values on a single line
`INSPECT? "DATA_ARRAY_1"`	to get the values of the first data array
`INSPECT? "DATA_ARRAY_2"`	to get the values of the second data array.

`INSPECT?` has its limitations; it is useful, but also wordy. As a query only, `INSPECT?` cannot be used to send a waveform back to the oscilloscope. If you want to do this and you want the information quickly, you should instead use `WAVEFORM`. With `WAVEFORM_SETUP` it is possible to examine just a part of the waveform or a sparsed form of it. See the following pages.

If you're a BASIC user you might also find it convenient to use `INSPECT?` and `WAVEFORM?` together to construct files containing a version of the waveform descriptor that both you and BASIC can read. Using a stored waveform, this can be done in a format suitable for retransfer to *WavePro* DSO with `MC:INSPECT? "WAVEDESC";WAVEFORM?`, and then placing the response directly into a disk file.

USE THE WAVEFORM QUERY

Use the `WAVEFORM?` query to transfer waveform data in block formats defined by the IEEE-488.2 standard. You can then download the response back to your *WavePro* DSO by using the `WAVEFORM` command. All your waveform's logical blocks can be read with the query `C1:WAVEFORM?` Completeness, as well as good use of time and space are the advantages of this approach when you have to read many waveforms with the same acquisition conditions, or when you are interested only in large amounts of raw integer data. Moreover, you can choose any single block for reading with a query such as `C1:WAVEFORM? DAT1`. See Part Two for the various block names.

You can place the binary response to a query of the form `C1:WAVEFORM?` or `C1:WAVEFORM? ALL` in a disk file, then dump it using the GPIB bus. Do this with default settings to show the hexadecimal and ASCII form, as on the following page.

NOTE: A waveform query response can easily be a block containing over 16 million bytes if it is in binary format, and twice as much if the HEX option is used.

BYTE OFFSET NUMBER		BINARY CONTENTS IN HEXADECIMAL	ASCII TRANSLATION (.... = UNINTERESTING)
0		43 31 3A 57 46 20 41 4C 4C 2C 23 39 30 30 30 30	C1:WFALL,#90000
16		30 30 34 35 30	00450
	0	57 41 56 45 44 45 53 43 00 00 00	WAVEDESC...
32	11	00 00 00 00 00 4C 45 43 52 4F 59 5F 32 5F 32 00	.....LECROY_2_2.
48	27	00 00 00 00 00 00 01 00 00 00 00 01 5A 00 00 00	..............Z...
64	43	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
80	59	00 00 00 00 68 00 00 00 00 00 00 00 00 00 00 00	....h...........
96	75	00 4C 45 43 52 4F 59 39 33 37 34 4C 00 00 00 00	.LECROYLT344....
112	91	00 37 84 09 40 00 00 00 00 00 00 00 00 00 00 00	.7..@...........
128	107	00 00 00 00 00 00 00 00 34 00 00 00 34 00 00 00	........4...4...
144	123	32 00 00 00 00 00 00 00 33 00 00 00 00 00 00 00	2.......3.......
160	139	01 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00	....
176	155	00 34 83 12 6F 3A 0D 8E C9 46 FE 00 00 C7 00 00	.4..o:...F......
192	171	00 00 08 00 01 32 2B CC 77 BE 6B A4 BB 51 A0 69	.....2+.w.k..Q.i
208	187	BB BE 6A D7 F2 A0 00 00 00 56 00 00 00 00 00 00	..j......V......
224	203	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
240	219	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
256	235	00 00 00 00 00 00 00 00 00 00 53 00 00 00 00 00	..........S.....
272	251	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
288	267	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	....
304	283	00 00 00 00 00 00 00 00 00 00 00 00 00 00 40 3B 00	..............@;.
320	299	00 00 00 00 00 17 0A 05 02 07 C8 00 00 00 00 00	..........H.....
336	315	00 00 00 00 00 00 00 00 01 00 0E 00 04 3F 80 00	.............?..
352	331	00 00 0A 00 00 3F 80 00 00 3A 0D 8E C9 00 00	.....?...:....
367	0	11	
368	1	00 13 00 04 00 FA 00 09 00 16 00 0B 00 F3 00 E8	
384	17	00 08 00 1B 00 1B 00 FA 00 EC 00 05 00 1B 00 1A	
400	33	00 FC 00 EC 00 05 00 14 00 18 00 03 00 F8 00 0D	
416	49	00 15 00 14 00 03 00 F4 00 05 00 11 00 10 00 F8	
432	65	00 F0 00 11 00 1D 00 1E 00 F9 00 EA 00 06 00 1D	
448	81	00 18 00 FE 00 EE 00 07 00 19 00 18 00 03 00 FA	
464	97	00 0A 00 16 00 11 00	
471	(Terminator)	0A	

Above: To illustrate the contents of the logical blocks, the relevant parts have been separated. To make counting easier, the corresponding Byte Offset numbering has been restarted each time a new block begins. The ASCII translation, only part of which is shown, has been similarly split and highlighted, showing how its parts correspond to the binary contents.

On the facing page... The first 10 bytes translate into ASCII and resemble the simple beginning of a query response. These are followed by the string `#9000000450`, the beginning of a binary block in which nine ASCII integers are used to give the length of the block (450 bytes). The waveform itself starts immediately after this, at Byte 21. The very first byte is 0, as it is for the first byte in each block.

The first object is a DESCRIPTOR_NAME, a string of 16 characters with the value `WAVEDESC`.

Then, 16 bytes after the beginning of the descriptor, at Byte 37, we find the beginning of the next string: the TEMPLATE_NAME with the value `LECROY_2_2`.

Several other parameters follow. The INSTRUMENT_NAME, `LECROYLT344`, 76 bytes from the descriptor start (Byte 97), is easily recognizable. On the preceding line, 38 bytes after the descriptor (Byte 59), a four-byte integer gives the length of the descriptor: WAVE_DESCRIPTOR = 00 00 01 5A (hex) = 346.

At 60 bytes from the descriptor start (Byte 81) we find another four-byte integer giving the length of the data array: WAVE_ARRAY_1 = 00 00 00 68 (hex) = 104.

And at 116 bytes after the descriptor (Byte 137), yet another four-byte integer gives the number of data points: WAVE_ARRAY_COUNT = 00 0000 34 (hex) = 52.

Now we know that the data will start at 346 bytes from the descriptor's beginning (Byte 367), and that each of the 52 data points will be represented by two bytes. The waveform has a total length of 346 + 104, which is the same as the ASCII string indicated at the beginning of the block. The final 0A at Byte 471 is the NL character associated with the GPIB message terminator <NL><EOI>.

As the example was taken using an oscilloscope with an eight-bit ADC, we see the eight bits followed by a 0 byte for each data point. However, for many other kinds of waveform this second byte will not be zero and will contain significant information. The data is coded in signed form (two's complement) with values ranging from –32768 = 8000 (hex) to 32767 = 7FFF (hex). If we had chosen to use the BYTE option for the data format, the values would have been signed integers in the range –128 = 80 (hex) to 127 = 7F (hex). The ADC values are mapped to the display grid in the following way:

0 is located on the grid's center axis

127 (BYTE format) or 32767 (WORD format) is located at the top of the grid

–128 (BYTE format) or –32768 (WORD format) is located at the bottom of the grid.

INTERPRET VERTICAL DATA

Knowing now how to decipher the data, you may wish to convert it to the appropriate measured values. The vertical reading for each data point depends on the vertical gain and the vertical offset given in the descriptor. For acquisition waveforms, this corresponds to the volts/ div and voltage offset selected after conversion for the data representation being used. The template tells us that the vertical gain and offset can be found at Bytes 156 and 160 and that they are stored as floating point numbers in the IEEE 32-bit format. An ASCII string giving the vertical unit is to be found in VERTUNIT, Byte 196. The vertical value is given by the relationship: value = VERTICAL_GAIN × data – VERTICAL_OFFSET, where:

VERTICAL_GAIN	2.44141e–07 from the floating point number 3483 126f at Byte 177
VERTICAL_OFFSET	0.00054 from the floating point number 3A0D 8EC9 at Byte 181
VERTICAL_UNIT	V = volts from the string 5600 ... at Byte 217

Therefore:

since data[4] = FA00 = 64000 from the hexadecimal word FA00 at byte 371. Overflows the maximum. 16 bit value of 32767, so must be a negative value. Using the two's complement conversion $64000–2^{16} = –1536$

value[4] = –0.000915 V as stated in the inspect command.

If the computer or the software available is not able to understand the IEEE floating point values, use the description in the template.

The data values in a waveform may not all correspond to measured points. FIRST_VALID_PNT and LAST_VALID_PNT give the necessary information. The descriptor also records the SPARSING_FACTOR, the FIRST_POINT, and the SEGMENT_INDEX to aid interpretation if the options of the `WAVEFORM_SETUP` command have been used.

For sequence acquisitions, the data values for each segment are given in their normal order and the segments are read out one after the other. The important descriptor parameters are the WAVE_ARRAY_COUNT and the SUBARRAY_COUNT, giving the total number of points and the number of segments.

For waveforms such as the extrema and the complex FFT there will be two arrays — one after the other — for the two of the result.

CALCULATE A DATA POINT'S HORIZONTAL POSITION

Each vertical data value has a corresponding horizontal position, usually measured in time or frequency units. The calculation of this position depends on the type of waveform. Each data value has a position, i, in the original waveform, with i = 0 corresponding to the first data point acquired. The descriptor parameter HORUNIT gives a string with the name of the horizontal unit.

Single Sweep waveforms: $x[i] = HORIZ_INTERVAL \times i + HORIZ_OFFSET$. For acquisition waveforms this time is from the trigger to the data point in question. It will be different from acquisition to acquisition since the HORIZ_OFFSET is measured for each trigger. In the case of the data shown above this means:

HORIZ_INTERVAL = 1e–08 from the floating point number 322b cc77 at Byte 194

HORIZ_OFFSET = –5.149e–08 from the double precision floating point number be6b a4bb 51a0 69bb at Byte 198

HORUNIT = S = seconds from the string 5300 ... at Byte 262

This gives: x[0] = –5.149e–08 S
x[1] = –4.149e–08 S.

Sequence waveforms: are really many independent acquisitions, so each segment will have its own horizontal offset. These can be found in the TRIGTIME array.

For the nth segment:

$$x[i,n] = \text{HORIZ_INTERVAL} \times i + \text{TRIGGER_OFFSET}[n].$$

The TRIGTIME array can contain up to 200 segments of timing information with two eight-byte double precision floating point numbers for each segment.

RIS (Random Interleaved Sampling) waveforms: are composed of many acquisitions interleaved together. The descriptor parameter, RIS_SWEEPS gives the number of acquisitions. The i^{th} point will belong to the m^{th} segment where:

$m = i$ modulo (RIS_SWEEPS) will have a value between 0 and RIS_SWEEPS -1.

Then with: $j = i - m$

$$x[i] = x[j,m] = \text{HORIZ_INTERVAL} \times j + \text{RIS_OFFSET}[m],$$

where the RIS_OFFSETs can be found in the RISTIME array. There can be up to 100 eight-byte double precision floating point numbers in this block. The instrument tries to get segments with times such that: $\text{RIS_OFFSET}[i] \cong \text{PIXEL_OFFSET} + (i - 0.5) \times \text{HORIZ_INTERVAL}$.

Thus, taking as an example a RIS with RIS_SWEEPS = 10, HORIZ_INTERVAL = 1 ns, and PIXEL_OFFSET = 0.0, we might find for a particular event that:

RIS_OFFSET[0] = –0.5 ns	RIS_OFFSET[1] = 0.4 ns
RIS_OFFSET[2] = 1.6 ns	RIS_OFFSET[3] = 2.6 ns
RIS_OFFSET[4] = 3.4 ns	RIS_OFFSET[5] = 4.5 ns
RIS_OFFSET[6] = 5.6 ns	RIS_OFFSET[7] = 6.4 ns
RIS_OFFSET[8] = 7.6 ns	RIS_OFFSET[9] = 8.5 ns

and therefore:

x[0] = RIS_OFFSET[0] = –0.5 ns

x[1] = RIS_OFFSET[1] = 0.4 ns

...

x[9] = RIS_OFFSET[9] = 8.5 ns

x[10] = 1 ns × 10 + (–0.5) = 9.5 ns

x[11] = 1 ns × 10 + 0.4 = 10.4 ns

...

x[19] = 1 ns ´ 10 + 8.5 = 18.5 ns

x[20] = 1 ns ´ 20 + (–0.5) = 19.5 ns.

...

USE THE WAVEFORM COMMAND

Waveforms you read with the WAVEFORM? query (page 37) can be sent back into your *WavePro* DSO using WAVEFORM and related commands. Since the descriptor contains all of the necessary information, you need not be concerned with any of the communication format parameters. The oscilloscope will learn all it needs to know from the waveform.

TIP: Because waveforms can only be sent back to WavePro DSO memory traces (M1, M2, M3, M4), consider removing or changing the prefix (C1 or CHANNEL_1) in the response to the WF? query. See Part Two for examples.

To ensure that the descriptor is coherent, however, when you synthesize waveforms for display or comparison read out a waveform of the appropriate size and then replace the data with the desired values.

Here are among the many ways to use WAVEFORM and its related commands to simplify or speed up work:

Partial Waveform Readout: Use WAVEFORM_SETUP to specify a short part of a waveform for readout, as well as to select a sparsing factor for reading every nth data point only.

Byte Swapping: The COMM_ORDER command allows you to swap two bytes of data presented in 16-bit word format, in the descriptor or in the data/time arrays, when sending the data via GPIB or RS-232-C ports. Depending on the computer system used, this will allow easier data interpretation. For Intel-based computers, you should send the data with the LSB first; the command should be CORD LO. For Motorola-based computers, send the data with the MSB first (CORD HI) — the default at power-up.

NOTE: Data written to WavePro DSO hard disk or floppy drive, or to the optional PC memory card drive, will always remain in LSB first, the default DOS format. Thus you cannot use the CORD command in these cases, as it is only for data sent via the GPIB and RS-232-C ports.

Data Length, Block Format, and Encoding: COMM_FORMAT gives you control over these parameters. If you do not need the extra precision of the lower order byte of the standard data value, the BYTE option will enable you to save by a factor of two the amount of data transferred or stored. If the computer you are using cannot read binary data, the HEX option allows a response form in which the value of each byte is given by a pair of hexadecimal digits.

Data-Only Transfers: COMM_HEADER OFF enables a response to WF? DAT1 with data only (the C1:WF DAT1 will disappear). If you have also specified COMM_FORMAT OFF,BYTE,BIN, the response will be data bytes only (the #90000nnnnn will disappear — see page 38).

Formatting for RS-232-C Users: The COMM_RS232 command can assist by splitting the very long WF? response into individual lines.

Transfer Waveforms at High Speed

You must take several important factors into account if you wish to achieve maximum, continuous data transfer rates from your *WavePro* DSO to the external controller. The single most important of these is to limit the amount of work done in the computer. This means that you should avoid writing data to disk wherever possible, minimize operations such as per-data-point computations, and reduce the number of calls to the I/O system. To do this, you can try the following:

Reduce the number of points to be transferred and the number of data bytes per point. The pulse parameter capability and the processing functions can save a great deal of computing and a lot of data transfer time if employed creatively.

Attempt to overlap waveform acquisition with waveform transfer. The oscilloscope is capable of transferring an already acquired or processed waveform after a new acquisition has been started. The total time that *WavePro* DSO takes to acquire events will be considerably increased if it is obliged to wait for triggers (live time).

Minimize the number of waveform transfers by using Sequence mode to accumulate many triggers for each transfer. This is preferable to using `WAVEFORM_SETUP` to reduce the number of data points for transfer. It also significantly reduces oscilloscope transfer overhead. For example, you could use `ARM; WAIT;C1:WF?` (wait for the event, transfer the data, and then start a new acquisition). You could also "loop" this line in the program as soon as it has finished reading the waveform.

CHAPTER FIVE: *Check Waveform Status*

In this chapter, see how to

Use status registers

Use Status Registers

A wide range of status registers allows you to quickly determine *WavePro* DSO internal processing status at any time. These registers and the oscilloscope's status reporting system, which group related functions together, are designed to comply with IEEE 488.2 recommendations. Some, such as the Status Byte Register (STB) or the Standard Event Status Register (ESR), are required by the IEEE 488.2 Standard. Others are device specific, including the Command Error Register (CMR) and Execution Error Register (EXR). Those commands associated with IEEE 488.2 mandatory status registers are preceded by an asterisk (*).

OVERVIEW

The Standard Event Status Bit (ESB) and the Internal Status Change Bit (INB) in the STB are summary bits of the ESR and the Internal State Change Register (INR). The Message Available Bit (MAV) is set whenever there are data bytes in the output queue. The Value Adapted Bit (VAB) indicates that a parameter value was adapted during a previous command interpretation. For example, if the command `TDIV 2.5 US` was received, the timebase would be set to 2 ms/div along with the VAB bit.

The Master Summary Status bit (MSS) indicates a request for service from the oscilloscope. You can only set the MSS bit if you have enabled one or more of the other STB bits with the Service Request Enable Register (SRE).

All Enable registers (SRE, ESE and INE) are used to generate a bit-wise AND with their associated status registers. The logical OR of this operation is reported to the STB register. At power-on, all Enable registers are zero, inhibiting any reporting to the STB.

The ESR primarily summarizes errors, whereas the INR reports internal changes to the instrument. Additional details of errors reported by ESR can be obtained with the queries `CMR?`, `DDR?`, `EXR?` and `URR?`.

The register structure contains one additional register, not shown on the next page (Fig.1). This is the Parallel Poll Enable Register (PRE), which behaves exactly like the SRE, but sets the "ist" bit used in the Parallel Poll. Read the "ist" bit with the `*IST?` query.

Example: If you were to send the erroneous command TRIG_MAKE SINGLE to your *WavePro* DSO, the oscilloscope would reject it and set the Command Error Register (CMR) to the value 1 (unrecognized command/query header). The non-zero value of CMR would be reported to Bit 5 of the Standard Event Status Register (ESR), which is then set. Nothing further would occur unless the corresponding Bit 5 of the Standard Event Status Enable Register (ESE) was set with the command `*ESE 32`, enabling Bit 5 of ESR to be set for reporting to the summary bit ESB of the STB.

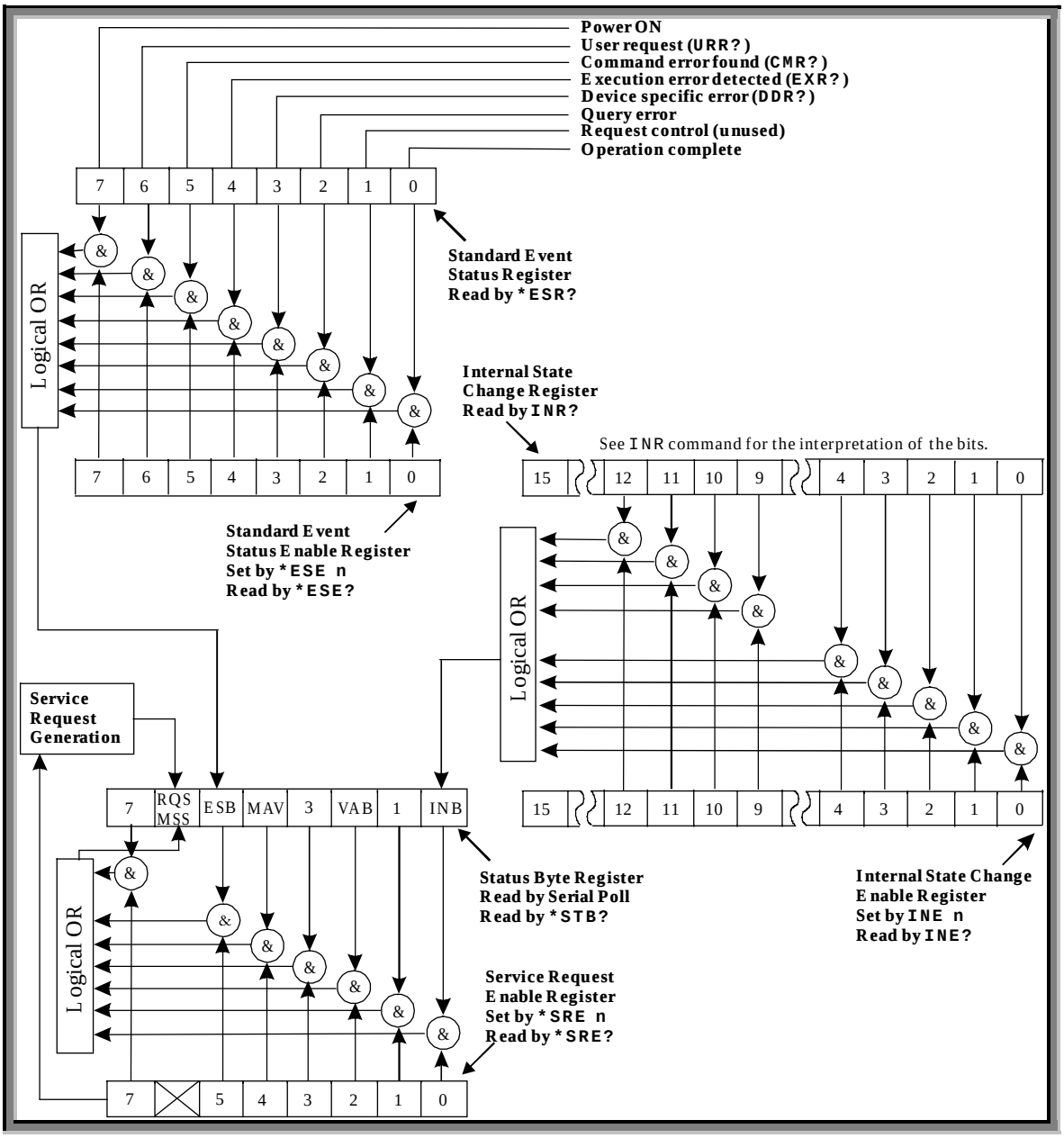


Figure 1.Status Register Structure

If you enabled the setting of the ESB summary bit in STB, again nothing would occur unless you enabled further reporting by setting the corresponding bit in the SRE register with the command *SRE 32. The generation of a non-zero value of CMR would ripple through to MSS, generating a Service Request (SRQ).

You can read the value of CMR and simultaneously reset to zero at any time with the command CMR?. The occurrence of a command error can also be detected by analyzing the response to *ESR?. However, if you must survey several types of potential errors, it is usually far more efficient to enable propagation of the errors of interest into the STB with the enable registers ESE and INE.

To summarize: a command error (CMR) sets Bit 5 of ESR if

a. Bit 5 of ESE is set, ESB of STB is also set, or

b. Bit 5 of SRE is set, MSS/RQS (Request for Service) of STB is also set and a Service Request is generated.

STATUS BYTE REGISTER (STB)

STB is the *WavePro* DSO central reporting structure. It is made up of eight single-bit summary messages, three of which are unused, that reflect the current status of the oscilloscope's associated data structures:

Bit 0 is the INB summary bit of the Internal State Change Register. It is set if any INR bits are set, provided they are enabled by the corresponding bit of the INE register.

Bit 2 is the VAB bit, indicating that a parameter value was adapted during a previous command interpretation.

Bit 4 is the MAV bit, indicating that the interface output queue is not empty.

Bit 5 is the summary bit ESB of the ESR. It is set if any of the bits of the ESR are set, provided they are enabled by the corresponding bit of the ESE register.

Bit 6 is either the MSS or RQS bit.

You can read the STB using the *STB? query. It reads and clears the STB, in which case Bit 6 is the MSS bit, and it indicates whether the oscilloscope has any reason to request service. The response to the query represents the binary weighted sum of the register bits. The register is cleared by *STB?, ALST?, *CLS, or with *WavePro* DSO powering up.

Another way to read the STB is using the serial poll (see Chapter 2). In this case, Bit 6 is the RQS bit, indicating that the instrument has activated the SRQ line on the GPIB. The serial poll clears only the RQS bit. And the STB's MSS bit, and any other bits which caused MSS to be set, will remain set after the poll. These bits must be reset.

STANDARD EVENT STATUS REGISTER (ESR)

ESR is a 16-bit register reflecting the occurrence of events. ESR bit assignments have been standardized by IEEE 488.2. Only the lower eight bits are currently in use.

Read ESR using *ESR?. The response is the binary weighted sum of the register bits. The register is cleared with *ESR? or ALST?, or with *CLS or powering on the scope.

Example: The response message `*ESR 160` tells you that a command error occurred and that the ESR is being read for the first time after power-on. The value 160 can be broken down into 128 (Bit 7) plus 32 (bit 5). See the table with the `ESR` command description in Part Two for the conditions corresponding to the bits set.

The Power ON bit appears only on the first `*ESR?` query after power-on, as the query clears the register. You can determine this type of command error by reading the CMR with `CMR?`. It is not necessary that you read, or simultaneously clear, this register in order to set the CMR bit in the ESR on the next command error.

STANDARD EVENT STATUS ENABLE REGISTER (ESE)

This register allows you to report one or more events in the ESR to the ESB summary bit in the STB.

Modify ESE with `*ESE` and clear it with `*ESE 0`, or with power-on. Read it with `*ESE?`.

Example: Use `*ESE 4` to set bit 2 (binary 4) of the ESE Register, and enable query errors to be reported.

SERVICE REQUEST ENABLE REGISTER (SRE)

SRE specifies which Status Byte Register summary bit or bits will bring about a service request. This register consists of eight bits. Setting a bit allows the summary bit located at the same bit position in the SBR to generate a service request, provided that the associated event becomes true. Bit 6 (MSS) cannot be set and is always reported as zero in response to `*SRE?`.

Modify SRE with `*SRE` and clear it with `*SRE 0`, or with power-on. Read it using `*SRE?`.

PARALLEL POLL ENABLE REGISTER (PRE)

This specifies which Status Byte Register summary bit or bits will set the "ist" individual local message. PRE is similar to SRE, but is used to set the parallel poll "ist" bit rather than MSS.

The value of the "ist" may also be read without a Parallel Poll via the query `*IST?`. The response indicates whether or not the "ist" message has been set (values are 1 or 0).

Modify PRE with `*PRE` and clear it with `*PRE 0`, or with power-on. Read this register with `*PRE?`.

Example: Use `*PRE 5` to set the register's bits 2 and 0 (decimal 4 and 1).

INTERNAL STATE CHANGE STATUS REGISTER (INR)

INR reports the completion of a number of internal operations (the events tracked by this 16-bit-wide register are listed with the `INR?` description in Part Two).

Read the register using `INR?`. The response is the binary weighted sum of the register bits. Clear the register with `INR?` or `ALST?`, a `*CLS` command, or with power-on.

INTERNAL STATE CHANGE ENABLE REGISTER (INE)

INE allows one or more events in the Internal State Change Status Register to be reported to the INB summary bit in the STB.

Modify INE with `INE` and clear it with `INE 0`, or after power-on. Read it with `INE?`.

COMMAND ERROR STATUS REGISTER (CMR)

This register contains the code of the last command error detected by the oscilloscope. List these error codes using `CMR?`.

Read CMR with `CMR?`. The response is the error code. Clear the register with a `CMR?` or `ALST?` query, a `*CLS` command, or with power-on.

DEVICE DEPENDENT ERROR STATUS REGISTER (DDR)

DDR indicates the type of hardware errors affecting your *WavePro* DSO. Individual bits in this register report specific hardware failures. List them using `DDR?`.

Also read this register using the `DDR?` query. The response is the binary weighted sum of the error bits. Clear it with another `DDR?` or with `ALST?`, a `*CLS` command, or with power-on.

EXECUTION ERROR STATUS REGISTER (EXR)

EXR contains the code of the last execution error detected by the oscilloscope. List these error codes with `EXR?`.

Read the register, again using the `EXR?` query. The response is the error code. Clear with another `EXR?` or with `ALST?`, a `*CLS` command, or with power-on.

USER REQUEST STATUS REGISTER (URR)

Finally, URR contains the identification code of the last menu button pressed. List these codes with `URR?`.

Read URR using the same query. The response is the decimal code associated with the selected menu button. And clear the register with another `URR?`, or with `ALST?`, a `*CLS` command, or with power-on.

BLANK PAGE

PART TWO

COMMANDS

Part Two describes the commands and queries you will need to remotely operate your *WavePro* oscilloscope.

PART TWO: COMMANDS

In this part of the manual, you'll find the commands and queries to

Run WavePro remotely.

Use *WavePro* DSO Commands and Queries

This part of the manual lists and describes the remote control commands and queries recognized by *WavePro* DSO. You can execute all of them in either local or remote state. Where commands or queries for special options are not included, you will find them in those options' dedicated *Operator's Manuals*.

The commands and queries are listed in alphabetical order according to the long form of their name. For example, the description of ATTENUATION, whose short form is ATTN, is listed before that of AUTO SETUP, whose short form is ASET. Each command or query description starts on a new page. The name (header) is given in both long and short form at the top of the first page of each description.

Queries perform actions such as obtaining information. They are recognized by **?** following their headers. Many commands can be used as queries with the question mark added.

A brief explanation of the operation performed by the command or query is followed the formal syntax, with the full-name header given in lower-case characters and the short form derived from it in upper-case characters (e.g., DoT_JoiN for DTJN). Where applicable, the syntax of the query is given with the format of its response. A short example illustrating a typical use is also presented. The GPIB examples assume that the controller is equipped with a National Instruments interface board, which calls to the related interface subroutines in BASIC. The device name of the oscilloscope is defined as **SCOPE%**.

Use the two tables that precede the descriptions to quickly find a command or query. The first of these lists the commands and queries in alphabetical order according to their long form. The second table groups them according to the subsystem or category they belong to.

COMMAND NOTATION

The following notation is used in the commands:

< >	Angular brackets enclose words that are used as placeholders, of which there are two types: the header path and the data parameter of a command.
: =	A colon followed by an equals sign separates a placeholder from the description of the type and range of values that can be used in a command instead of the placeholder.
{ }	Braces enclose a list of choices, one of which must be made.
[ ]	Square brackets enclose optional items.
...	An ellipsis indicates that the items left and right of it can be repeated any number of times.

Example: consider the syntax notation for the command to set the vertical input sensitivity:

1. <channel> : VOLT_DIV <v_gain>
2. <channel> : = { C1, C2 }
3. <v_gain> : = 5.0 mV to 2.5 V

The first line shows the formal appearance of the command: <channel> denotes the placeholder for the header path; <v_gain> is the placeholder for the vertical gain value.

The second line indicates that either C1 or C2 must be chosen for the header path.

The third line means that the actual vertical gain can be set to any value from 5 mV to 2.5 V.

Table of Commands and Queries — By Short Form

SHORT FORM	LONG FORM	SUBSYSTEM (CATEGORY)	WHAT THE COMMAND OR QUERY DOES
ACAL	AUTO_CALIBRATE	MISCELLANEOUS	Enables or disables automatic calibration.
ALST?	ALL_STATUS?	STATUS	Reads and clears the contents of all status registers.
ARM	ARM_ACQUISITION	ACQUISITION	Changes acquisition state from “stopped” to “single.”
ASCR	AUTO_SCROLL	DISPLAY	Controls the Auto Scroll viewing feature.
ASET	AUTO_SETUP	ACQUISITION	Adjusts vertical, timebase and trigger parameters.
ATTN	ATTENUATION	ACQUISITION	Selects the vertical attenuation factor of the probe.
BUZZ	BUZZER	MISCELLANEOUS	Controls the built-in piezoelectric buzzer.
BWL	BANDWIDTH_LIMIT	ACQUISITION	Enables/ disables bandwidth-limiting low-pass filter.
*CAL?	*CAL?	MISCELLANEOUS	Performs complete internal calibration of oscilloscope.
CFMT	COMM_FORMAT	COMMUNICATION	Selects the format for sending waveform data.
CHDR	COMM_HEADER	COMMUNICATION	Controls formatting of query responses.
CHLP	COMM_HELP	COMMUNICATION	Controls operational level of the RC Assistant.
CHL	COMM_HELP_LOG	COMMUNICATION	Returns the contents of the RC Assistant log.
CHST	CALL_HOST	DISPLAY	Allows manual generation of a service request (SRQ).
CLM	CLEAR_MEMORY	FUNCTION	Clears the specified memory.
*CLS	*CLS	STATUS	Clears all status data registers.
CLSW	CLEAR_SWEEPS	FUNCTION	Restarts the cumulative processing functions.
CMR?	CMR?	STATUS	Reads and clears the CoMmand error Register (CMR).
COLR	COLOR	DISPLAY	Selects color of individual on-screen objects.
CSCH	COLOR_SCHEME	DISPLAY	Selects the display color scheme.
COMB	COMBINE_CHANNELS	ACQUISITION	Controls the channel interleaving function.
CONET	COMM_NET	COMMUNICATION	Specifies network addresses of scope and printers.
CORD	COMM_ORDER	COMMUNICATION	Controls the byte order of waveform data transfers.
CORS	COMM_RS232	COMMUNICATION	Sets remote control parameters of the RS-232-C port.
COUT	CAL_OUTPUT	MISCELLANEOUS	Sets signal type put out at the CAL connector.
CPL	COUPLING	ACQUISITION	Selects the specified input channel’s coupling mode.
CRMS	CURSOR_MEASURE	CURSOR	Specifies the type of cursor/ parameter measurement.
CRRD	CURSOR_READOUT	CURSOR	Sets the cursor amplitude in volts or dBm.
CRST?	CURSOR_SET?	CURSOR	Allows positioning of any one of eight cursors.
CRVA?	CURSOR_VALUE?	CURSOR	Returns trace values measured by specified cursors.
DPNT	DATA_POINTS	DISPLAY	Controls bold/ single pixel display of sample points.
DATE	DATE	MISCELLANEOUS	Changes the date/ time of the internal real-time clock.
DDR?	DDR?	STATUS	Reads, clears the Device Dependent Register (DDR).
DEF	DEFINE	FUNCTION	Specifies math expression for function evaluation.
DELF	DELETE_FILE	MASS STORAGE	Deletes files from mass storage.
DIR	DIRECTORY	MASS STORAGE	Creates and deletes file directories.
DISP	DISPLAY	DISPLAY	Controls the display screen.

SHORT FORM	LONG FORM	SUBSYSTEM (CATEGORY)	WHAT THE COMMAND OR QUERY DOES
DTJN	DOT_JOIN	DISPLAY	Controls the interpolation lines between data points.
DZOM	DUAL_ZOOM	DISPLAY	Sets horizontal magnification and positioning.
EKEY	ENABLE_KEY	DISPLAY	Allows use of the KEY command in local mode.
*ESE	*ESE	STATUS	Sets the Standard Event Status Enable register (ESE).
*ESR?	*ESR?	STATUS	Reads, clears the Event Status Register (ESR).
EXR?	EXR?	STATUS	Reads, clears the EXecution error Register (EXR).
FATC	FAT_CURSOR	DISPLAY	Controls width of cursors.
FCR	FIND_CTR_RANGE	FUNCTION	Automatically sets the center and width of a histogram.
FCRD	FORMAT_CARD	MASS STORAGE	Formats the memory card.
FFLP	FORMAT_FLOPPY	MASS STORAGE	Formats a floppy disk.
FHDD	FORMAT_HDD	MASS STORAGE	Formats the removable hard disk.
FVDISK	FORMAT_VDISK	MASS STROAGE	Formats non-volatile RAM.
FLNM	FILENAME	MASS STORAGE	Changes default filenames.
FSCR	FULL_SCREEN	DISPLAY	Selects magnified view format for the grid.
FRST	FUNCTION_RESET	FUNCTION	Resets a waveform-processing function.
GBWL	GLOBAL_BWL	ACQUISITION	Enables/ disables the Global Bandwidth Limit.
GRID	GRID	DISPLAY	Specifies single-, dual- or quad-mode grid display.
HCSU	HARDCOPY_SETUP	HARD COPY	Configures the hard-copy driver.
HCTR	HARDCOPY_TRANSMIT	HARD COPY	Sends string of ASCII characters to hard-copy unit.
HMAG	HOR_MAGNIFY	DISPLAY	Horizontally expands the selected expansion trace.
HPOS	HOR_POSITION	DISPLAY	Horizontally positions intensified zone's center.
*IDN?	*IDN?	MISCELLANEOUS	For identification purposes.
INE	INE	STATUS	Sets the INternal state change Enable register (INE).
INR?	INR?	STATUS	Reads, clears INternal state change Register (INR).
INSP?	INSPECT?	WAVEFORM TRANSFER	Allows acquired waveform parts to be read.
INTS	INTENSITY	DISPLAY	Sets the grid or trace/ text intensity level.
ILVD	INTERLEAVED	ACQUISITION	Enables/ disables Random Interleaved Sampling (RIS).
IST?	IST?	STATUS	Reads the current state of the IEEE 488.
KEY	KEY	DISPLAY	Displays a string in the menu field.
LOGO	LOGO	DISPLAY	Displays LeCroy logo at top of grid.
MASK	MASK	CURSOR	Invokes PolyMask draw and fill tools.
MLIM	MATH_LIMITS	FUNCTIONS	Limits averaging to only the points of interest.
MGAT	MEASURE_GATE	DISPLAY	Controls highlighting of the measurement gate region.
MSG	MESSAGE	DISPLAY	Displays a string of characters in the message field.
MSIZ	MEMORY_SIZE	ACQUISITION	Selects max. memory length.
MZOM	MULTI_ZOOM	DISPLAY	Sets horizontal magnification and positioning.
OFST	OFFSET	ACQUISITION	Allows output channel vertical offset adjustment.
OFCT	OFFSET_CONSTANT	ACQUISITION	Sets offset in volts or divisions.
*OPC	*OPC	STATUS	Sets the OPC bit in the Event Status Register (ESR).

SHORT FORM	LONG FORM	SUBSYSTEM (CATEGORY)	WHAT THE COMMAND OR QUERY DOES
***OPT?**	***OPT?**	MISCELLANEOUS	Identifies oscilloscope options.
OPMZ	**OPTIMIZE**	ACQUISITION	Allows faster readout by not applying bandwidth compensation.
PNSU	**PANEL_SETUP**	SAVE/RECALL	Complements the *SAV/*RST commands.
PACL	**PARAMETER_CLR**	CURSOR	Clears all current parameters in Custom, Pass/Fail.
PACU	**PARAMETER_CUSTOM**	CURSOR	Controls parameters with customizable qualifiers.
PADL	**PARAMETER_DELETE**	CURSOR	Deletes a specified parameter in Custom, Pass/Fail.
PAST?	**PARAMETER_STATISTICS?**	CURSOR	Returns current statistics parameter values.
PAVA?	**PARAMETER_VALUE?**	CURSOR	Returns current parameter, mask test values.
PFCO	**PASS_FAIL_CONDITION**	CURSOR	Adds a Pass/Fail test condition or custom parameter.
PFCT	**PASS_FAIL_COUNTER**	CURSOR	Resets the Pass/Fail acquisition counters.
PFDO	**PASS_FAIL_DO**	CURSOR	Defines desired outcome, actions after Pass/Fail test.
PFMS	**PASS_FAIL_MASK**	CURSOR	Generates tolerance mask on a trace and stores it.
PFST?	**PASS_FAIL_STATUS?**	CURSOR	Returns the Pass/Fail test for a given line number.
PECS	**PER_CURSOR_SET**	CURSOR	Positions independent cursors.
PECV?	**PER_CURSOR_VALUE?**	CURSOR	Returns values measured by cursors.
PERS	**PERSIST**	DISPLAY	Enables or disables the persistence display mode.
PECL	**PERSIST_COLOR**	DISPLAY	Controls color rendering method of persistence traces.
PELT	**PERSIST_LAST**	DISPLAY	Shows the last trace drawn in a persistence data map.
PESA	**PERSIST_SAT**	DISPLAY	Sets the color saturation level in persistence.
PESU	**PERSIST_SETUP**	DISPLAY	Selects display persistence duration.
***PRE**	***PRE**	STATUS	Sets the PaRallel poll Enable register (PRE).
PRCA?	**PROBE_CAL?**	PROBES	Performs auto-calibration of connected current probe.
PRDG?	**PROBE_DEGAUSS?**	PROBES	Degausses, calibrates connected current probe.
PRIT?	**PROBE_INFOTEXT?**	PROBES	Returns the connected probe's informative text.
PRNA	**PROBE_NAME?**	PROBES	Names the probe connected to the oscilloscope.
***RCL**	***RCL**	SAVE/RECALL	Recalls one of five non-volatile panel setups.
REC	**RECALL**	WAVEFORM TRANSFER	Recalls a file from mass storage to internal memory.
RCPN	**RECALL_PANEL**	SAVE/RECALL	Recalls a front panel setup from mass storage.
ROUT	**REAR_OUTPUT**	MICELLANEOUS	Sets the type of signal put out at rear BNC connector.
RCLK	**REFERENCE_CLOCK**	ACQUISITION	Selects the system clock source: internal or external.
***RST**	***RST**	SAVE/RECALL	Initiates a device reset.
***SAV**	***SAV**	SAVE/RECALL	Stores current state in non-volatile internal memory.
SCDP	**SCREEN_DUMP**	HARD COPY	Causes a screen dump to the hard-copy device.
SCLK	**SAMPLE_CLOCK**	ACQUISITION	Allows control of an external timebase.
SCREEN	**SCREEN**	DISPLAY	Turns the screen on or off.
SCSV	**SCREEN_SAVE**	DISPLAY	Controls the automatic screen saver.
SEL	**SELECT**	DISPLAY	Selects the specified trace for manual display control.
SEQ	**SEQUENCE**	ACQUISITION	Sets the conditions for the sequence mode acquisition.

SHORT FORM	LONG FORM	SUBSYSTEM (CATEGORY)	WHAT THE COMMAND OR QUERY DOES
SKEY	SKEY	DISPLAY	Allows you to assign text, borders, and files to Custom DSO menu soft keys.
SLEEP	SLEEP	MICELLANEOUS	Makes the scope wait before it interprets new commands.
*SRE	*SRE	STATUS	Sets the Service Request Enable register (SRE).
*STB?	*STB?	STATUS	Reads the contents of the IEEE 488.
STITLE	STITLE	DISPLAY	Allows you to title a panel of menus.
STOP	STOP	ACQUISITION	Immediately stops signal acquisition.
STO	STORE	WAVEFORM TRANSFER	Stores a trace in internal memory or mass storage.
STPN	STORE_PANEL	SAVE/RECALL	Stores front panel setup to mass storage.
STST	STORE_SETUP	WAVEFORM TRANSFER	Controls the way in which traces are stored.
STTM	STORE_TEMPLATE	WAVEFORM TRANSFER	Stores the waveform template to mass storage.
TDIV	TIME_DIV	ACQUISITION	Modifies the timebase setting.
TDISP	TDISP	MISCELLANEOUS	Changes time display from current to trigger or none.
TMPL?	TEMPLATE?	WAVEFORM TRANSFER	Produces a complete waveform template copy.
TRA	TRACE	DISPLAY	Enables or disables the display of a trace.
TRLB	TRACE_LABEL	DISPLAY	Allows you to enter a label for a trace.
TOPA	TRACE_OPACITY	DISPLAY	Controls the opacity of the trace color.
TORD	TRACE_ORDER	DISPLAY	Allows you to specify the display order of traces.
TRFL	TRANSFER_FILE	WAVEFORM TRANSFER	Transfers ASCII files to and from storage media, or between scope and computer.
*TRG	*TRG	ACQUISITION	Executes an ARM command.
TRCP	TRIG_COUPLING	ACQUISITION	Sets the coupling mode of the specified trigger source.
TRDL	TRIG_DELAY	ACQUISITION	Sets the time at which the trigger is to occur.
TRLV	TRIG_LEVEL	ACQUISITION	Adjusts the trigger level of the specified trigger source.
TRMD	TRIG_MODE	ACQUISITION	Specifies the trigger mode.
TRPA	TRIG_PATTERN	ACQUISITION	Defines a trigger pattern.
TRSE	TRIG_SELECT	ACQUISITION	Selects the condition that will trigger acquisition.
TRSL	TRIG_SLOPE	ACQUISITION	Sets the trigger slope of the specified trigger source.
TRWI	TRIG_WINDOW	ACQUISITION	Sets window amplitude on current Edge trigger source.
*TST?	*TST?	MISCELLANEOUS	Performs an internal self-test.
URR?	URR?	STATUS	Reads, clears User Request status Register (URR).
VDIV	VOLT_DIV	ACQUISITION	Sets the vertical sensitivity.
VMAG	VERT_MAGNIFY	DISPLAY	Vertically expands the specified trace.
VPOS	VERT_POSITION	DISPLAY	Adjusts the vertical position of the specified trace.
*WAI	*WAI	STATUS	WAIt to continue - required by the IEEE 488.
WAIT	WAIT	ACQUISITION	Prevents new analysis until current is completed.

SHORT FORM	LONG FORM	SUBSYSTEM (CATEGORY)	WHAT THE COMMAND OR QUERY DOES
WF	**WAVEFORM**	WAVEFORM TRANSFER	Transfers a waveform from controller to scope.
WFSU	**WAVEFORM_SETUP**	WAVEFORM TRANSFER	Specifies amount of waveform data to go to controller.
WFTX	**WAVEFORM_TEXT**	WAVEFORM TRANSFER	Documents acquisition conditions.
XYAS?	**XY_ASSIGN?**	DISPLAY	Returns traces currently assigned to the XY display.
XYCO	**XY_CURSOR_ORIGIN**	CURSOR	Sets origin position of absolute cursor measurements.
XYCS	**XY_CURSOR_SET**	CURSOR	Allows positioning of XY voltage cursors.
XYCV?	**XY_CURSOR_VALUE?**	CURSOR	Returns the current values of the X vs. Y cursors.
XYDS	**XY_DISPLAY**	DISPLAY	Enables or disables the XY display mode.
XYRD	**XY_RENDER**	DISPLAY	Controls XY plot: smooth or disconnected points.
XYSA	**XY_SATURATION**	DISPLAY	Sets persistence color saturation level in XY display.

Table of Commands and Queries — By Subsystem

SHORT FORM	LONG FORM	WHAT THE COMMAND OR QUERY DOES
ACQUISITION — TO CONTROL WAVEFORM CAPTURE		
ARM	ARM_ACQUISITION	Changes acquisition state from "stopped" to "single."
ASET	AUTO_SETUP	Adjusts vertical, timebase and trigger parameters for signal display.
ATTN	ATTENUATION	Selects the vertical attenuation factor of the probe.
BWL	BANDWIDTH_LIMIT	Enables or disables the bandwidth-limiting low-pass filter.
COMB	COMBINE_CHANNELS	Controls the channel interleaving function.
CPL	COUPLING	Selects the specified input channel's coupling mode.
GBWL	GLOBAL_BWL	Enables/ disables the Global Bandwidth Limit.
ILVD	INTERLEAVED	Enables or disables Random Interleaved Sampling (RIS).
MSIZ	MEMORY_SIZE	Allows selection of maximum memory length.
OFST	OFFSET	Allows vertical offset adjustment of the specified input channel.
OFCT	OFFSET_CONSTANT	Sets offset in volts or divisions.
OPMZ	OPTIMIZE	Allows faster readout by not applying bandwidth compensation.
RCLK	REFERENCE_CLOCK	Selects the system clock source: internal or external.
SCLK	SAMPLE_CLOCK	Allows control of an external timebase.
SEQ	SEQUENCE	Sets the conditions for Sequence-mode acquisition.
STOP	STOP	Immediately stops signal acquisition.
TDIV	TIME_DIV	Modifies the timebase setting.
*TRG	*TRG	Executes an ARM command.
TRCP	TRIG_COUPLING	Sets the coupling mode of the specified trigger source.
TRDL	TRIG_DELAY	Sets the time at which the trigger is to occur.
TRLV	TRIG_LEVEL	Adjusts the level of the specified trigger source.
TRMD	TRIG_MODE	Specifies Trigger mode.
TRPA	TRIG_PATTERN	Defines a trigger pattern.
TRSE	TRIG_SELECT	Selects the condition that will trigger acquisition.
TRSL	TRIG_SLOPE	Sets the slope of the specified trigger source.
TRWI	TRIG_WINDOW	Sets the window amplitude in volts on the current Edge trigger source.
VDIV	VOLT_DIV	Sets the vertical sensitivity in volts/ div.
WAIT	WAIT	Prevents new command analysis until current acquisition completion.
COMMUNICATION — TO SET COMMUNICATION CHARACTERISTICS		
CFMT	COMM_FORMAT	Selects the format to be used for sending waveform data.
CHDR	COMM_HEADER	Controls formatting of query responses.
CHLP	COMM_HELP	Controls operational level of the RC Assistant.
CHL	COMM_HELP_LOG	Returns the contents of the RC Assistant log.
CONET	COMM_NET	Specifies network addresses of scope and printers.
CORD	COMM_ORDER	Controls the byte order of waveform data transfers.
CORS	COMM_RS232	Sets remote control parameters of the RS-232-C port.

SHORT FORM	LONG FORM	WHAT THE COMMAND OR QUERY DOES
CURSOR — TO PERFORM MEASUREMENTS		
CRMS	CURSOR_MEASURE	Specifies the type of cursor or parameter measurement for display.
CRRD	CURSOR_READOUT	Sets the cursor amplitude in volts or dBm.
CRST?	CURSOR_SET?	Allows positioning of any one of eight independent cursors.
CRVA?	CURSOR_VALUE?	Returns the values measured by the specified cursors for a given trace.
MASK	MASK	Invokes PolyMask draw and fill tools.
PACL	PARAMETER_CLR	Clears all current parameters in Custom and Pass/Fail modes.
PACU	PARAMETER_CUSTOM	Controls parameters with customizable qualifiers.
PADL	PARAMETER_DELETE	Deletes a specified parameter in Custom and Pass/Fail modes.
PAST?	PARAMETER_STATISTICS?	Returns current statistics values for the specified pulse parameter.
PAVA?	PARAMETER_VALUE?	Returns current value(s) of parameter(s) and mask tests.
PECS	PER_CURSOR_SET	Allows positioning of any one of six independent cursors.
PECV?	PER_CURSOR_VALUE?	Returns the values measured by specified cursors for a given trace.
PFCO	PASS_FAIL_CONDITION	Adds a Pass/Fail test condition or custom parameter to display.
PFCT	PASS_FAIL_COUNTER	Resets the Pass/Fail acquisition counters.
PFDO	PASS_FAIL_DO	Defines the desired outcome and actions following a Pass/Fail test.
PFMS	PASS_FAIL_MASK	Generates a tolerance mask around a chosen trace and stores it.
PFST?	PASS_FAIL_STATUS?	Returns the Pass/Fail test for a given line number.
XYCO	XY_CURSOR_ORIGIN	Sets position of origin for absolute cursor measurements on XY display.
XYCS	XY_CURSOR_SET	Allows positioning of any one of six independent XY voltage cursors.
XYCV?	XY_CURSOR_VALUE?	Returns current values of X vs. Y cursors.
DISPLAY — TO DISPLAY WAVEFORMS		
ASCR	AUTO_SCROLL	Controls the Auto Scroll viewing feature.
CHST	CALL_HOST	Allows manual generation of a service request (SRQ).
COLR	COLOR	Selects color of individual objects such as traces, grids or cursors.
CSCH	COLOR_SCHEME	Selects the display color scheme.
DPNT	DATA_POINTS	Controls display of sample points in single display pixels or bold.
DISP	DISPLAY	Controls the oscilloscope display screen.
DTJN	DOT_JOIN	Controls the interpolation lines between data points.
DZOM	DUAL_ZOOM	Sets horizontal magnification and positioning for all expanded traces.
EKEY	ENABLE_KEY	Allows use of the KEY command in local mode.
FATC	FAT_CURSOR	Controls width of cursors.
FSCR	FULL_SCREEN	Selects magnified view format for the grid.
GRID	GRID	Specifies grid display in single, dual or quad mode.
HMAG	HOR_MAGNIFY	Horizontally expands the selected expansion trace.
HPOS	HOR_POSITION	Horizontally positions the intensified zone's center on the source trace.
INTS	INTENSITY	Sets grid or trace/text intensity level.
KEY	KEY	Displays a string in the menu field.
LOGO	LOGO	Displays LeCroy logo at top of grid.

SHORT FORM	LONG FORM	WHAT THE COMMAND OR QUERY DOES
MGAT	MEASURE_GATE	Controls highlighting of the region between the parameter cursors.
MSG	MESSAGE	Displays a string of characters in the message field.
MZOM	MULTI_ZOOM	Sets horizontal magnification and positioning for all expanded traces.
PERS	PERSIST	Enables or disables the Persistence Display mode.
PECL	PERSIST_COLOR	Controls color rendering method of persistence traces.
PELT	PERSIST_LAST	Shows the last trace drawn in a persistence data map.
PESA	PERSIST_SAT	Sets the color saturation level in persistence.
PESU	PERSIST_SETUP	Selects display persistence duration in Persistence mode.
SCREEN	SCREEN	Turns the screen on or off.
SCSV	SCREEN_SAVE	Controls the automatic screen saver.
SEL	SELECT	Selects the specified trace for manual display control.
SKEY	SKEY	Allows you to assign text, borders, and files to Custom DSO menu soft keys.
STITLE	STITLE	Allows you to title a panel of menus.
TRA	TRACE	Enables or disables the display of a trace.
TRLB	TRACE_LABEL	Allows you to enter a label for a trace.
TOPA	TRACE_OPACITY	Controls the opacity of the trace color.
TORD	TRACE_ORDER	Allows you to specify the display order of traces.
VMAG	VERT_MAGNIFY	Vertically expands the specified trace.
VPOS	VERT_POSITION	Adjusts the vertical position of the specified trace.
XYAS?	XY_ASSIGN?	Returns the traces currently assigned to the XY display.
XYDS	XY_DISPLAY	Enables or disables the XY display mode.
XYRD	XY_RENDER	Controls XY plot: smooth or disconnected points.
XYSA	XY_SATURATION	Sets persistence color saturation level in XY display.
FUNCTION — TO PERFORM WAVEFORM MATHEMATICAL OPERATIONS		
CLM	CLEAR_MEMORY	Clears the specified memory.
CLSW	CLEAR_SWEEPS	Restarts the cumulative processing functions.
DEF	DEFINE	Specifies the mathematical expression to be evaluated by a function.
FCR	FIND_CTR_RANGE	Automatically sets the center and width of a histogram.
FRST	FUNCTION_RESET	Resets a waveform processing function.
MLIM	MATH_LIMITS	Limits averaging to only the points of interest.
HARD COPY — TO PRINT THE CONTENTS OF THE DISPLAY		
HCSU	HARDCOPY_SETUP	Configures the hard-copy driver.
HCTR	HARDCOPY_TRANSMIT	Sends a string of unmodified ASCII characters to the hard-copy unit.
SCDP	SCREEN_DUMP	Causes a screen dump to the hardcopy device.
MASS STORAGE — TO CREATE AND DELETE FILE DIRECTORIES		
DELF	DELETE_FILE	Deletes files from the currently selected directory on mass storage.
DIR	DIRECTORY	Creates and deletes file directories on mass-storage devices.
FCRD	FORMAT_CARD	Formats the memory card.

SHORT FORM	LONG FORM	WHAT THE COMMAND OR QUERY DOES
FFLP	FORMAT_FLOPPY	Formats a floppy disk in the Double- or High-Density format.
FHDD	FORMAT_HDD	Formats the removable hard disk.
FVDISK	FORMAT_VDISK	Formats non-volatile RAM.
FLNM	FILENAME	Changes the default filename of any stored trace, setup or hard copy.
MISCELLANEOUS — TO CALIBRATE AND TEST		
ACAL	AUTO_CALIBRATE	Enables or disables automatic calibration.
BUZZ	BUZZER	Controls the built-in piezoelectric buzzer.
*CAL?	*CAL?	Performs a complete internal calibration of the oscilloscope.
COUT	CAL_OUTPUT	Sets the type of signal put out at the CAL connector.
DATE	DATE	Changes the date/time of the oscilloscope's internal real-time clock.
*IDN?	*IDN?	Used for identification purposes.
*OPT?	*OPT?	Identifies oscilloscope options.
ROUT	REAR_OUTPUT	Sets the type of signal put out at the rear BNC connector.
SLEEP	SLEEP	Makes the scope wait before it interprets new commands
TDISP	TDISP	Changes time display from current to trigger or none.
*TST?	*TST?	Performs an internal self-test.
PROBES — TO USE PROBES		
PRCA?	PROBE_CAL?	Performs a complete calibration of the connected current probe.
PRDG	PROBE_DEGAUSS?	Degausses and calibrates the connected current probe.
PRIT?	PROBE_INFOTEXT?	Returns the connected probe's informative text.
PRNA	PROBE_NAME?	Gives an identification of the probe connected to the oscilloscope.
SAVE/RECALL SETUP — TO PRESERVE AND RESTORE FRONT PANEL SETTINGS		
PNSU	PANEL_SETUP	Complements the *SAV/*RST commands.
*RCL	*RCL	Recalls one of five non-volatile panel setups.
RCPN	RECALL_PANEL	Recalls a front panel setup from mass storage.
*RST	*RST	Initiates a device reset.
*SAV	*SAV	Stores the current state in non-volatile internal memory.
STPN	STORE_PANEL	Stores the complete front panel setup on a mass-storage file.
STATUS — TO OBTAIN STATUS INFORMATION AND SET UP SERVICE REQUESTS		
ALST?	ALL_STATUS?	Reads and clears the contents of all (but one) of the status registers.
*CLS	*CLS	Clears all the status data registers.
CMR?	CMR?	Reads and clears the contents of the CoMmand error Register (CMR).
DDR?	DDR?	Reads and clears the Device-Dependent error Register (DDR).
*ESE	*ESE	Sets the standard Event Status Enable (ESE) register.
*ESR?	*ESR?	Reads and clears the Event Status Register (ESR).
EXR?	EXR?	Reads and clears the EXecution error Register (EXR).
INE	INE	Sets the INternal state change Enable register (INE).
INR?	INR?	Reads and clears the INternal state change Register (INR).
IST?	IST?	Individual STatus reads the current state of IEEE 488.

SHORT FORM	LONG FORM	WHAT THE COMMAND OR QUERY DOES
*OPC	*OPC	Sets to true the OPC bit (0) in the Event Status Register (ESR).
*PRE	*PRE	Sets the PaRallel poll Enable register (PRE).
*SRE	*SRE	Sets the Service Request Enable register (SRE).
*STB?	*STB?	Reads the contents of IEEE 488.
URR?	URR?	Reads and clears the User Request status Register (URR).
*WAI	*WAI	WAIt to continue — required by IEEE 488.
WAVEFORM TRANSFER — TO PRESERVE AND RESTORE WAVEFORMS		
INSP?	INSPECT?	Allows acquired waveform parts to be read.
REC	RECALL	Recalls a waveform file from mass storage to internal memories M1–4.
STO	STORE	Stores a trace in one of the internal memories M1–4 or mass storage.
STST	STORE_SETUP	Controls the way in which traces are stored.
STTM	STORE_TEMPLATE	Stores the waveform template in a mass-storage device.
TMPL?	TEMPLATE?	Produces a copy of the template describing a complete waveform.
TRFL	TRANSFER_FILE	Transfers ASCII files to and from storage media, or between scope and computer.
WF	WAVEFORM	Transfers a waveform from the controller to the oscilloscope.
WFSU	WAVEFORM_SETUP	Specifies amount of waveform data for transmission to controller.
WFTX	WAVEFORM_TEXT	Documents the conditions under which a waveform has been acquired

STATUS

ALL_STATUS?, ALST?

Query

DESCRIPTION

The ALL_STATUS? query reads and clears the contents of all status registers: STB, ESR, INR, DDR, CMR, EXR and URR except for the MAV bit (bit 6) of the STB register. For an interpretation of the contents of each register, refer to the appropriate status register.

The query is useful in a complete overview of the state of your *WavePro* oscilloscope.

QUERY SYNTAX

```
ALl_STatus?
```

RESPONSE FORMAT

```
ALl_STatus STB,<value>,ESR,<value>,INR,<value>,
DDR,<value>,CMR,<value>,EXR,<value>,URR,<value>
```

<value> : = 0 to 65535

EXAMPLE (GPIB)

The following reads the contents of all the status registers:

```
CMD$="ALST?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
ALST
TB,000000,ESR,000052,INR,000005,DDR,000000,
CMR,000004,EXR,000024,URR,000000
```

RELATED COMMANDS

```
*CLS, CMR?, DDR?, *ESR?, EXR?, *STB?, URR?
```

ACQUISITION

ARM_ACQUISITION, ARM

Command

DESCRIPTION

The ARM_ACQUISITION command enables the signal acquisition process by changing the acquisition state (trigger mode) from "stopped" to "single".

COMMAND SYNTAX

```
ARM_acquisition
```

EXAMPLE

The following enables signal acquisition:

```
CMD$="ARM": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

STOP, *TRG, TRIG_MODE, WAIT

ACQUISITION

ATTENUATION, ATTN

Command/ Query

DESCRIPTION

The ATTENUATION command selects the vertical attenuation factor of the probe. Values of 1, 2, 5, 10, 20, 25, 50, 100, 200, 500, 1000 or 10000 can be specified.

The ATTENUATION? query returns the attenuation factor of the specified channel.

COMMAND SYNTAX

<channel> : ATTeNuation <attenuation>

<channel> := { C1, C2, C3, C4, EX, EX10}
<attenuation> := { 1, 2, 5, 10, 20, 25, 50, 100, 200, 500, 1000, 10000}

QUERY SYNTAX

<channel> : ATTeNuation?

RESPONSE FORMAT

<channel> : ATTeNuation <attenuation>

AVAILABILITY

<channel> : { C3, C4} available only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following sets to 100 the attenuation factor of Channel 1:

```
CMD$="C1:ATTN 100": CALL IBWRT(SCOPE%,CMD$)
```

MISCELLANEOUS

AUTO_CALIBRATE, ACAL

Command/ Query

DESCRIPTION

The AUTO_CALIBRATE command is used to enable or disable the automatic calibration of your *WavePro* oscilloscope. At power-up, auto-calibration is turned ON, i.e. all input channels are periodically calibrated for the current input amplifier and timebase settings.

The automatic calibration can be disabled by issuing the command ACAL OFF. Whenever convenient, a *CAL? query may be issued to fully calibrate the oscilloscope. When the oscilloscope is returned to local control, the periodic calibrations are resumed.

The response to the AUTO_CALIBRATE? query indicates whether auto-calibration is enabled.

COMMAND SYNTAX

Auto_CALibrate <state>

<state> := { ON, OFF}

QUERY SYNTAX

Auto_CALibrate?

RESPONSE FORMAT

Auto_CALibrate <state>

EXAMPLE (GPIB)

The following disables auto-calibration:

```
CMD$="ACAL OFF": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

*CAL?

DISPLAY

AUTO_SCROLL, ASCR

Command/ Query

DESCRIPTION

The AUTO_SCROLL command and query controls the Auto Scroll feature, accessed through the front panel using the MATH SETUP button and ZOOM + MATH menus. This automatically moves the selected trace (or all traces if multi-zoom is on) across the screen. The command turns the scroll on and off and sets the scrolling speed in divisions per second, and the query returns the current scroll rate.

COMMAND SYNTAX

Auto_SCRoll <action>,<mode>,<speed>

<action> : = { PLAY, REVERSE, STOP}

<mode> : = { DIV_S, DIV_U}

<speed> : = 0.01 to 10.00 for DIV_U mode;
0.10 to 10.00 for DIV_S mode.

QUERY SYNTAX

Auto_SCRoll?

RESPONSE FORMAT

Auto_SCRoll <action>,<mode>,<speed>

EXAMPLE (GPIB)

The following activates Auto Scroll and start scrolling the data to the right at a rate of 2 s/ div.:

```
CMD$="ASCR PLAY,DIV_S,2": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

MULTI_ZOOM, HOR_MAGNIFY, HOR_POSITION

ACQUISITION

AUTO_SETUP, ASET
Command

DESCRIPTION

The AUTO_SETUP command attempts to display the input signal(s) by adjusting the vertical, timebase and trigger parameters. AUTO_SETUP operates only on the channels whose traces are currently turned on. If no traces are turned on, AUTO_SETUP operates on all channels and turns on all of the traces.

If signals are detected on several channels, the lowest numbered channel with a signal determines the selection of the timebase and trigger source.

If only one input channel is turned on, the timebase will be adjusted for that channel.

The <channel> : AUTO_SETUP FIND command adjusts gain and offset only for the specified channel.

COMMAND SYNTAX

<channel> : Auto_SETup [FIND]

<channel> : = { C1, C2, C3, C4}

If the FIND keyword is present, gain and offset adjustments will be performed only on the specified channel. In this case, if no <channel> prefix is added, then an auto-setup will be performed on the channel used on the last ASET FIND remote command. In the absence of the FIND keyword, the normal auto-setup will be performed, regardless of the <channel> prefix.

AVAILABILITY

<channel> : = { C3, C4} only on four-channel *WavePro* oscilloscopes.

EXAMPLE

The following instructs the oscilloscope to perform an auto-setup:

```
CMD$="ASET": CALL IBWRT(SCOPE%,CMD$)
```

ACQUISITION

BANDWIDTH_LIMIT, BWL

Command/ Query

DESCRIPTION

BANDWIDTH_LIMIT enables or disables the bandwidth-limiting low-pass filter. When Global_BWL (see page 143) is on, the BWL command applies to all channels; when off, the command is used to set the bandwidth individually for each channel. The response to the BANDWIDTH_LIMIT? query indicates whether the bandwidth filters are on or off.

COMMAND SYNTAX

BandWidth_Limit <mode>

Or, alternatively, to choose the bandwidth limit of an individual channel or channels when Global_BWL is off:

BandWidth_Limit <channel>,<mode>[,<channel>,<mode>

[,<channel>,<mode>[,<channel>,<mode>]]]

<mode> : = {OFF, ON, 200MHZ}

<channel> : = {C1, C2, C3, C4}

QUERY SYNTAX

BandWidth_Limit?

RESPONSE FORMAT

When Global_BWL is on, or if Global_BWL is off and all four channels have the same bandwidth limit, the response is:

BandWidth_Limit <mode>

Or, alternatively, if at least two channels have their bandwidth limit filters set differently from one another, the response is:

BandWidth_Limit <channel>,<mode>[,<channel>,<mode>

[,<channel>,<mode>[,<channel>,<mode>]]]

AVAILABILITY

{C3, C4} : Available only on four-channel models.

EXAMPLE

The following turns on the bandwidth filter for all channels, when Global_BWL is on (as it is by default):

```
CMD$="BWL ON": CALL IBWRT(SCOPE%,CMD$)
```

The following turns the bandwidth filter on for Channel 1 only (the first turns off Global_BWL):

```
CMD$="GBWL OFF": CALL IBWRT(SCOPE%,CMD$)
CMD$="BWL C1,ON": CALL IBWRT(SCOPE%,CMD$)
CMD$="GBWL OFF": CALL IBWRT(SCOPE%,CMD$)
CMD$="BWL C1,ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS GLOBAL_BWL

MISCELLANEOUS

BUZZER, BUZZ
Command

DESCRIPTION

The BUZZER command controls the built-in piezoelectric buzzer. This is useful for attracting the attention of a local operator in an interactive working application. The buzzer can either be activated for short beeps (about 400 ms long in BEEP mode) or continuously for a certain time interval you select by turning the buzzer ON or OFF.

NOTE: This command is only able to be used in oscilloscopes fitted with the CLBZ hard option.

COMMAND SYNTAX

BUZZer <state>

<state> : = { BEEP, ON, OFF}

EXAMPLE (GPIB)

Sending the following will cause the oscilloscope to sound two short tones.

```
CMD$="BUZZ BEEP;BUZZ BEEP":
CALL IBWRT(SCOPE%,CMD$)
```

MISCELLANEOUS

*CAL?
Query

DESCRIPTION

The *CAL? query causes the oscilloscope to perform an internal self-calibration and generates a response that indicates whether or not your *WavePro* oscilloscope completed the calibration without error. This internal calibration sequence is the same as that which occurs at power-up. At the end of the calibration, after the response has indicated how the calibration terminated, the oscilloscope returns to the state it was in just prior to the calibration cycle.

QUERY SYNTAX

*CAL?

RESPONSE FORMAT

*CAL <diagnostics>

<diagnostics> := 0 or other

0 = Calibration successful

EXAMPLE (GPIB)

The following forces a self-calibration:

```
CMD$="*CAL?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Response message (if no failure): *CAL 0

RELATED COMMANDS

AUTO_CALIBRATE

MISCELLANEOUS

CAL_OUTPUT, COUT

Command/ Query

DESCRIPTION

The CAL_OUTPUT command is used to set the type of signal put out at the *WavePro* DSO front panel CAL BNC connector.

COMMAND SYNTAX

Cal_OUTput <mode>[,<level>[,<rate>]]

Cal_OUTput PULSE[,<width>]

<mode> : = {OFF, CALSQ, CALPU, PF, TRIG, LEVEL, PULSE, TRDY}
<level> : = –1.0 to 1.00 V into 1 MΩ
<rate> : = 500 Hz to 2 MHz.
<width> : = 10 us to 10 s (applies only to PULSE)

QUERY SYNTAX

Cal_OUTput?

RESPONSE FORMAT

Cal_OUTput <mode>,<level>[,<rate>]

AVAILABILITY

<mode> : PULSE or LEVEL will only be accepted if the Cal_OUTput? mode was previously OFF.

EXAMPLE (GPIB)

The following sets the calibration signal to give a 0–0.2 volt pulse of 25 ns width at a 10 kHz rate:

```
CMD$="COUT CALPU,0.2 V,10 kHz":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PASS_FAIL_DO

ADDITIONAL INFORMATION

NOTATION	
CALPU	Provides a pulse signal
CALSQ	Provides a square signal
LEVEL	Provides a DC signal at the requested level
OFF	Provides no signal (ground level)
PF	Pass/Fail mode
PULSE	Provides a single pulse
TRIG	Trigger Out mode

DISPLAY

CALL_HOST, CHST

Command/ Query

DESCRIPTION

The CALL_HOST command allows you to manually generate a service request (SRQ). Once the CALL_HOST command has been received, the message "Call Host" will be displayed next to the lowest button on the menu-button column immediately next to the screen. Pressing this button while in the root menu sets the User Request status Register (URR) and the URQ bit of the Event Status Register. This can generate a SRQ in local mode, provided the service request mechanism has been enabled.

The response to the CALL_HOST? query indicates whether CALL_HOST is enabled (on) or disabled (off).

COMMAND SYNTAX

```
Call_HoST <state>
```

<state> : = { ON, OFF}

QUERY SYNTAX

```
Call_HoST?
```

RESPONSE FORMAT

```
Call_HoST <state>
```

EXAMPLE (GPIB)

After executing the following an SRQ request will be generated whenever the button is pressed (it is assumed that SRQ servicing has already been enabled):

```
CMD$="CHST ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

URR

FUNCTION **CLEAR_MEMORY, CLM**
Command

DESCRIPTION

The CLEAR_MEMORY command clears the specified memory. Data previously stored in this memory are erased and memory space is returned to the free memory pool.

COMMAND SYNTAX

CLear_Memory < memory>

<memory> := {M1, M2, M3, M4}

EXAMPLE (GPIB)

The following clears the memory M2.

```
CMD$="CLM M2": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

STORE

FUNCTION

CLEAR_SWEEPS, CLSW
Command

DESCRIPTION

The CLEAR_SWEEPS command restarts the cumulative processing functions: summed or continuous average, extrema, FFT power average, histogram, pulse parameter statistics, Pass/Fail counters, and persistence.

COMMAND SYNTAX

```
CLear SWeeps
```

EXAMPLE (GPIB)

The following example will restart the cumulative processing:

```
CMD$="CLSW": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DEFINE, INR

STATUS

*CLS
Command

DESCRIPTION

The *CLS command clears all status data registers.

COMMAND SYNTAX

```
*CLS
```

EXAMPLE (GPIB)

The following causes all the status data registers to be cleared:

```
CMD$="*CLS": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

ALL_STATUS, CMR, DDR, *ESR, EXR, *STB, URR

STATUS

CMR?
Query

DESCRIPTION

The CMR? query reads and clears the contents of the CoMmand error Register (CMR) — see table next page — which specifies the last syntax error type detected by your *WavePro* oscilloscope.

QUERY SYNTAX

CMR?

RESPONSE FORMAT

CMR <value>
<value> : = 0 to 13

EXAMPLE (GPIB)

The following reads the contents of the CMR register:

```
CMD$="CMR?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

RELATED COMMANDS

ALL_STATUS?, *CLS

ADDITIONAL INFORMATION

COMMAND ERROR STATUS REGISTER STRUCTURE (CMR)	
Value	**Description**
1	Unrecognized command/ query header
2	Illegal header path
3	Illegal number
4	Illegal number suffix
5	Unrecognized keyword
6	String error
7	GET embedded in another message
10	Arbitrary data block expected
11	Non-digit character in byte count field of arbitrary data block
12	EOI detected during definite length data block transfer
13	Extra bytes detected during definite length data block transfer

DISPLAY

COLOR, COLR
Command/ Query

DESCRIPTION

The COLOR command is used to select the color of an individual display object such as text, trace, grid or cursor.

The response to the COLOR? query indicates the color assigned to each display object, whether or not it is currently displayed.

NOTE: This command is only effective if the color scheme (CSCH) is chosen from the user schemes U1, U2, U3 or U4.

COMMAND SYNTAX

COLoR <object, color>[,...<object>,<color>]

<object> : = { BACKGND, C1, C2, C3, C4, TA, TB, TC, TD, GRID, TEXT, CURSOR, NEUTRAL, WARNING},

<color> : = { WHITE, CYAN, YELLOW, GREEN, MAGENTA, BLUE, RED, LTGRAY, GRAY, SLGRAY, CHGRAY, DKCYAN, CREAM, SAND, AMBER, OLIVE, LTGEEN, JADE, LMGREEN, APGREEN, EMGREEN, GRGREEN, OCSPRAY, ICEBLUE, PASTBLUE, PALEBLUE, SKYBLUE, ROYLBLUE, DEEPBLUE, NAVY, PLUM, PURPLE, AMETHYST, FUCHSIA, RASPBRY, NEONPINK, PALEPINK, PINK, VERMIL, ORANGE, CERISE, KHAKI, BROWN, BLACK}

QUERY SYNTAX

COLoR?

RESPONSE FORMAT

COLoR <object>,<color>[,...<object>,<color>]

EXAMPLE (GPIB)

The following selects color scheme U1, and then red as the color of Channel 1:

```
CMD$="CSCH U1": CALL IBWRT(SCOPE%,CMD$)
CMD$="COLR C1,RED": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

COLOR_SCHEME, PERSIST_COLOR

ADDITIONAL INFORMATION

NOTATION			
<color>	**Color**	**<color>**	**Color**
WHITE	White	OCSPRAY	Ocean Spray
CYAN	Cyan	ICEBLUE	Ice Blue
YELLOW	Yellow	PASTBLUE	Pastel Blue
GREEN	Green	PALEBLUE	Pale Blue
MAGENTA	Magenta	SKYBLUE	Sky Blue
BLUE	Blue	ROYLBLUE	Royal Blue
RED	Red	DEEPBLUE	Deep Blue
LTGRAY	Light Gray	NAVY	Navy
GRAY	Gray	PLUM	Plum
SLGRAY	Slate Gray	PURPLE	Purple
CHGRAY	Charcoal Gray	AMETHYST	Amethyst
DKCYAN	Dark Cyan	FUCHSIA	Fuchsia
CREAM	Cream	RASPB	Raspberry
SAND	Sand	NEONPINK	Neon Pink
AMBER	Amber	PALEPINK	Pale Pink
OLIVE	Olive	PINK	Pink
LTGREEN	Light Green	VERMIL	Vermilion
JADE	Jade	ORANGE	Orange
LMGREEN	Lime Green	CERISE	Cerise
APGREEN	Apple Green	KHAKI	Khaki

EMGREEN	Emerald Green	BROWN	Brown
GRGREEN	Grass Green	BLACK	Black
<object>	**Display Object**	**<object>**	**Display Object**
BACKGND	Background	CURSOR	Cursors
C1..C4	Channel Traces	WARNING	Warning Messages
TA..TD	Function Traces	NEUTRAL	Neutral Color
GRID	Grid Lines	OVERLAYS	Menu background color in FULL SCREEN

DISPLAY

COLOR_SCHEME, CSCH

Command/ Query

DESCRIPTION

The COLOR_SCHEME command is used to select the color scheme for the display.

The response to the COLOR_SCHEME? query indicates the color scheme in use.

COMMAND SYNTAX

Color_SCHeme <scheme>

<scheme> := {1, 2, 3, 4, 5, 6, 7, U1, U2, U3, U4}

QUERY SYNTAX

Color_SCHeme?

RESPONSE FORMAT

Color_SCHeme <scheme>

EXAMPLE (GPIB)

The following selects the user color scheme U2:

```
CMD$="CSCH U2": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

COLOR, PERSIST_COLOR

ACQUISITION

COMBINE_CHANNELS, COMB

Command/ Query

DESCRIPTION

The COMBINE_CHANNELS command controls the channel interleaving function of the acquisition system. The COMBINE_CHANNELS? query returns the channel interleaving function's current status.

COMMAND SYNTAX

COMBine_channels <state>

<state> : = { 1, 2, 4, AUTO}

QUERY SYNTAX

COMBine_channels?

RESPONSE FORMAT

COMB <state>

EXAMPLE (GPIB)

The following instruction engages channel interleaving between Channels 1 and 2, and Channels 3 and 4:

CMD$="COMB 2": CALL IBWRT(SCOPE%,CMD$)

The following instruction sets Auto-Combine mode:

CMD$="COMB AUTO": CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

TDIV

COMMUNICATION

COMM_FORMAT, CFMT

Command/ Query

DESCRIPTION

The COMM_FORMAT command selects the format the oscilloscope uses to send waveform data. The available options allow the block format, the data type and the encoding mode to be modified from the default settings.

The COMM_FORMAT? query returns the currently selected waveform data format.

COMMAND SYNTAX

Comm_ForMaT <block_format>,<data_type>,<encoding>

<block_format> := { DEF9, IND0, OFF}

<data_type> := { BYTE, WORD}

<encoding> := { BIN, HEX}

(GPIB uses both encoding forms, RS-232-C always uses HEX)

Initial settings (i.e. after power-on) are:

For GPIB: DEF9, WORD, BIN

For RS-232-C: DEF9, WORD, HEX

QUERY SYNTAX

Comm_ForMaT?

RESPONSE FORMAT

Comm_ForMaT <block_format>,<data_type>,<encoding>

EXAMPLE (GPIB)

The following redefines the transmission format of waveform data. The data will be transmitted as a block of indefinite length. Data will be coded in binary and represented as eight-bit integers.

```
CMD$="CFMT IND0,BYTE,BIN": CALL IBWRT(SCOPE%,CMD$)
```

ADDITIONAL INFORMATION

Block Format

DEF9: Uses the IEEE 488.2 definite length arbitrary block response data format. The digit 9 indicates that the byte count consists of 9 digits. The data block directly follows the byte count field.

For example, a data block consisting of three data bytes would be sent as:

`WF DAT1,#9000000003`<DAB><DAB><DAB>

where <DAB> represents an eight-bit binary data byte.

IND0: Uses the IEEE 488.2 indefinite length arbitrary block response data format.

A <NL^END> (new line with EOI) signifies that block transmission has ended.

The same data bytes as above would be sent as:

`WF DAT1,#0`<DAB><DAB><DAB><NL^END>

OFF: Same as IND0. In addition, the data block type identifier and the leading #0 of the indefinite length block will be suppressed. The data presented above would be sent as:

`WF` <DAB><DAB><DAB><NL^END>

NOTE: The format OFF does not conform to the IEEE 488.2 standard and is only provided for special applications where the absolute minimum of data transfer may be important.

Data Type

BYTE: Transmits the waveform data as eight-bit signed integers (one byte).

WORD: Transmits the waveform data as 16-bit signed integers (two bytes).

NOTE: The data type BYTE transmits only the high-order bits of the internal 16-bit representation. The precision contained in the low-order bits is lost.

Encoding

BIN: Binary encoding (GPIB only)

HEX: Hexadecimal encoding (bytes are converted to 2 hexadecimal ASCII digits (0, ...9, A, ...F))

RELATED COMMANDS `WAVEFORM`

COMMUNICATION

COMM_HEADER, CHDR

Command/ Query

DESCRIPTION

The COMM_HEADER command controls the way the oscilloscope formats responses to queries. There are three response formats: LONG, in which responses start with the long form of the header word; SHORT, where responses start with the short form of the header word; and OFF, for which headers are omitted from the response and suffix units in numbers are suppressed.

Unless you request otherwise, the SHORT response format is used.

NOTE: The default format, i.e. that just after power-on, is SHORT.

This command does not affect the interpretation of messages sent to the oscilloscope. Headers can be sent in their long or short form regardless of the COMM_HEADER setting.

Querying the vertical sensitivity of Channel 1 may result in one of the following responses:

COMM_HEADER	RESPONSE
LONG	C1:VOLT_DIV 200E-3 V
SHORT	C1:VDIV 200E-3 V
OFF	200E-3

COMMAND SYNTAX

Comm_HeaDeR <mode>

<mode> := { SHORT, LONG, OFF}

QUERY SYNTAX

Comm_HeaDeR?

RESPONSE FORMAT

Comm_HeaDeR <mode>

EXAMPLE (GPIB)

The following code sets the response header format to SHORT:

```
CMD$="CHDR SHORT": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

COMM_HELP_LOG

COMMUNICATION

COMM_HELP, CHLP

Command/ Query

DESCRIPTION

The COMM_HELP command controls the level of operation of the diagnostics utility Remote Control Assistant (see the *WavePro* DSO Operator's Manual), which assists in debugging remote control programs. Selected using your *WavePro* oscilloscope's front panel (see the Operator's Manual), Remote Control Assistant can log all message transactions occurring between the external controller and the oscilloscope. You can view the log at any time on-screen and can choose from four levels:

OFF	Don't assist at all.
EO	Log detected Errors Only (default after power-on).
FD	Log the Full Dialog between the controller and the oscilloscope.
RS	Log the Full Dialog and send it to a recording device connected to the RS232 port.

COMMAND SYNTAX

Comm_HeLP <level>

<level> := { OFF, EO, FD, RS, }

The default level (i.e. the level just after power-on) is EO.

QUERY SYNTAX

Comm_HeLP?

RESPONSE FORMAT

Comm_HeLP <level>

EXAMPLE (GPIB)

After sending this command, all the following commands and responses will be logged:

```
CMD$="CHLP FD": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

COMM_HELP_LOG

COMMUNICATION **COMM_HELP_LOG?, CHL?**
Query

DESCRIPTION

The COMM_HELP_LOG query returns the current contents of the log generated by the Remote Control Assistant (see CHLP description). If the optional parameter CLR is specified, the log will be cleared after the transmission. Otherwise, it will be kept.

QUERY SYNTAX

Comm_HeLP_Log? [CLR]

RESPONSE FORMAT

Comm_Help_Log <string containing the logged text>

EXAMPLE (GPIB)

The following reads the remote control log and prints it:

```
CMD$="CHL?": CALL IBWRT(SCOPE%,CMD$)PRINT
```

RELATED COMMANDS

COMM_HELP

COMMUNICATION

COMM_NET, CONET

Command/ Query

DESCRIPTION

The COMM_NET command specifies the IP network address of the scope. It also specifies the network address of a workstation or printer that is running the LeCroy DSO Print Gateway. The COMM_NET? query returns the current network address.

COMMAND SYNTAX

COmm_NET <subaddress>

<subaddress> : = { IP, "X.X.X.X.", MASK, "X.X.X.X", GATEWAY, "X.X.X.X",PRINTSVR,"X.X.X.X"}

QUERY SYNTAX

COmm_NET?

RESPONSE FORMAT

COmm_NET <subaddress>

EXAMPLE

_____WARNING_____

This command acts immediately. The software used to send the command will need to be initialized with the new address before continuing.

Query and response:

CONET?

IP,"172.28.11.49",MASK,"255.255.248.0", GATEWAY,"172.28.8.1",PRINTSVR,"172.29.14.68"

This command changes the IP:

COMM_NET IP,"172.28.11.77"

COMMUNICATION

COMM_ORDER, CORD

Command/ Query

DESCRIPTION

The COMM_ORDER command controls the byte order of waveform data transfers. Waveform data can be sent with the most significant byte (MSB) or the least significant byte (LSB) in the first position. The default mode is to send the MSB first. COMM_ORDER applies equally to the waveform's descriptor and time blocks. In the descriptor some values are 16 bits long ("word"), 32 bits long ("long" or "float"), or 64 bits long ("double"). In the time block all values are floating values, i.e. 32 bits long. When COMM_ORDER HI is specified, the MSB is sent first; when COMM_ORDER LO is specified, the LSB is sent first.

The COMM_ORDER? query returns the byte transmission order in current use.

COMMAND SYNTAX

Comm_ORDer <mode>

<mode> := {HI, LO}

NOTE: The initial mode, i.e. the mode after power-on, is HI.

QUERY SYNTAX

Comm_ORDer?

RESPONSE FORMAT

Comm_ORDer <mode>

EXAMPLE

The order of transmission of waveform data depends on the data type. The following table illustrates the different possibilities:

TYPE	CORD HI	CORD LO
Word	<MSB><LSB>	<LSB><MSB>
Long or Float	<MSB><byte2><byte3><LSB>	<LSB><byte3><byte2><MSB>
Double	<MSB><byte2>...<byte7><LSB>	<LSB><byte7>...<byte2><MSB>

RELATED COMMANDS WAVEFORM

COMMUNICATION

COMM_RS232, CORS

Command/ Query

DESCRIPTION

The COMM_RS232 command sets the parameters of the RS-232-C port for remote control.

The COMM_RS232? query reports the settings of the parameters.

NOTE: This command and query is only valid when you control the oscilloscope remotely using the WavePro RS-232-C port.

PARAMETERS

End Input character: When received by the oscilloscope, this character is interpreted as the END-of-a-command message marker. The commands received will be parsed and executed.

End Output string: The oscilloscope adds this string at the end of a response message. When the host computer receives this string, it knows that the oscilloscope has completed its response.

Line Length: This parameter defines the maximum number of characters sent to the host in a single line. Remaining characters of the response are output in separate additional lines. This parameter is only applicable if a line separator has been selected.

Line Separator: This parameter is used to select the line-splitting mechanism and to define the characters used to split the oscilloscope response messages into many lines. Possible line separators are: CR, LF, CRLF. <CR>, <LF> or <CR> followed by <LF>. These are sent to the host computer after <line_length> characters.

SRQ string: This string is sent each time the oscilloscope signals an SRQ to the host computer.

Some COMM_RS232 parameters require ASCII strings as actual arguments. In order to facilitate the embedding of non-printable characters into such strings, escape sequences can be used. The back-slash character ('\') is used as an escape character. The following escape sequences are recognized:

"\a":	Bell character
"\b":	Back space character
"\e":	Escape character
"\n":	Line feed character
"\r":	Carriage return character
"\t":	Horizontal tab character
"\\":	The back-slash character itself
"\ddd":	Represents one to three decimal digit characters giving the code value of the corresponding ASCII character. This allows any ASCII code in the range 1 to 127 to be inserted.

Before using the string, the oscilloscope will replace the escape sequence by the corresponding ASCII character.

For example, the escape sequences "\r", "\13" and "\013" are all replaced by the single ASCII character <Carriage Return>.

NOTATION	
EI	End input character
EO	End output string
LL	Line length
LS	Line separator
SRQ	SRQ service request

COMMAND SYNTAX

COmm_RS232 EI,<ei_char>,EO,'<eo_string>',LL,<line_length>,LS,<line_sep>,SRQ,'<srq_string>'

<ei_char> := 1 to 126 (default: 13 = Carriage Return)

<eo_string> := A non-empty ASCII string of up to 20 characters (default: "\n\r")

<line_length> := 40 to 1024 (default: 256)

<line_sep> := {OFF, CR, LF, CRLF} (default: OFF)

<srq_string> := An ASCII string of up to 20 characters which may be empty (default: empty string)

QUERY SYNTAX

COmm_RS232?

RESPONSE FORMAT

COmm_RS232 EI,<ei_char>,EO,"p <eo_string>",LL,<line_length>,LS,<line_sep>,SRQ,"<srq_string>"

EXAMPLE

After executing the command

```
COMM_RS232 EI,3,EO, "\r\nEND\r\n"
```

the oscilloscope will assume that it has received a complete message each time the <ETX> (decimal value 3) is detected. Response messages will be terminated by sending the character sequence "<CR><LF>END<CR><LF>".

ACQUISITION

COUPLING, CPL

Command/ Query

DESCRIPTION

The COUPLING command selects the coupling mode of the specified input channel.

The COUPLING? query returns the coupling mode of the specified channel.

COMMAND SYNTAX

<channel> : CouPLing <coupling>

<channel> : = { C1, C2, C3, C4, EX, EX10}

<coupling> : = { A1M, D1M, D50, GND}

QUERY SYNTAX

<channel> : CouPLing?

RESPONSE FORMAT

<channel> : CouPLing <coupling>

<coupling> : = { A1M, D1M, D50, GND, OVL}

<coupling> : OVL is returned in the event of signal overload while in DC 50 Ω coupling. In this condition, the oscilloscope will disconnect the input.

AVAILABILITY

<channel> : = { C3, C4} only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following sets the coupling of Channel 2 to 50 Ω DC:

```
CMD$="C2:CPL D50": CALL IBWRT(SCOPE%,CMD$)
```

CURSOR

CURSOR_MEASURE, CRMS

Command/ Query

DESCRIPTION

The CURSOR_MEASURE command specifies the type of cursor or parameter measurement to be displayed, and is the main command for displaying parameters and Pass/ Fail.

The CURSOR_MEASURE? query indicates which cursors or parameter measurements are currently displayed.

NOTATION	
ABS	absolute reading of relative cursors
CUST	custom parameters
FAIL	Pass/ Fail: fail
HABS	horizontal absolute cursors
HPAR	standard time parameters
HREL	horizontal relative cursors
OFF	cursors and parameters off
PARAM	synonym for VPAR
PASS	Pass/ Fail: pass
SHOW	custom parameters (old form)
STAT	parameter statistics
VABS	vertical absolute cursors
VPAR	standard voltage parameters
VREL	vertical relative cursors

NOTE: The PARAM mode is turned OFF when XY mode is ON.

COMMAND SYNTAX

CuRsor_MeaSure <mode>[,<submode>]

<mode> : = { CUST, FAIL, HABS, HPAR, HREL, OFF, PARAM, PASS, SHOW, VABS, VPAR, VREL}

<submode> : = { STAT, ABS}

NOTE: The keyword STAT is optional with modes CUST, HPAR, and VPAR. If present, STAT turns parameter statistics on. Absence of STAT turns parameter statistics off.

The keyword ABS is optional with mode HREL. If it is present, ABS chooses absolute amplitude reading of relative cursors. Absence of ABS selects relative amplitude reading of relative cursors.

QUERY SYNTAX

```
CuRsor_MeaSure?
```

RESPONSE FORMAT

```
CuRsor_MeaSure <mode>
```

EXAMPLE (GPIB)

The following switches on the vertical relative cursors:

```
CMD$="CRMS VREL": CALL IBWRT(SCOPE%,CMD$)
```

The following determines which cursor is currently turned on:

```
CMDS$="CRMS?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Example of response message:

```
CRMS OFF
```

RELATED COMMANDS

```
CURSOR_SET, PARAMETER_STATISTICS,
PARAMETER_VALUE, PASS_FAIL_CLEAR,
PASS_FAIL_CONDITION, PASS_FAIL_DELETE,
PASS_FAIL_MASK,
```

ADDITIONAL INFORMATION

To turn off the cursors, parameter measurements or Pass/Fail tests, use:

```
CURSOR_MEASURE OFF
```

To turn on a cursor display, use one of these five forms:

```
CURSOR_MEASURE HABS
CURSOR_MEASURE HREL
CURSOR_MEASURE VABS
CURSOR_MEASURE VREL
CURSOR_MEASURE FAIL
```

To select parameters in the Custom mode, and to modify the test conditions in the Pass/Fail mode, use the command:

```
PASS_FAIL_CONDITION
```

CURSOR

CURSOR_READOUT, CRRD?

Command/ Query

DESCRIPTION

This command sets the readout of the time cursor amplitude in volts or dBm.

COMMAND SYNTAX

CuRsor_ReaDout <scale>

<scale> : = { VOLTS,DBM}

QUERY SYNTAX

CRRD?

RESPONSE FORMAT

CURSOR_READOUT VOLTS

EXAMPLE (GPIB)

The following command sets the amplitude of the time cursors to read out in dBm:

```
CMD$=”CRRD DBM”
CALL IBWRT (SCOPE%,CMD$)
```

RELATED COMMANDS

CURSOR_MEASURE

CURSOR

CURSOR_SET, CRST

Command/ Query

DESCRIPTION

The CURSOR_SET command allows you to position any one of the eight independent cursors at a given screen location. The positions of the cursors can be modified or queried even if the required cursor is not currently displayed on the screen.

When setting a cursor position, a trace must be specified, relative to which the cursor will be positioned.

NOTE: If the parameter display is turned on (or the Pass/Fail display or the extended parameters display), the parameters of the specified trace will be shown unless the newly chosen trace is not displayed or has been acquired in sequence mode; these conditions will produce an environment error (see table on page 130). To change only the trace without repositioning the cursors, the CURSOR_SET command can be given with no argument (for example, TB:CRST).

The CURSOR_SET? query indicates the current position of the cursor(s). The values returned depend on the grid type selected.

NOTATION			
HABS	horizontal absolute	PREF	parameter reference
HDIF	horizontal difference	VABS	vertical absolute
HREF	horizontal reference	VDIF	vertical difference
PDIF	parameter difference	VREF	vertical reference

COMMAND SYNTAX

<trace> : CuRsor_SeT <cursor>,<position>[,<cursor>,<position>,<cursor> ,<position>]

<trace> : = { TA, TB, TC, TD, C1, C2, C3,C4}

<cursor> : = { HABS, VABS, HREF, HDIF, VREF, VDIF, PREF, PDIF}

<position> := 0 to 10 DIV (horizontal)

<position> := −29.5 to 29.5 DIV (vertical)

NOTE: Parameters are grouped in pairs. The first parameter specifies the cursor to be modified and the second one indicates its new value. Parameters can be grouped in any order and restricted to those items to be changed.

The suffix DIV is optional.

QUERY SYNTAX

<trace> : CuRsor_SeT? [<cursor>,...<cursor>]

<cursor> := { HABS, VABS, HREF, HDIF, VREF, VDIF, PREF, PDIF, ALL}

RESPONSE FORMAT

<trace> : CuRsor_SeT
<cursor>,<position>[,<cursor>,<position>,...<cursor>,<position>]

If <cursor> is not specified, ALL will be assumed. If the position of a cursor cannot be determined in a particular situation, its position will be indicated as UNDEF.

AVAILABILITY

<trace> : { C3, C4} available only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following positions the VREF and VDIF cursors at +3 DIV and −7 DIV respectively, using Trace A as a reference:

```
CMD$="TA:CRST VREF,3DIV,VDIF,−7DIV":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR VALUE, PARAMETER_VALUE, PER_CURSOR_SET, XY_CURSOR_SET

CURSOR

CURSOR_VALUE?, CRVA?

Query

DESCRIPTION

The CURSOR_VALUE? query returns the values measured by the specified cursors for a given trace. (The PARAMETER_VALUE? query is used to obtain measured waveform parameter values.)

NOTATION	
HABS	horizontal absolute
HREL	horizontal relative

QUERY SYNTAX

<trace> : CuRsor_VAlue? [<mode>,...<mode>]

<trace> : = { TA, TB, TC, TD, C1, C2, C3, C4}

<mode> : = { HABS, HREL, VABS, VREL, ALL}

RESPONSE FORMAT

<trace> : CuRsor_VAlue HABS,<abs_hori>,<abs_vert>

<trace> : CuRsor_VAlue HREL,<delta_hori>,<delta_vert>,

<absvert_ref>,<absvert_dif>,<slope>

The dV/dt value <slope> is displayed in the appropriate trace label box.

<trace> : CuRsor_VAlue VABS,<abs_vert>

<trace> : CuRsor_VAlue VREL,<delta_vert>

For horizontal cursors, both horizontal as well as vertical values are given. For vertical cursors only vertical values are given.

NOTE: If <mode> is not specified or equals ALL, all the measured cursor values for the specified trace are returned. If the value of a cursor cannot be determined in the current environment, the value UNDEF will be returned.

AVAILABILITY

<trace> : = { C3, C4} available only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following query reads the measured absolute horizontal value of the cross-hair cursor (HABS) on Channel 2:

```
CMD$="C2:CRVA? HABS": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
C2:CRVA HABS,34.2E-6 S, 244 E-3 V
```

RELATED COMMANDS

CURSOR_SET, PARAMETER_VALUE, PER_CURSOR_VALUE, XY_CURSOR_VALUE

DISPLAY

DATA_POINTS, DPNT
Command/Query

DESCRIPTION

The DATA_POINTS command is used to control whether the waveform sample points are shown as single display pixels or are made bold.

The response to the DATA_POINTS? query indicates whether the waveform sample points are being displayed as single display pixels or in bold face.

COMMAND SYNTAX

Data_PoiNTs <state>

<state> := {NORMAL, BOLD}

QUERY SYNTAX

Data_PoiNTs?

RESPONSE FORMAT

Data_PoiNTs <state>

EXAMPLE (GPIB)

The following highlights the waveform sample points:

```
CMD$="DPNT BOLD": CALL IBWRT(SCOPE%,CMD$)
```

MISCELLANEOUS

DATE

Command/ Query

DESCRIPTION

The DATE command changes the date/ time of the oscilloscope's internal real-time clock.

The DATE? query returns the current date/ time setting.

COMMAND SYNTAX

DATE <day>,<month>,<year>,<hour>,<minute>,<second>

<day> : = 1 to 31

<month> : = { JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC}

<year> : = 1990 to 2037

<hour> : = 0 to 23

<minute> : = 0 to 59

<second> : = 0 to 59

TIP: You need only specify those DATE parameters up to and including the parameter to be changed in order to change the year setting: day, month and year. The time settings will remain unchanged. But to change the second setting, all the DATE parameters must be specified with the required settings.

QUERY SYNTAX

DATE?

RESPONSE FORMAT

DATE <day>,<month>,<year>,<hour>,<minute>,<second>

EXAMPLE (GPIB)

This will change the date to January 1, 1997 and the time to 1:21:16 p.m. (13:21:16 in 24-hour notation):

```
CMD$="DATE 1,JAN,1997,13,21,16": CALL
IBWRT(SCOPE%,CMD$)
```

STATUS

DDR?
Query

DESCRIPTION

The DDR? query reads and clears the contents of the Device Dependent or device specific error Register (DDR). In the case of a hardware failure, the DDR register specifies the origin of the failure. The following table gives details.

BIT	BIT VALUE	DESCRIPTION	
15...14		0	Reserved
13	8192	1	Timebase hardware failure detected
12	4096	1	Trigger hardware failure detected
11	2048	1	Channel 4 hardware failure detected
10	1024	1	Channel 3 hardware failure detected
9	512	1	Channel 2 hardware failure detected
8	256	1	Channel 1 hardware failure detected
7	128	1	External input overload condition detected
6...4		0	Reserved
3	8	1	Channel 4 overload condition detected
2	4	1	Channel 3 overload condition detected
1	2	1	Channel 2 overload condition detected
0	1	1	Channel 1 overload condition detected

QUERY SYNTAX

DDR?

RESPONSE FORMAT

DDR <value>
<value> : = 0 to 65535

AVAILABILITY

<value> : Bit 2, 3, 10, 11 only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following reads the contents of the DDR register:

```
CMD$="DDR?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
DDR 0
```

RELATED COMMANDS

ALL_STATUS, *CLS

FUNCTION

DEFINE, DEF
Command/ Query

DESCRIPTION

The DEFINE command specifies the mathematical expression to be evaluated by a function. This command is used to control all math tools and zoom in the standard oscilloscope as well as those in the Extended Math and WaveAnalyzer options. See the *Operator's Manual* for more about *WavePro* DSO Math Tools.

COMMAND SYNTAX

<function> : DEFine EQN,'<equation>'
[,<param_name>,<value>,...]

NOTE: Function parameters are grouped in pairs. The first in the pair names the variable to be modified, <param_name>, while the second one gives the new value to be assigned. Pairs can be given in any order and restricted to the variables to be changed.

Space (blank) characters inside equations are optional.

QUERY SYNTAX

<function> : DEFine?

RESPONSE FORMAT

<function> : DEFine EQN,'<equation>'[,MAXPTS,<max_points>]
[,SWEEPS,<max_sweeps>][,WEIGHT,<weight>][,BITS,<bits>]

FUNCTION PARAMETERS

<param_name>	<value>	Description
BITS	<bits>	Number of ERES bits
CENTER		Horizontal center position for histogram display.
EQN	'<equation>'	Function equation as defined below
LENGTH	<length>	Number of points to use from first waveform
MAX_EVENTS	<max_values>	Maximum number of values in histogram
MAXBINS	<bins>	Number of bins in histogram
MAXPTS	<max_points>	Maximum number of points to compute

START	<start>	Starting point in second waveform
SWEEPS	<max_sweeps>	Maximum number of sweeps
UNITS	<units>	Physical units
VERT	<vert_scale>	Vertical scaling type
WEIGHT	<weight>	Continuous Average weight
WIDTH	<width>	Width of histogram display
WINDOW	<window_type>	FFT window function

FUNCTION EQUATIONS AND NAMES	
NOTE: These are available according to the options installed in your WavePro oscilloscope. See Chapter 5 of the Operator's Manual for math and waveform processing options.	
–<source>	Negation
+<source>	Identity
<source>	
<source1> - <source2>	Subtraction
<source1> + <source2>	Addition
<source1>/<source2>	Ratio
<source1><source2>	Multiplication
1/<source>	Reciprocal
ABS(<source>)	Absolute Value
AVGC(<source>)	Continuous Average
AVGS(<source>)	Average Summed
DERI(<source>)	Derivative
ERES(<source>)	Enhanced Resolution
EXP(<source>)	Exponential (power of e)
EXP10(<source>)	Exponential (power of 10)
EXTR(<source>)	Extrema (Roof and Floor)

NOTE: For FFT functions, the source must be a time-domain, single-segment waveform.	
FFT(<source>)	Fast Fourier Transform (complex result)
FLOOR(EXTR(<source>))	Floor (Extrema source only)
HIST(<custom_line>)	Histogram of parameter on custom line
IMAG(FFT(<source>))	Imaginary part of complex result
INTG(<source>[{+,–} <addend>])	Integral
LN(<source>)	Logarithm base e
LOG10(<source>)	Logarithm base 10
MAG(AVGP(<function>))	FFT power average of magnitude
NOTE: For FFT Average functions, the source waveform must also be defined as an FFT function.	
MAG(FFT(<source>))	Magnitude of complex result
PHASE(FFT(<source>))	Phase angle (degrees) of complex result
PS(AVGP(<function>))	FFT average of power spectrum
PS(FFT(<source>))	Power spectrum
PSD(AVGP(<function>))	FFT power average of power density
PSD(FFT(<source>))	Power density
REAL(FFT(<source>))	Real part of complex result
RESC([{+,–}][<multiplier>*]<source>[{+,–} <addend>])	Rescale
ROOF(EXTR(<source>))	Roof (Extrema source only)
SINX(<source>)	Sin(x)/ x interpolator
SQR(<source>)	Square
SQRT(<source>)	Square Root
ZOOMONLY (<extended_source>)	Zoom only (No Math)

NOTE: The numbers in CUST1, CUST2, CUST3, CUST4, and CUST5 refer to the line numbers of the selected custom parameters.

SOURCE VALUES

<sourceN> : = { TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3, C4}

<function> : = { TA, TB, TC, TD}

<custom_line> : = { CUST1, CUST2, CUST3, CUST4, CUST5}

<extended_source> : = { C1, C2, C3,C4, TA, TB, TC, TD, M1, M2, M3, M4}

VALUES TO DEFINE NUMBER OF POINTS/SWEEPS:

<max_points> : = 50 to 10 000 000

<max sweeps> : = 1 to 1000

<max_sweeps> : = 1 to 1 000 000 (with WaveAnalyzer only)

<max_sweeps> : = 1 to 50 000

VALUES FOR RESCALE FUNCTION:

<addend> : = 0.0 to 1e15

<multiplier> : = 0.0 to 1e15

<units> : = { UNCHANGED, A, CEL, C, HZ, K, N, OHM, PAL, V, W, DB, DEG, PCT, RAD, S}

RESCALE PHYSICAL UNITS VALUE NOTATION			
UNCHANGED	The unit remains unchanged.	PAL	Pascal
A	Amperes	V	Volt
CEL	Celsius	W	Watt
C	Coulomb	DB	decibel
HZ	Hertz	DEG	degree
K	Kelvin	PCT	percent
N	Newton	RAD	radian
OHM	Ohm	S	second

VALUES FOR SUMMATION AVERAGE AND ERES:

<weight> : = { 1, 3, 7, 15, 31, 63, 127, 255, 511, 1023}

<bits> : = { 0.5, 1.0, 1.5, 2.0, 2.5, 3.0}

VALUES FOR FFT WINDOWS:

<window_type> : = { BLHA, FLTP, HAMM, HANN, RECT}

FFT WINDOW FUNCTION NOTATION	
LHA	Blackman–Harris window
FLTP	Flat Top window
HAMM	Hamming window
HANN	von Hann window
RECT	Rectangular window

HISTOGRAM VALUES

<max bins> : = { 20, 50, 100, 200, 500, 1000, 2000}

<max_events> : = 20 to 2e9 (in a 1–2–5 sequence)

 : = –1e15 to 1e15

<width> : = 1e–30 to 1e30 (in a 1–2–5 sequence)

<vert_scale> : = { LIN, LOG, CONSTMAX}

HISTOGRAM NOTATION	
LIN	Use linear vertical scaling for histogram display
LOG	Use log vertical scaling for histogram display
CONSTMAX	Use constant maximum linear scaling for histogram display

PRML CORRELATION VALUES

<length> : = 0 to 10 divisions

<start> : = 0 to 10 divisions

AVAILABILITY

<sourceN> : = { C3, C4} only on four-channel oscilloscopes.

<extended_source> : = { C3, C4} only on four-channel oscilloscopes

SWEEPS is the maximum number of sweeps (Average and Extrema only).

NOTE: The pair SWEEPS,<max_sweeps> applies only to the summed averaging (AVGS).

EXAMPLES (GPIB)

The following defines Trace A to compute the summed average of Channel 1 using 5000 points over 200 sweeps:

```
CMD$="TA:DEF
EQN,'AVGS(C1)',MAXPTS,5000,SWEEPS,200":

CALL IBWRT(SCOPE%,CMD$)
```

The following defines Trace A to compute the product of Channel 1 and Channel 2, using a maximum of 10 000 input points:

```
CMD$="TA:DEF EQN,'C1*C2',MAXPTS,10000": CALL
IBWRT(SCOPE%,CMD$)
```

The following defines Trace A to compute the Power Spectrum of the FFT of Channel 1. A maximum of 1000 points will be used for the input. The window function is Rectangular.

```
CMD$="TA:DEF EQN,'PS(FFT(C1))',MAXPTS,1000,WINDOW,
RECT": CALL IBWRT(SCOPE%,CMD$)
```

The following defines Trace B to compute the Power Spectrum of the Power Average of the FFT being computed by Trace A, over a maximum of 244 sweeps.

```
CMD$="TB:DEF EQN,'PS(AVGP(TA))',SWEEPS,244":
CALL IBWRT(SCOPE%,CMD$)
```

The following defines Trace C to construct the histogram of the all risetime measurements made on source Channel 1. The risetime measurement is defined on custom line 2. The histogram has a linear vertical scaling and the risetime parameter values are binned into 100 bins.

```
CMD$="PACU 2,RISE,C1":CALL IBWRT(SCOPE%,CMD$)

CMD$="TC:DEF
EQN,'HIST(CUST2)',VERT,LIN,MAXBINS,100":

CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

```
FIND_CTR_RANGE, FUNCTION_RESET, INR?,
PARAMETER_CUSTOM, PARAMETER_VALUE?,
PASS_FAIL_CONDITION
```

MASS STORAGE

DELETE_FILE, DELF
Command

DESCRIPTION

The DELETE_FILE command deletes files from the currently selected directory on mass storage.

COMMAND SYNTAX

DELete_File DISK,<device>,FILE,'<filename>'

<device> : = { CARD, FLPY, HDD, VDISK}

<filename> : = An alphanumeric string of up to eight characters, followed by a dot and an extension of up to three characters.

AVAILABILITY

<device> : CARD available only when Memory Card option is fitted.

<device> : HDD available only when removable Hard Disk Drive option is fitted.

<device> : VDISK = non-volatile RAM

EXAMPLE (GPIB)

The following deletes a front panel setup from the memory card:

```
CMD$="DELF DISK,CARD,FILE,'P001.PNL'":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DIRECTORY, FORMAT_CARD, FORMAT_FLOPPY, FORMAT_HDD

MASS STORAGE

DIRECTORY, DIR

Command/ Query

DESCRIPTION

The DIRECTORY command is used to manage the creation and deletion of file directories on mass storage devices. It also allows selection of the current working directory and listing of files in the directory.

The query response consists of a double-quoted string containing a DOS-like listing of the directory. If no mass storage device is present, or if it is not formatted, the string will be empty.

COMMAND SYNTAX

DIRectory
DISK,<device>, ACTION,<action>,'<directory>'

QUERY SYNTAX

DIRectory? DISK,<device> [,'<directory>']

<device> : = { CARD, FLPY, HDD, VDISK }

<action> : = { CREATE, DELETE, SWITCH}

<directory> : = A legal DOS path or filename. (This can include the '\' character to define the root directory.)

RESPONSE FORMAT

DIRectory DISK,<device> "<directory> "

<directory> : = A variable length string detailing the file content of the memory card, floppy disk, hard disk, virtual memory.

AVAILABILITY

<device> : CARD available only with the Memory Card option installed.

<device> : HDD available only with the removable Hard Disk option installed.

<device> : VDISK = non-volatile RAM

EXAMPLE (GPIB)

The following asks for a listing of the directory of the memory card:

```
CMD$="DIR? DISK,CARD": CALL
IBWRT(SCOPE%,CMD$):
CALL IBRD (SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
DIR "
Directory          LECROY       1 DIR of04-SEP-2000
  10:46:20 on Memory Card
SC1         000    2859         19-SEP-200016:33:06
SC1         001    2859         19-SEP-200016:34:32
TEST5       002    20359        12-SEP-200013:34:12
3 File(s) 1948672 bytes free
"
```

DISPLAY

DISPLAY, DISP
Command/ Query

DESCRIPTION

The DISPLAY command controls the display screen of the oscilloscope. When remotely controlling the oscilloscope, and if you do not need to use the display, it can be useful to switch off the display via the DISPLAY OFF command. This improves oscilloscope response time, since the waveform graphic generation procedure is suppressed.

The response to the DISPLAY? query indicates the display state of the oscilloscope.

NOTE: When you set the display to OFF, the real-time clock and the message field are updated. But waveforms and associated texts remain unchanged.

COMMAND SYNTAX

DISPlay <state>
<state> : = {ON, OFF}

QUERY SYNTAX

DISPlay?

RESPONSE FORMAT

DISPlay <state>

EXAMPLE (GPIB)

The following turns off the display:

```
CMD$="DISP OFF": CALL IBWRT(SCOPE%,CMD$)
```

DISPLAY

DOT_JOIN, DTJN
Command/ Query

DESCRIPTION

The DOT_JOIN command controls the interpolation lines between data points.

COMMAND SYNTAX

DoT_JoiN <state>
<state> := {ON, OFF}

QUERY SYNTAX

DoT_JoiN?

RESPONSE FORMAT

DoT_JoiN <state>

EXAMPLE (GPIB)

The following turns off the interpolation lines:

```
CMD$="DTJN OFF": CALL IBWRT(SCOPE%,CMD$)
```

DISPLAY

DUAL_ZOOM, DZOM

Command/ Query

DESCRIPTION

By setting DUAL_ZOOM ON, the horizontal magnification and positioning controls are applied to all expanded traces simultaneously. This command is useful if the contents of all expanded traces are to be examined at the same time.

The DUAL_ZOOM? query indicates whether multiple zoom is enabled or not.

NOTE: This command has the same effect as MULTI_ZOOM.

COMMAND SYNTAX

Dual_ZOoM <mode>

<mode> := { ON, OFF}

QUERY SYNTAX

Dual_ZOoM?

RESPONSE FORMAT

Dual_ZOoM <mode>

EXAMPLE (GPIB)

The following turns dual zoom on:

```
CMD$="DZOM ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

HOR_MAGNIFY, HOR_POSITION, MULTI_ZOOM

STATUS

ENABLE_KEY, EKEY

Command/ Query

DESCRIPTION

The ENABLE_KEY command allows you to assign menus to the lower six menu buttons for use in local mode.

COMMAND SYNTAX

ENABLE_KEY <state>

<state> : = { ON, OFF}

QUERY SYNTAX

ENABLE_KEY?

RESPONSE FORMAT

ENABLE_KEY <state>

EXAMPLE (GPIB)

The following enables the menus assigned to the soft keys:

```
CMD$="ENABLE_KEY": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

KEY

STATUS

*ESE

Command/ Query

DESCRIPTION

The *ESE command sets the Standard Event Status Enable register (ESE). This command allows one or more events in the ESR register to be reflected in the ESB summary message bit (bit 5) of the STB register. For an overview of the ESB defined events, refer to the ESR table on page 127.

The *ESE? query reads the contents of the ESE register.

COMMAND SYNTAX

*ESE <value>

<value> := 0 to 255

QUERY SYNTAX

*ESE?

RESPONSE FORMAT

*ESE <value>

EXAMPLE (GPIB)

The following allows the ESB bit to be set if a user request (URQ bit 6, i.e. decimal 64) and/or a device dependent error (DDE bit 3, i.e. decimal 8) occurs. Summing these values yields the ESE register mask 64+8=72.

```
CMD$="*ESE 72": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

*ESR

STATUS

*ESR?
Query

DESCRIPTION

The *ESR? query reads and clears the contents of the Event Status Register (ESR). The response represents the sum of the binary values of the register bits 0 to 7. The table below gives an overview of the ESR register structure.

QUERY SYNTAX

```
*ESR?
```

RESPONSE FORMAT

```
*ESR <value>
```

<value> := 0 to 255

EXAMPLE (GPIB)

The following reads and clears the contents of the ESR register:

```
CMD$="*ESR?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
*ESR 0
```

RELATED COMMANDS

ALL_STATUS, *CLS, *ESE

ADDITIONAL INFORMATION

STANDARD EVENT STATUS REGISTER (ES)					
Bit	**Bit Value**	**Bit Name**	**Description**		**See Note ...**
15...8			0	Reserved by IEEE 488.2	
7	128	PON	1	Power off-to-ON transition has occurred	1.
6	64	URQ	1	User ReQuest has been issued	2.
5	32	CME	1	CoMmand parser Error has been detected	3.
4	16	EXE	1	EXecution Error detected	4.
3	8	DDE	1	Device Dependent (specific) Error occurred	5.
2	4	QYE	1	QuerY Error occurred	6.
1	2	RQC	0	Oscilloscope never ReQuests bus Control	7.
0	1	OPC	0	OPeration Complete bit not used	8.

NOTE: (refer to table above)

1. ***The Power On (PON) bit is always turned on (1) when the unit is powered up.***
2. ***The User Request (URQ) bit is set true (1) when a soft key is pressed. An associated register URR identifies which key was selected. For further details refer to the URR? query.***
3. ***The CoMmand parser Error bit (CME) is set true (1) whenever a command syntax error is detected. The CME bit has an associated CoMmand parser Register (CMR) which specifies the error code. Refer to the query CMR? for further details.***
4. ***The EXecution Error bit (EXE) is set true (1) when a command cannot be executed due to some device condition (e.g. oscilloscope in local state) or a semantic error. The EXE bit has an associated Execution Error Register (EXR) which specifies the error code. Refer to query EXR? for further details.***

5. *The Device specific Error (DDE) is set true (1) whenever a hardware failure has occurred at power-up, or execution time, such as a channel overload condition, a trigger or a timebase circuit defect. The origin of the failure can be localized via the DDR? or the self test *TST? query.*

6. *The QuerY Error bit (QYE) is set true (1) whenever (a) an attempt is made to read data from the Output Queue when no output is either present or pending, (b) data in the Output Queue has been lost, (c) both output and input buffers are full (deadlock state), (d) an attempt is made by the controller to read before having sent an <END>, (e) a command is received before the response to the previous query was read (output buffer flushed).*

7. *The ReQuest Control bit (RQC) is always false (0), as the oscilloscope has no GPIB controlling capability.*

8. *The OPeration Complete bit (OPC) is set true (1) whenever *OPC has been received, since commands and queries are strictly executed in sequential order. The oscilloscope starts processing a command only when the previous command has been entirely executed.*

STATUS

EXR?
Query

DESCRIPTION

The EXR? query reads and clears the contents of the EXecution error Register (EXR). The EXR register specifies the type of the last error detected during execution. Refer to the table next page.

QUERY SYNTAX

```
EXR?
```

RESPONSE FORMAT

```
EXR <value>
```

<value> : = 21 to 64

EXAMPLE (GPIB)

The following reads the contents of the EXR register:

```
CMD$="EXR?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message (if no fault):

```
EXR 0
```

RELATED COMMANDS

```
ALL_STATUS, *CLS
```

ADDITIONAL INFORMATION

EXECUTION ERROR STATUS REGISTER STRUCTURE (EXR)	
Value	**Description**
21	Permission error. The command cannot be executed in local mode.
22	Environment error. The oscilloscope is not configured to correctly process a command. For instance, the oscilloscope cannot be set to RIS at a slow timebase.
23	Option error. The command applies to an option which has not been installed.
24	Unresolved parsing error.
25	Parameter error. Too many parameters specified.
26	Non-implemented command.
30	Hex data error. A non-hexadecimal character has been detected in a hex data block.
31	Waveform error. The amount of data received does not correspond to descriptor indicators.
32	Waveform descriptor error. An invalid waveform descriptor has been detected.
33	Waveform text error. A corrupted waveform user text has been detected.
34	Waveform time error. Invalid RIS or TRIG time data has been detected.
35	Waveform data error. Invalid waveform data have been detected.
36	Panel setup error. An invalid panel setup data block has been detected.
50	No mass storage present when user attempted to access it. *
51	Mass storage not formatted when user attempted to access it. *
53	Mass storage was write protected when user attempted to create a file, to delete a file, or to format the device. *
54	Bad mass storage detected during formatting. *
55	Mass storage root directory full. Cannot add directory. *
56	Mass storage full when user attempted to write to it. *
57	Mass storage file sequence numbers exhausted (999 reached). *
58	Mass storage file not found. *
59	Requested directory not found. *
61	Mass storage filename not DOS compatible, or illegal filename. *
62	Cannot write on mass storage because filename already exists. [1]

[1] Only with memory card or removable hard disk option.

DISPLAY

FAT_CURSOR, FATC

Command/ Query

DESCRIPTION

FAT_CURSOR controls the width of the cursors.

COMMAND SYNTAX

FAT_CURSOR,FATC <state>
<state> : = { ON, OFF}

QUERY SYNTAX

FAT_CURSOR?

RESPONSE FORMAT

FAT_CURSOR <state>

EXAMPLE (GPIB)

The following sets the cursor appearance to fat:

```
CMD$="FAT_CURSOR ON": CALL IBWRT(SCOPE%,CMD$)
```

MASS STORAGE

FILENAME, FLNM

Command/ Query

DESCRIPTION

The FILENAME command is used to change the default filename given to any traces, setups and hard copies when they are being stored to a mass storage device.

COMMAND SYNTAX

FiLeNaMe TYPE,<type>,FILE,'<filename>'

<type> : = { C1, C2, C3, C4, TA, TB, TC, TD, SETUP, HCOPY }

<filename> : = For C1 to TD, an alphanumeric string of up to eight characters forming a legal DOS filename. Up to five characters for SETUP and HCOPY.

NOTE: No extension can be specified, as the oscilloscope automatically does this.

QUERY SYNTAX

FiLeNaMe? TYPE,<type>
<type> : = { ALL, C1, C2, C3, C4, TA, TB, TC, TD, SETUP, HCOPY}

RESPONSE FORMAT

FiLeNaMe
TYPE,<type>,FILE,"<filename>"[,TYPE,<type>,FILE,"<filename>"...]

AVAILABILITY

<trace> : = { C3, C4} available only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following designates channel 1 waveform files as TESTPNT6.xxx" where xxx is a numeric extension assigned by the scope:

```
CMD$="FLNM TYPE,C1, FILE, 'TESTPNT6'":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DIRECTORY, FORMAT_CARD, FORMAT_FLOPPY, FORMAT_HDD, DELETE_FILE

FUNCTION — FIND_CTR_RANGE, FCR

Command

DESCRIPTION

The FIND_CTR_RANGE command automatically sets the center and width of a histogram to best display the accumulated events.

COMMAND SYNTAX

<function> : Find_Ctr_Range

<function> : = { TA,TB,TC,TD}

AVAILABILITY

Only available with an option installed that includes Histograms.

EXAMPLE (GPIB)

Assuming that Trace A (TA) has been defined as a histogram of one of the custom parameters, the following example will determine the best center and width and then rescale the histogram:

```
CMD$="TA:FCR": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DEFINE, PARAMETER_CUSTOM

MASS STORAGE

FORMAT_CARD, FCRD

Command/Query

DESCRIPTION

The FORMAT_CARD command formats the memory card according to the PCMIA/JEIDA standard with a DOS partition.

The FORMAT_CARD? query returns the status of the card.

COMMAND SYNTAX

```
Format_CaRD
```

QUERY SYNTAX

```
Format_CaRD?
```

RESPONSE FORMAT

Format_CaRD <card_status>[,<read/write>,<free_space>,<card_size>,<battery_status>]

<card_status> := {NONE, BAD, BLANK, DIR_MISSING, OK}

<read/write> := {WP, RW}

<free_space> := A decimal number giving the number of bytes still available on the card

<card_size> := A decimal number giving the total number of bytes on the card.

<battery_status> := {BAT_OK, BAT_LOW, BAT_BAD}

AVAILABILITY

Available only with the Memory Card option installed.

EXAMPLE (GPIB)

The following will first format a memory card and then verify its status:

```
CMD$="FCRD": CALL IBWRT(SCOPE%,CMD$)
CMD$="FCRD?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
FCRD OK,RW,130048,131072,BAT_OK
```

RELATED COMMANDS

DIRECTORY

ADDITIONAL INFORMATION

NOTATION	
BAD	Bad card after formatting
BAT_BAD	Bad battery or no battery
BAT_LOW	Battery should be replaced
BAT_OK	Battery is in order
BLANK	Current directory empty
DIR_MISSING	No subdirectory present. The directory “LECROY1_DIR” will be automatically created with the next “store” command
NONE	No card
OK	Card is correctly formatted
RW	Read/Write authorized
WP	Write protected

MASS STORAGE

FORMAT_FLOPPY, FFLP

Command/ Query

DESCRIPTION

The FORMAT_FLOPPY command formats a floppy disk in the Double Density or High Density format.

The FORMAT_FLOPPY? query returns the status of the floppy disk.

COMMAND SYNTAX

Format_FLoPpy [<type>]

<type> : = { DD, HD}

If no argument is supplied, HD is used by default.

QUERY SYNTAX

Format_FLoPpy?

RESPONSE FORMAT

Format_FloPpy <floppy_status>[,<read/ write>,<free_space>, <floppy_size>]

<floppy_status> : = { NONE, BAD, BLANK, DIR_MISSING, OK}

<read/ write> : = { WP, RW}

<free_space> : = A decimal number giving the number of bytes still available on the floppy.

<floppy_size> : = A decimal number giving the total number of bytes on the floppy.

EXAMPLE (GPIB)

The following will first format a floppy in the Double Density (720 kB) format and then verify its status:

```
CMD$="FFLP DD":IBWRT(SCOPE%,CMD$)CMD$="FFLP?":
CALL IBWRT(SCOPE%,CMD$):

CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
FFLP OK,RW,728064,737280,
```

RELATED COMMANDS

DIRECTORY

ADDITIONAL INFORMATION

NOTATION	
BAD	Bad floppy after formatting
BLANK	Current directory empty
DD	Double Density 720 kB formatted
DIR_MISSING	No subdirectory present. The directory “LECROY1_DIR” will be automatically created with the next “store” command.
HD	High Density 1.44 MB formatted
NONE	No floppy
OK	Floppy is correctly formatted
RW	Read/Write authorized
WP	Write protected

MASS STORAGE

FORMAT_HDD, FHDD
Command/ Query

DESCRIPTION

The FORMAT_HDD command formats the removable hard disk according to the PCMIA/ JEIDA standard with a DOS partition.

The FORMAT_HDD? query returns the status of the hard disk.

COMMAND SYNTAX

Format_HDD <type>

<type> : = { QUICK, FULL} . If no argument is given, QUICK is used.

QUERY SYNTAX

Format_HDD?

RESPONSE FORMAT

Format_HDD <hdd_status>[,<read/ write>,<free_space>,

<hdd_size>]

<hdd_status> : = { NONE, BAD, BLANK, DIR_MISSING, OK}
<read/ write> : = { WP, RW}
<free_space> : = A decimal number giving the number of byte still available on the hard disk
<hdd_size> : = A decimal number giving the total number of bytes on the hard disk.

AVAILABILITY

Available only when the removable hard disk option is fitted.

EXAMPLE (GPIB)

The following will first format a hard disk and then verify its status:

```
CMD$="FHDD": CALL IBWRT(SCOPE%,CMD$)
CMD$="FHDD?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
FHDD OK,RW,3076096,105744896
```

RELATED COMMANDS DIRECTORY

ADDITIONAL INFORMATION

NOTATION	
BAD	Bad hard disk after formatting
BLANK	Current directory empty
DIR_MISSING	No subdirectory present. The directory “LECROY1_DIR” will be automatically created with the next “store” command
NONE	No hard disk
OK	Hard disk is correctly formatted
RW	Read/Write authorized
WP	Write protected

MASS STORAGE

FORMAT_VDISK, FVDISK

Command/ Query

DESCRIPTION

The FORMAT_VDISK command formats the non-volatile RAM (virtual disk).

The FORMAT_HDD? query returns the status of the VDISK.

COMMAND SYNTAX

Format_VDISK <type>

<type> : = { QUICK, FULL} . If no argument is given, QUICK is used.

QUERY SYNTAX

Format_VDISK?

RESPONSE FORMAT

Format_VDISK <hdd_status>[,<read/ write>,<free_space>,

<hdd_size>]

<hdd_status> : = { NONE, BAD, BLANK, DIR_MISSING, OK}
<read/ write> : = { WP, RW}
<free_space> : = A decimal number giving the number of byte still available on the hard disk
<hdd_size> : = A decimal number giving the total number of bytes on the hard disk.

AVAILABILITY

Available on all *WavePro* scopes.

EXAMPLE (GPIB)

The following will first format the virtual disk and then verify its status:

```
CMD$="FVDISK": CALL IBWRT(SCOPE%,CMD$)
CMD$="FVDISK?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
VDISK OK,RW,3076096,105744896
```

DISPLAY

FULL_SCREEN, FSCR

Command/ Query

DESCRIPTION

The FULL_SCREEN command is used to control whether the currently selected grid style is displayed in normal presentation format or with a full-screen grid. In Full Screen format, the waveform display areas are enlarged to the maximum possible size.

The response to the FULL_SCREEN? query indicates whether or not the display is operating in Full Screen presentation format.

COMMAND SYNTAX

FullSCReen <state>

<state> : = {ON, OFF}

QUERY SYNTAX

FullSCReen?

RESPONSE FORMAT

FullSCReen <state>

EXAMPLE (GPIB)

The following enables the Full Screen presentation format:

```
CMD$="FSCR ON": CALL IBWRT(SCOPE%,CMD$)
```

FUNCTION

FUNCTION_RESET, FRST
Command

DESCRIPTION

The FUNCTION_RESET command resets a waveform processing function. The number of sweeps will be reset to zero and the process restarted.

COMMAND SYNTAX

<function> : Function_ReSeT

EXAMPLE (GPIB)

<function> : = { TA,TB,TC,TD}

Assuming that Trace A (TA) has been defined as the summed average of Channel 1, the following will restart the averaging process:

```
CMD$="TA:FRST": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DEFINE, INR

ACQUISITION

GLOBAL_BWL, GBWL

Command/Query

DESCRIPTION

The GLOBAL_BWL command turns on or off the Global Bandwidth Limit. When on, the selected bandwidth limit will apply to all channels; when off, a bandwidth limit can be set individually for each channel (see BWL, page 71). The response to the GLOBAL_BWL? query indicates whether the Global Bandwidth Limit is on or off.

COMMAND SYNTAX

Global_BWL <mode>

<mode> := {OFF, ON}

QUERY SYNTAX

Global_BWL?

RESPONSE FORMAT

Global_BWL <mode>

EXAMPLE

The following deactivates the Global Bandwidth Limit, allowing a bandwidth limit to be set individually for each channel (using the BWL command syntax for individual channels):

```
CMD$="GBWL OFF": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

BANDWIDTH_LIMIT

DISPLAY

GRID

Command/ Query

DESCRIPTION

The GRID command defines the style of the grid used in the display.

The GRID? query returns the grid style currently in use.

COMMAND SYNTAX

GRID <grid>

In standard display mode:

<grid> : = { SINGLE, DUAL, QUAD, OCTAL}

In XY display mode:

<grid> : = { SINGLE, DUAL, XYONLY}

Note: The display must be in XY mode. The GRID command will not work if the display is in STANDARD mode. Send XYDIS to change the display to XY mode.

QUERY SYNTAX

GRID?

RESPONSE FORMAT

GRID <grid>

EXAMPLE (GPIB)

The following sets the screen display to dual grid mode:

```
CMD$="GRID DUAL": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

XY_DiSplay

HARD COPY

HARDCOPY_SETUP, HCSU
Command/ Query

DESCRIPTION

The HARDCOPY_SETUP command configures the oscilloscope's hard-copy driver. It enables you to specify the device type and transmission mode of the hard-copy unit connected to the oscilloscope. One or more individual settings can be changed by specifying the appropriate keyword(s), together with the new value(s). See following pages for command notation and printer or plotter model availability.

COMMAND SYNTAX

HardCopy_SetUp DEV,<device>,PORT,<port>,PFEED,
<page_feed>,PENS,<plot_pens>, PSIZE,<paper_size> CMDIV,
<cmdiv>,AUTO,<auto>,FORMAT,<format>,BCKG,<bckg>

<device> := {BMP, BMPCOMP,CANONCOL, EPSON, EPSONCOL, HPDJ, HPDJBW, HPPJ, HPTJ, HPLJ, HP7470A, HP7550A, TIFF, TIFFCOL TIFFCOMP}

<port> := {GPIB, RS, CENT, FLPY, CARD, HDD, PRT, NETPRT}

<page_feed> := {OFF, ON}

<plot_pens> := 1 to 8

<paper_size> := {A5, A4}

<cmdiv> := {1, 2, 5, 10, 20, 50, 100, 200}

<auto> := {OFF, ON}

<format> := {PORTRAIT, LANDSCAPE}

<bckg> := {BLACK, WHITE}

QUERY SYNTAX

HardCopy_SetUp?

RESPONSE FORMAT

HardCopy_SetUp DEV,<device>, PORT,<port>, PFEED,
<page_feed>,PENS,<plot_pens>, PSIZE,<paper_size>, CMDIV,
<cmdiv>,AUTO,<auto>,FORMAT,<format>,BCKG,<bckg>

AVAILABILITY

<card> : CARD only with Memory Card option installed.

<port> : HDD only with removable Hard Disk option installed.
<port> : PRT only with internal graphics printer installed.
<cmdiv> only with internal graphics printer installed.
<auto> only with internal graphics printer installed.
<device> See table below.

EXAMPLE (GPIB)

The following example selects an EPSON printer connected via the RS232 port:

```
CMD$="HCSU PORT,RS,DEV,EPSON"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

HARDCOPY_TRANSMIT, SCREEN_DUMP

ADDITIONAL INFORMATION

Hard-copy command parameters are grouped in pairs. The first in the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs can be given in any order and restricted to those variables to be changed.

The table below lists the printer and graphic formats you can use for producing hardcopies remotely using <device>.

NOTATION	PRINTER, PLOTTER OR PROTOCOL
BMP	BMP
BMPCOMP	BMP compressed
CANONCOL	Canon 200/600/800 Series color printers
EPSON	Epson b & w
EPSONCOL	Epson color
HPLJ	HP LaserJet
HPDJ	HP Desk Jet color
HPDJBW	HP Desk Jet b & w
HP7470A	HP 7470A plotter
HP7550A	HP 7550A plotter
HPGL	Vector screen file
TIFF	TIFF
TIFFCOL	TIFF color

The table below gives the HARDCOPY command notations and their meanings.

HARD COPY COMMAND NOTATION	
DEV	Device
PENS	Plotter: plot pens
PFEED	Page feed
PORT	Transmission mode
CARD	Memory Card
HDD	Hard Disk
CENT	Centronics port
FLPY	Floppy disk
GPIB	IEEE-488 port
NETPRT	Network Printer
PRT	Internal printer
RS	RS-232-C port
CMDIV	Internal printer: cm/ division
PSIZE	Plotter: paper size
AUTO	Auto print
FORMAT	Orientation of print: Portrait or Landscape

HARD COPY

HARDCOPY_TRANSMIT, HCTR
Command

DESCRIPTION

The HARDCOPY_TRANSMIT command sends a string of ASCII characters without modification to the hard-copy unit. This allows you to control the hard-copy unit by sending device-specific control character sequences. It also allows placing of additional text on a screen dump for documentation purposes.

COMMAND SYNTAX

HardCopy_TRansmit '<string>'

<string> : = Any sequence of ASCII characters or escape sequences.

NOTE: This command accepts the escape sequences as described under the command COMM_RS232. Before sending the string to the hard-copy unit, the escape sequence is converted to the ASCII character code.

EXAMPLE (GPIB)

The following sends documentation data to a printer:

```
CMD$="HCTR 'Data from Oct.15\r\n'" CALL
IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

HARDCOPY_SETUP, SCREEN_DUMP

DISPLAY

HOR_MAGNIFY, HMAG

Command/ Query

DESCRIPTION

The HOR_MAGNIFY command horizontally expands the selected expansion trace by a specified factor. Magnification factors not within the range of permissible values will be rounded off to the closest legal value.

If multiple zoom is enabled, the magnification factor for all expansion traces is set to the specified factor. If the specified factor is too large for any of the expanded traces (depending on their current source), it is reduced to an acceptable value and only then applied to the traces.

The VAB bit (bit 2) in the STB register (see table on page 224) is set when a factor outside the legal range is specified.

The HOR_MAGNIFY? query returns the current magnification factor for the specified expansion function.

COMMAND SYNTAX

<exp_trace> : Hor_MAGnify <factor>

<exp_trace> : = { TA, TB, TC, TD}

<factor> : = 1 to 20000

QUERY SYNTAX

<exp_source> : Hor_MAGnify?

RESPONSE FORMAT

<exp_source> : Hor_MAGnify <factor>

EXAMPLE (GPIB)

The following horizontally magnifies Trace A (TA) by a factor of 5:

```
CMD$="TA:HMAG 5": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DUAL_ZOOM, MULTI_ZOOM

DISPLAY

HOR_POSITION, HPOS

Command/ Query

DESCRIPTION

The HOR_POSITION command horizontally positions the geometric center of the intensified zone on the source trace. Allowed positions range from division 0 through 10. If the source trace was acquired in sequence mode, horizontal shifting will only apply to a single segment at a time.

If the multiple zoom is enabled, the difference between the specified and the current horizontal position of the specified trace is applied to all expanded traces. If this would cause the horizontal position of any expanded trace to go outside the left or right screen boundaries, the difference of positions is adapted and then applied to the traces.

If the sources of expanded traces are sequence waveforms, and the multiple zoom is enabled, the difference between the specified and the current segment of the specified trace is applied to all expanded traces. If this would cause the segment of any expanded trace to go outside the range of the number of source segments, the difference is adapted and then applied to the traces.

The VAB bit (bit 2) in the STB register (see table on page 224) is set if a value outside the legal range is specified.

The HOR_POSITION? query returns the position of the geometric center of the intensified zone on the source trace.

NOTE: Segment number 0 has the special meaning "Show All Segments Unexpanded".

COMMAND SYNTAX

<exp_trace> : Hor_POSition <hor_position>,
<exp_trace> : = { TA, TB, TC, TD}
<hor_position> : = 0 to 10 DIV
 : = 0 to max segments

NOTE: The segment number is only relevant for waveforms acquired in sequence mode; it is ignored in single waveform acquisitions. When the segment number is set to 0, all segments will be shown.

The suffix DIV is optional.

QUERY SYNTAX

<exp_trace> : Hor_POSition?

RESPONSE FORMAT

<exp_trace> : Hor_POSition <hor_position>[,]

NOTE: The segment number is only given for sequence waveforms.

EXAMPLE (GPIB)

The following positions the center of the intensified zone on the trace currently viewed by Trace A (TA) at division 3:

```
CMD$="TA:HPOS 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DUAL_ZOOM, MULTI_ZOOM

MISCELLANEOUS

*IDN?
Query

DESCRIPTION

The *IDN? query is used for identification purposes. The response consists of four different fields providing information on the manufacturer, the scope model, the serial number and the firmware revision level.

QUERY SYNTAX

*IDN?

RESPONSE FORMAT

*IDN LECROY,<model>,<serial_number>,<firmware_level>

<model> := A six- or seven-character model identifier
<serial_number> := A nine- or 10-digit decimal code
<firmware_level> := two digits giving the major release level followed by a period, then one digit giving the minor release level followed by a period and a single-digit update level (xx.y.z)

EXAMPLE (GPIB)

This issues an identification request to the scope:

```
CMD$="*IDN?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
*IDN LECROY,WP950VL,WP01000,7.7.0
```

STATUS

INE

Command/ Query

DESCRIPTION

The INE command sets the INternal state change Enable register (INE). This command allows one or more events in the INR register to be reflected in the INB summary message bit (bit 0) of the STB register. For an overview of the INR defined events, refer to the table next page.

The INE? query reads the contents of the INE register.

COMMAND SYNTAX

INE <value>

<value> := 0 to 65535

QUERY SYNTAX

INE?

RESPONSE FORMAT

INE <value>

EXAMPLE (GPIB)

The following allows the INB bit to be set whenever a screen dump has finished (bit 1, i.e. decimal 2), or a waveform has been acquired (bit 0, i.e. decimal 1), or both of these. Summing these two values yields the INE mask 2+1=3.

```
CMD$="INE 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

INR?

STATUS

INR?
Query

DESCRIPTION

The INR? Query reads and clears the contents of the INternal state change Register (INR). The INR register (table below) records the completion of various internal operations and state transitions.

INTERNAL STATE REGISTER STRUCTURE (INR)

Bit	Bit Value		Description
15		0	Reserved for future use.
14	16384	1	Probe was changed.
13	8192	1	Trigger is ready.
12	4096	1	Pass/Fail test detected desired outcome.
11	2048	1	Waveform processing has terminated in Trace D.
10	1024	1	Waveform processing has terminated in Trace C.
9	512	1	Waveform processing has terminated in Trace B.
8	256	1	Waveform processing has terminated in Trace A.
7	128	1	A memory card, floppy or hard disk exchange has been detected.
6	64	1	Memory card, floppy or hard disk has become full in AutoStore Fill mode.
5	32	0	Reserved for LeCroy use.
4	16	1	A segment of a sequence waveform has been acquired in acquisition memory but not yet read out into the main memory.
3	8	1	A time-out has occurred in a data block transfer.
2	4	1	A return to the local state is detected.
1	2	1	A screen dump has terminated.
0	1	1	A new signal has been acquired in acquisition memory and read out into the main memory.

QUERY SYNTAX

`INR?`

RESPONSE FORMAT

`INR` <state>

<state> : = 0 to 65535

EXAMPLE (GPIB)

The following reads the contents of the INR register:

```
CMD$="INR?": CALL IBWRT(SCOPE%,CMD$)
```

Response message:

```
INR 1026
```

i.e. waveform processing in Function C and a screen dump have both terminated.

RELATED COMMANDS

`ALL_STATUS, *CLS, INE`

WAVEFORM TRANSFER

INSPECT?, INSP?

Query

DESCRIPTION

The INSPECT? query allows you to read parts of an acquired waveform in intelligible form. The command is based on the explanation of the format of a waveform given by the template (use the TEMPLATE? query to obtain an up-to-date copy).

Any logical block of a waveform can be inspected using this query by giving its name enclosed in quotes as the first (string) parameter (see the template itself).

The special logical block named WAVEDESC can also be inspected in more detail. By giving the name of a variable in the block WAVEDESC, enclosed in quotes as the first (string) parameter, it is possible to inspect only the actual value of that variable. See Chapter 4 for more on INSPECT?.

NOTATION	
BYTE	raw data as integers (truncated to 8 most significant bits)
FLOAT	normalized data (gain, offset applied) as floating point numbers (gives measured values in volts or units)
WORD	raw data as integers (truncated to 16 most significant bits)

QUERY SYNTAX

<trace> : INSPect? '<string>'[,<data_type>]

<trace> : = { TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3, C4}

<data_type> : = { BYTE, WORD, FLOAT}

NOTE: The optional parameter <data_type> applies only for inspecting the data arrays. It selects the representation of the data. The default <data_type> is FLOAT.

RESPONSE FORMAT

<trace> : INSPect "<string>"

<string> : = A string giving name(s) and value(s) of a logical block or a variable.

AVAILABILITY

<trace> : = { C3, C4} only on four-channel oscilloscopes.

EXAMPLES (GPIB)

The following reads the value of the timebase at which the last waveform in Channel 1 was acquired:

```
CMD$="C1:INSP? 'TIMEBASE'"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RSP$)
PRINT RSP$
```

Response message:

```
C1:INSP "TIMEBASE: 500 US/DIV"
```

The following reads the entire contents of the waveform descriptor block:

```
CMD$="C1:INSP? 'WAVEDESC'"
```

RELATED COMMANDS

TEMPLATE, WAVEFORM_SETUP

DISPLAY

INTENSITY, INTS

Command/ Query

DESCRIPTION

The INTENSITY command sets the intensity level of the grid, or the trace or text.

The intensity level is expressed as a percentage (PCT). A level of 100 PCT corresponds to the maximum intensity while a level of 0 PCT sets the intensity to its minimum value.

The response to the INTENSITY? query indicates the grid and trace intensity levels.

COMMAND SYNTAX

INTenSity GRID,<value>,TRACE,<value>

<value> : = 0 to 100 [PCT]

NOTE: Parameters are grouped in pairs. The first of the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs can be given in any order and be restricted to those variables to be changed.

The suffix PCT is optional.

QUERY SYNTAX

INTenSity?

RESPONSE FORMAT

INTenSity TRACE,<value>,GRID,<value>

EXAMPLE (GPIB)

The following enables remote control of the intensity, and changes the grid intensity level to 75 %:

```
CMD$="INTS GRID,75": CALL IBWRT(SCOPE%,CMD$)
```

ACQUISITION

INTERLEAVED, ILVD

Command/ Query

DESCRIPTION

The INTERLEAVED command enables or disables random interleaved sampling (RIS) for timebase settings where both single shot and RIS mode are available. See the specifications in the Operator's Manual.

RIS is not available for sequence mode acquisitions.

The response to the INTERLEAVED? query indicates whether the oscilloscope is in RIS mode.

COMMAND SYNTAX

InterLeaVeD <mode>

<mode> := { ON, OFF}

QUERY SYNTAX

InterLeaVeD?

RESPONSE FORMAT

InterLeaVeD <mode>

EXAMPLE

The following instructs the oscilloscope to use RIS mode:

```
CMD$="ILVD ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TIME_DIV, TRIG_MODE, MEMORY_SIZE

STATUS

*IST?
Query

DESCRIPTION

The *IST? (Individual STatus) query reads the current state of the IEEE 488.1-defined "ist" local message. The "ist" individual status message is the status bit sent during a parallel poll operation.

QUERY SYNTAX

```
*IST?
```

RESPONSE FORMAT

```
*IST <value>
```

<value> := 0 or 1

EXAMPLE (GPIB)

The following cause the contents of the IST bit to be read:

```
CMD$="*IST?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message

```
*IST 0
```

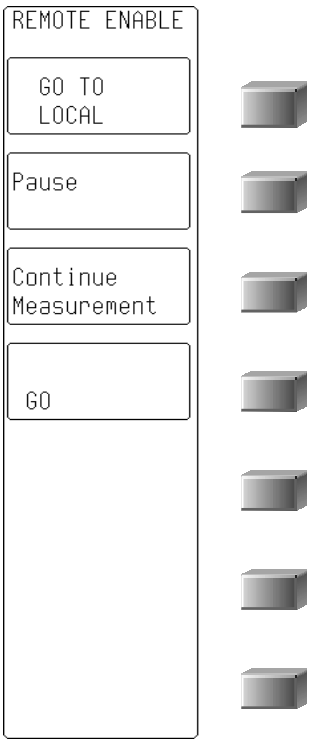
RELATED COMMANDS

*PRE

DISPLAY

KEY
Command

DESCRIPTION



The KEY command allows control of a program from the *WavePro* DSO front panel. You can display one or two lines of 13 characters each in menus corresponding to the lower six menu buttons (soft keys). The top menu is reserved for GO TO LOCAL.

You can also assign an executable file, such as AUTOEXEC.DSO, to a menu button. AUTOEXEC.DSO is a text file containing remote commands. If the file is present, the scope will read the file from the floppy or memory card and execute the remote commands contained in it.

String text that you assign to these menus disappears when you switch to local, but reappears when you switch back to the remote state. Text is cleared at power-up, whenever you reset the oscilloscope, or when you assign an empty string to a location. For example: `KEY 2, ' '`.

Pressing any one of the menu buttons in remote mode causes the User Request status Register (URR) and the URQ bit of the Event Status Register to be set. This can generate an SRQ, provided that the service request mechanism has been enabled.

COMMAND SYNTAX

`KEY` <button>`,'`<string>`','`<string>`'`

`KEY` <button>`,'`<string>`','`<string>`','`<file_name>`'`
<button> : = 1 to 5
<string> : = Up to two 13-character strings (any ASCII code)
<file_name> : = autoexec.dso

EXAMPLE (GPIB)

The menus illustrated this page were created by issuing the following:

```
CMD$="KEY 2, 'Pause'; KEY 3,
'Continue','Measurement';
KEY 4, ' ', 'GO'": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

URR

DISPLAY

LOGO

Command/ Query

DESCRIPTION

The LOGO command controls the placement of the LeCroy logo at the top left corner of the time grid or the XY grid.

COMMAND SYNTAX

LOGO <state>

<state> := {ON, OFF}

QUERY SYNTAX

LOGO?

RESPONSE FORMAT

LOGO <state>

EXAMPLE (GPIB)

The following turns on the logo:

```
CMD$="LOGO ON": CALL IBWRT(SCOPE%,CMD$)
```

MASK

Command/ Query

DESCRIPTION

For PolyMask:

`MASK COLOR` allows you to select two colors: one for the mask and one for displaying circles around sample points outside the mask.

`MASK DISP_FILLED` selects whether the mask is filled or not. During testing, the mask will always be filled, regardless of the state of this selection.

`MASK DRAWTO` draws a line from the current position to a new position. It is a command only.

`MASK ERASE` erases the current mask.

`MASK FILL` fills the enclosed polygram from starting position. It is a command only.

`MASK MOVETO` Moves the cursor to a new position without drawing a line. It is a command only.

`SHOW_FAIL` determines if errors inside or outside the mask should be circled.

COMMAND SYNTAX

<destination>:`MASK COLOR` <mask colors>`,`<error color>

See the table of color choices under `COLOR` command

<destination>:`MASK DISP_FILLED` <state>
<state> : = { YES,NO}

<destination>:`MASK DRAWTO` <x_value>,<y_value>
<x_value> : = 0 to 10 divisions (–4 to +4 divisions for XY Plot)
<y_value> : = –4 to +4 divisions

<destination>:`MASK ERASE`

<destination>:`MASK FILL` <x_value>,<y_value>
<x_value> : = 0 to 10 divisions
<y_value> : = –4 to +4 divisions

<destination>:MASK MOVETO <x_value>,<y_value>

<x_value> : = 0 to 10 divisions

<y_value> : = –4 to +4 divisions

<destination>:SHOW_FAIL <state>,<count>
<state> : = { OFF, INSIDE, OUTSIDE}
<count> : = 1 to 1000

<destination> : = { C1, C2, C3, C4, TA, TB, TC, TD, TXY, PMXY}

QUERY SYNTAX

MASK? COLOR

MASK? DISP_FILLED

EXAMPLE (GPIB)

TA:MASK COLOR,BLUE,RED

CURSOR

MATH_LIMITS, MLIM?

Command/ Query

DESCRIPTION

This command limits averaging to the area between horizontal relative cursors for increased throughput.

COMMAND SYNTAX

<source_header_pref> : Math_LIMits <ON, OFF>

<source_header_pref> : = {TA, TB, TC, TD}

QUERY SYNTAX

MLIM?

RESPONSE FORMAT

MATH_LIMIT <state>

EXAMPLE (GPIB)

The following instruction limits summation averaging to the data between the cursors:

```
CMD$="MLIM ON"
CALL IBWRT (SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$):PRINT RD$
```

RESPONSE MESSAGE

MLIM OFF

RELATED COMMANDS

DEFINE, CURSOR_MEASURE

DISPLAY

MEASURE_GATE, MGAT

Command/ Query

DESCRIPTION

The MEASURE_GATE command is used to control whether or not the parameter measurement gate region (the region between the parameter cursors) is highlighted. Highlighting is performed by making the trace area outside the measurement gate region a neutral color.

The response to the MEASURE_GATE? query indicates whether or not the parameter measurement gate region is highlighted.

COMMAND SYNTAX

Measure_GATe <state>

<state> := {ON, OFF}

QUERY SYNTAX

Measure_GATe?

RESPONSE FORMAT

Measure_GATe <state>

EXAMPLE (GPIB)

The following highlights the measurement gate region:

```
CMD$="MGAT ON": CALL IBWRT(SCOPE%,CMD$)
```

ACQUISITION

MEMORY_SIZE, MSIZ

Command/ Query

DESCRIPTION

On most models where this command/ query is available, MEMORY_SIZE allows selection of the maximum memory length used for acquisition. See the specifications in the *Operator's Manual*.

TIP: Reduce the number of data points for faster throughput.

The MEMORY_SIZE? query returns the current maximum memory length used to capture waveforms. When the optional suffix NUM is used with the query, the response will be returned in standard numeric format.

COMMAND SYNTAX

Memory_SIZe <size>

NOTE: The oscilloscope will adapt to the closest valid <size> or numerical <value> according to available channel memory.

<size> : = { 500, 1e+3, …, 2e+6, 4e+6, 8e+6} , for example, in standard numeric format.

Or, alternatively:

= {500, 1000, 2500, 5000, 10K, 25K, 50K, 100K, 250K, 500K, 1M, 2.5M, 5M, 10M, 25M}

However, values not absolutely identical to those listed immediately above will be recognized by the scope as *numerical data* (see the table under this heading in Chapter 1). For example, the scope will recognize 1.0M as 1 millisample. But it will recognize 1.0MA as 1 megasample.

QUERY SYNTAX

Memory_SIZe? [NUM]

RESPONSE FORMAT

Memory_SIZe <size>

EXAMPLE

The following will set the oscilloscope to acquire at most 10 000 data samples per single-shot or RIS acquisition:

```
CMD$="MSIZ 10K": CALL IBWRT(SCOPE%,CMD$)
```

or `CMD$="MSIZ 10e+3": CALL IBWRT(SCOPE%,CMD$)`

RELATED COMMANDS

TDIV

DISPLAY

MESSAGE, MSG

Command/Query

DESCRIPTION

The MESSAGE command displays a string of characters in the Message Field above the grid. The string can be up to 49 characters in length. The string is displayed as long as the oscilloscope is in remote mode and no internal status message is generated. Turning the oscilloscope back to local mode deletes the message. After the next transition from local to remote the message will be redisplayed. The message is cleared at power-up, when the oscilloscope is reset, or if an empty string is sent (MSG " ").

The MESSAGE? query allows you to read the last message sent.

COMMAND SYNTAX

MeSsaGe '<string>'

<string> := A string of a maximum of 49 characters

QUERY SYNTAX

MeSsaGe?

RESPONSE FORMAT

MeSsaGe "<string>"

EXAMPLE (GPIB)

The following causes the message Connect Probe 1 to appear in the message field:

```
CMD$="MSG '*Connect Probe 1*'": CALL
IBWRT(SCOPE%,CMD$)
```

DISPLAY

MULTI_ZOOM, MZOM

Command/ Query

DESCRIPTION

With MULTI_ZOOM ON, the horizontal magnification and positioning controls apply to all expanded traces simultaneously. This command is useful if the contents of all expanded traces are to be examined at the same time.

The MULTI_ZOOM? query indicates whether multiple zoom is enabled or not.

NOTE: This command has the same effect as DUAL_ZOOM.

COMMAND SYNTAX

Multi_ZOoM <mode>

<mode> : = { ON, OFF}

QUERY SYNTAX

Multi_ZOoM?

RESPONSE FORMAT

Multi_ZOoM <mode>

EXAMPLE (GPIB)

The following example turns the multiple zoom on:

```
CMD$="MZOM ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

HOR_MAGNIFY, HOR_POSITION, DUAL_ZOOM

ACQUISITION

OFFSET, OFST

Command/ Query

DESCRIPTION

The OFFSET command allows adjustment of the vertical offset of the specified input channel.

The maximum ranges depend on the fixed sensitivity setting. See the Operator's Manual.

If an out-of-range value is entered, the oscilloscope is set to the closest possible value and the VAB bit (bit 2) in the STB register is set.

NOTE: The probe attenuation factor is not taken into account for adjusting the offset.

The suffix V is optional.

The OFFSET? query returns the DC offset value of the specified channel.

COMMAND SYNTAX

<channel> : OFfSeT <offset>

<channel> : = { C1, C2, C3, C4}

<offset> : = See the Operator's Manual for specifications.

QUERY SYNTAX

<channel> : OFfSeT?

RESPONSE FORMAT

<channel> : OFfSeT <offset>

AVAILABILITY

<channel> : { C3, C4} only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following sets the offset of Channel 2 to –3 V:

```
CMD$="C2:OFST -3V": CALL IBWRT(SCOPE%,CMD$)
```

CURSOR

OFFSET_CONSTANT, OFCT

Command/ Query

DESCRIPTION

On gain changes, this command keeps the offset fixed, either volts or divisions.

COMMAND SYNTAX

`OFfset_ConstanT` <VOLTS, DIV>

QUERY SYNTAX

OFCT?

RESPONSE FORMAT

OFCT VOLTS

EXAMPLE (GPIB)

STATUS

*OPC

Command/ Query

DESCRIPTION

The *OPC (OPeration Complete) command sets to true the OPC bit (bit 0) in the standard Event Status Register (ESR). This command has no other effect on the operation of the oscilloscope because the oscilloscope starts parsing a command or query only after it has completely processed the previous command or query.

The *OPC? query always responds with the ASCII character "1" because the oscilloscope only responds to the query when the previous command has been entirely executed.

COMMAND SYNTAX

```
*OPC
```

QUERY SYNTAX

```
*OPC?
```

RESPONSE FORMAT

```
*OPC 1
```

RELATED COMMANDS

```
*WAI
```

MISCELLANEOUS

***OPT?**
Query

DESCRIPTION

The `*OPT?` query identifies oscilloscope options: installed firmware or hardware that is additional to the standard *WavePro* DSO configuration. The response consists of a series of response fields listing all the installed options.

QUERY SYNTAX

`*OPT?`

RESPONSE FORMAT

`*OPT` <option_1>,<option_2>,..,<option_N>

<option_n> := A three- or four-character ASCII string

NOTE: If no option is present, the character 0 will be returned.

EXAMPLE (GPIB)

The following queries the installed options:

```
CMD$="*OPT?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

If, for example, the waveform processing options EMM, GP02, JTA, MC04 and WAVA and are installed, the response will be returned as:

```
*EMM, GP02, JTA, MC04, WAVA
```

Response message if no options are installed:

```
*OPT 0
```

ADDITIONAL INFORMATION

NOTATION	
GP02	Internal Printer/Centronics Option
HD02	Hard Disk Option
JTA	Jitter and Timing Analysis Option
MC02	PC Card Slot
EMM	Extended Math and Measurement Option
WAVA	WaveAnalyzer Option (includes histograms)

ACQUISITION

OPTIMIZE, OPMZ

Command/ Query

DESCRIPTION

The OPMZ command allows you to optimize readout speed by trading off channel bandwidth. The default bandwidth is 2 GHz operation. This command applies only to the WavePro 960 DSO.

COMMAND SYNTAX

OPtimiMZe { bandwidth, readout}

QUERY SYNTAX

OPMZ?

RESPONSE FORMAT

OPTIMIZE <state>

EXAMPLE (GPIB)

The following instruction selects maximum bandwidth:

```
CMD$="OPTIMIZE BANDWIDTH"
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DEFINE, CURSOR_MEASURE

SAVE/RECALL SETUP

PANEL_SETUP, PNSU
Command/ Query

DESCRIPTION

The PANEL_SETUP command complements the *SAV or *RST commands. PANEL_SETUP allows you to archive panel setups in encoded form on external storage media.

Only setup data read by the PNSU? query can be recalled into the oscilloscope. A panel setup error (see table on page 130) will be generated if the setup data block contains invalid data.

NOTE: The communication parameters (those modified by commands CFMT, CHDR, CHLP, CORD and WFSU) and the enable registers associated with the status reporting system (SRE, PRE, ESE, INE) are* not *saved by this command.

COMMAND SYNTAX

PaNel_SetUp <setup>

<setup> : = A setup data block previously read by PNSU?

QUERY SYNTAX

PaNel_SetUp?

RESPONSE SYNTAX

PaNel_SetUp <setup>

EXAMPLE (GPIB)

The following saves the oscilloscope's current panel setup in the file PANEL.SET:

```
FILE$ = "PANEL.SET": CMD$="PNSU?":
CALL IBWRT(SCOPE%,CMD$): CALL IBRDF(SCOPE%,FILE$)
```

Whereas the following recalls the front panel setup, stored previously in the file PANEL.SET, into the oscilloscope:

```
CALL IBWRTF(SCOPE%,FILE$)
```

RELATED COMMANDS

*RCL, *SAV

CURSOR

PARAMETER_CLR, PACL
Command

DESCRIPTION

The PARAMETER_CLR command clears all the current parameters from the five-line list used in the Custom and Pass/ Fail modes.

NOTE: This command has the same effect as the command PASS_FAIL_CONDITION, given without any arguments.

COMMAND SYNTAX

PArameter_CLear

RELATED COMMANDS

PARAMETER_DELETE, PARAMETER_VALUE, PASS_FAIL_CONDITION

CURSOR

PARAMETER_CUSTOM, PACU

Command/ Query

DESCRIPTION

The PARAMETER_CUSTOM command controls the parameters that have customizable qualifiers and can also be used to assign any parameter for histograms.

TIP: Use PAVA? to read the measured value of a parameter set up with PACU.

COMMAND SYNTAX

PArameter_Custom
<line>,<parameter>,<qualifier>[,<qualifier>,...]

<line> := 1 to 5

<parameter> := { a parameter from the table below or any parameter listed in the PAVA? command}

<qualifier> := Measurement qualifier(s) specific to each <param>. See below.

<param>	Definition	<qualifier> list
CUSTOMIZABLE PARAMETERS ON ALL MODELS		
DDLY	delta delay	<source1>,<source2>
PHASE	phase difference	<source1>,<edge1>,<level1>,<source2>,<edge2>,<level2>, <hysteresis>,<angular unit>
CUSTOMIZABLE PARAMETERS WITH EXTENDED MATH OPTION		
DTLEV	delta time at level	<source1>,<slope1>,<level1>,<source2>,<slope2>,<level2>, <hysteresis>
FLEV	fall at level	<source>,<high>,<low>
RLEV	rise at level	<source>,<low>,<high>
TLEV	time at level	<source>,<slope>,<level>,<hysteresis>
CUSTOMIZABLE PARAMETERS WITH WAVEANALYZER OPTION		
CALC*x*	calculated parameter results	<source1>,<optional setup of source1>,<operator>,<source2>
FWXX	full width at xx % of max	<source>,<threshold>
PCTL	percentile	<source>,<threshold>
XAPK	x position at peak	<source>,<rank>

Where:

<sourceN> : = { C1, C2, C3, C4, TA, TB, TC, TD}

<slopeN> : = { POS, NEG, FIRST}

<edgeN> : = { POS, NEG}

<clock edge> : = { POS, NEG, ALL}

<levelN>, <low>, <high> : = 1 to 99 if level is specified in percent (PCT), or

<levelN>, <low>, <high> : = Level in <sourceN> in the units of the waveform.

<delay> : = –100 PCT to 100 PCT

<freq> : = 10 to 1e9 Hz (Narrow Band center frequency)

<hysteresis> : = 0.01 to 8 divisions

<length> : = 1e–9 to 0.001 seconds

<operator> : = { ADD, SUB, MUL, DIV} [1]

<rank> : = 1 to 100

<threshold> : = 0 to 100 percent

<angular unit> = { PCT, DEG, RAD}

QUERY SYNTAX

PArameter_CUstom? <line>

RESPONSE FORMAT

PArameter_Custom <line>,<parameter>,<qualifier>[,<qualifier>,...]

AVAILABILITY

<sourceN> : = { C3, C4} only on four-channel oscilloscopes.

EXAMPLE 1

DTLEV

Command Example:

```
PACU 2,DTLEV,C1,POS,345E–3,C2,NEG,–789E–3
```

Query/Response Examples:

PACU? 2 returns:

```
PACU 2,DTLEV,C1,POS,345E–3,C2,NEG,–789E–3
```

PAVA? CUST2 returns:

```
C2:PAVA CUST2,789 NS
```

[1] For Parameter Math option

EXAMPLE 2 — **DDLY**

Command Example:

```
PACU 2,DDLY,C1,C2
```

Query/Response Examples:

PACU? 2 returns:

```
PACU 2,DDLY,C1,C2
```

PAVA? CUST2 returns:

```
C2:PAVA CUST2,123 NS
```

EXAMPLE 3 — **RLEV**

Command Example:

```
PACU 3,RLEV,C1,2PCT,67PCT
```

Query/Response Examples:

PACU? 3 returns:

```
PACU 3,RLEV,C1,2PCT,67PCT
```

PAVA? CUST3 returns:

```
C1:PAVA CUST3,23 MS
```

EXAMPLE 4 — **FLEV**

Command Example:

```
PACU 3,FLEV,C1,345E–3,122E–3
```

Query/Response Examples:

PACU? 3 returns:

```
PACU 3,FLEV,C1,345E–3,122E–3
```

PAVA? CUST3 returns:

```
C1:PAVA CUST3,23 MS
```

EXAMPLE 5 (Parameter Math) — **CALC*x***

Command Example:

```
PACU 5,CALC1,AMPL,C3,DIV,AMPL,C2
```

Query/Response Examples:

PACU? 5 returns:

```
PACU 5,CALC1,AMPL,C3,DIV,AMPL,C2
```

PAVA? CUST5 returns:

```
C2:PAVA CUST1,4.884,OK
```

RELATED COMMANDS

PARAMETER_DELETE, PARAMETER_VALUE, PASS_FAIL_CONDITION

CURSOR

PARAMETER_DELETE, PADL
Command

DESCRIPTION

The PARAMETER_DELETE command deletes a parameter at a specified line from the list of parameters used in the Custom and Pass/Fail modes.

NOTATION		
1	line 1	of Custom or Pass/Fail display
2	line 2	of Custom or Pass/Fail display
3	line 3	of Custom or Pass/Fail display
4	line 4	of Custom or Pass/Fail display
5	line 5	of Custom or Pass/Fail display

COMMAND SYNTAX

PArameter_DeLete <line>

<line> : = {1, 2, 3, 4, 5}

NOTE: This command has the same effect as the command PASS_FAIL_CONDITION <line>, given without any further arguments.

EXAMPLE (GPIB)

The following deletes the third test condition in the list:

```
CMD$="PADL 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PARAMETER_CLR, PARAMETER_VALUE, PASS_FAIL_CONDITION

CURSOR PARAMETER_STATISTICS?, PAST?

Query

DESCRIPTION

The PARAMETER_STATISTICS? query returns the current values of statistics for the specified pulse parameter mode and the result type, for all five lines of the pulse parameters display.

NOTATION	
AVG	average
CUST	custom parameters
HIGH	highest value
HPAR	horizontal standard parameters
LOW	lowest value
PARAM	parameter definition for each line
SIGMA	sigma (standard deviation)
SWEEPS	number of sweeps accumulated for each line
VPAR	vertical standard parameters

QUERY SYNTAX

PArameter_STatistics? <mode>, <result>

<mode> : = { CUST, HPAR, VPAR}

<result> : = { AVG, LOW, HIGH, SIGMA, SWEEPS, PARAM}

NOTE: If keyword PARAM is specified, the query returns the list of the five pairs <parameter_name>,<source>.

EXAMPLE (GPIB)

The following query reads the average values of the five standard vertical parameters:

```
CMD$="PAST? VPAR, AVG": CALL
IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD%
```

RESPONSE FORMAT

```
PAST VPAR, AVG, 13V, 26V, 47V, 1V, 0V
```

RELATED COMMANDS

PARAMETER_VALUE

CURSOR

PARAMETER_VALUE?, PAVA?

Query

DESCRIPTION

The PARAMETER_VALUE query returns the current value or values of the pulse waveform parameter or parameters and mask tests for the specified trace. Traces do not need to be displayed or selected to obtain the values measured by the pulse parameters or mask tests.

AVAILABLE ON ALL MODELS					
ALL	all parameters	FALL	falltime	PKPK	volts
AMPL	amplitude	FALL82	fall 80 to 20 %	PNTS	points
AREA	area	FREQ	frequency	RISE	risetime
BASE	base	MAX	maximum	RISE28	rise 20 to 80 %
CMEAN	mean for cyclic waveform	MEAN	mean	RMS	root mean square
CRMS	root mean square for cyclic part of waveform	MIN	minimum	SDEV	standard deviation
CYCL	cycles	OVSN	negative overshoot	TOP	top
DLY	delay	OVSP	positive overshoot	WID	width
DUR	duration of acquisition	PER	period		
DUTY	duty cycle	PHASE	phase difference		
CUSTOM PARAMETERS DEFINED USING PARAMETER_CUSTOM COMMAND[1]					
CUST1		CUST2		CUST3	CUST4	CUST5

[1] The numbers in the terms CUST1, CUST2, CUST3, CUST4 and CUST5 refer to the line numbers of the selected custom parameters.

AVAILABLE WITH EXTENDED MATH OR WAVEANALYZER OPTION					
AVG	average of distribution	HRMS	histogram rms value	PKS	number of peaks
CMEDI	median for cyclic waveform	HTOP	histogram top value	RANGE	range of distribution
CSDEV	standard deviation for cyclic part of waveform	LAST	last point	SIGMA	sigma of distribution
FRST	first point	MAXP	maximum population	TOTP	total population
HIGH	high of histogram	MEDI	median value		
HMEDI	median of a histogram	MODE	mode of distribution		

PARAMETER COMPUTATION STATES			
AV	averaged over several (up to 100) periods	OF	signal partially in overflow
GT	greater than given value	OK	deemed to be determined without problem
IV	invalid value (insufficient data provided)	OU	signal partially in overflow and underflow
LT	less than given value	PT	window has been period truncated
NP	no pulse waveform	UF	signal partially in underflow

MASK TEST NAMES			
ALL_IN	all points of waveform inside mask (TRUE = 1, FALSE = 0)	SOME_IN	some points of waveform inside mask (TRUE = 1, FALSE = 0)
ALL_OUT	all points of waveform outside mask (TRUE = 1, FALSE = 0)	SOME_OUT	some points of waveform outside mask (TRUE = 1, FALSE = 0)

QUERY SYNTAX

<trace> : PArameter_VAlue? [<parameter>,...,<parameter>]

<trace> : = { TA, TB, TC, TD, C1, C2, C3, C4}

<parameter> : = See table of parameters.

Alternative forms of query for mask tests:

<trace> : PArameter_VAlue? <mask_test>, <mask>

<mask_test> : = { ALL_IN, SOME_IN, ALL_OUT, SOME_OUT}

<mask> : = { TA, TB, TC, TD}

RESPONSE FORMAT

<trace> : PArameter_VAlue <parameter>,<value>,

<state> [,...,<parameter>,<value>,<state>]

<value> : = A decimal numeric value

<state> : = { OK, AV, PT, IV, NP, GT, LT, OF, UF, OU}

NOTE: If <parameter> is not specified, or is equal to ALL, all standard voltage and time parameters are returned followed by their values and states.

AVAILABILITY

<trace> : { C3, C4} only available on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following query reads the risetime of Trace B (TB):

```
CMD$="TB:PAVA? RISE": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD (SCOPE%,RD$): PRINT RD$
```

Response message:

```
TB:PAVA RISE,3.6E-9S,OK
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PARAMETER_CUSTOM, PARAMETER_STATISTICS

CURSOR

PASS_FAIL_CONDITION, PFCO

Command/Query

DESCRIPTION

The PASS_FAIL_CONDITION command adds a Pass/Fail test condition or a custom parameter at the specified line on the Pass/Fail or Custom Parameter display.

The PASS_FAIL_CONDITION? query indicates the current Pass/Fail test setup or the current selection of custom parameters at the specified line.

NOTE: Up to five test conditions (or custom parameters) can be specified at five different display lines on the screen. The command PASS_FAIL_CONDITION deals with one line at a time.

NOTATION			
GT	greater than	LT	lower than

COMMAND SYNTAX

Pass_Fail_Condition
[<line>,<trace>,<parameter>[,<rel_op> [,<ref_value>]]]

<line> : = {1,2,3,4,5}

<trace> : = {TA, TB, TC, TD, C1, C2, C3, C4}

<parameter> : = See tables of parameters on pages 178 and 183.

<rel_op> : = {GT, LT}

<ref_value> : = –1e15 to +1e15

NOTE: The PFCO command with no arguments (i.e. "PFCO") deletes all conditions. The PFCO command with a single argument (i.e. "PFCO <line>") deletes the condition at <line>.

Alternative form of command for mask tests:

Pass_Fail_COndition
[<line>,<trace>,<mask_test>,<mask>]

<mask_test> : = { ALL_IN, SOME_IN, ALL_OUT, SOME_OUT}

<mask> : = { TA, TB, TC, TD}

QUERY SYNTAX

PFCO? <line>

RESPONSE FORMAT

PFCO <line>,<trace>,<parameter>,<rel_op>,<ref_value>

Alternative form of response for mask tests:

PFCO <line>,<trace>,<mask_test>,<mask>

AVAILABILITY

<trace> : = { C3, C4} only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following sets the first test condition in the list to be "frequency on Channel 1 lower than 10 kHz":

```
CMD$="PFCO 1,C1,FREQ,LT,10000":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PASS_FAIL_COUNTER, PASS_FAIL_DO, PASS_FAIL_MASK, PARAMETER_VALUE

CURSOR

PASS_FAIL_COUNTER, PFCT

Command/ Query

DESCRIPTION

The PASS_FAIL_COUNTER command resets the Passed/ Failed acquisitions counters. The PASS_FAIL_COUNTER? query returns the current counts.

COMMAND SYNTAX

Pass_Fail_CounTer

QUERY SYNTAX

Pass_Fail_CounTer?

RESPONSE FORMAT

Pass_Fail_CounTer <pass/ fail>,<value>,OF,<value>

<value> : = 0 to 999999

<pass/ fail> : = { PASS, FAIL}

EXAMPLE (GPIB)

The following query reads the counters:

```
CMD$="PFCT?": CALL IBWRT(SCOPE%,CMD$)
```

Response message:

```
PFCT PASS, 8, OF, 9
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PASS_FAIL_DO, PASS_FAIL_MASK, PARAMETER_VALUE

CURSOR

PASS_FAIL_DO, PFDO

Command/ Query

DESCRIPTION

The PASS_FAIL_DO command defines the desired outcome and the actions that have to be performed by the oscilloscope after a Pass/ Fail test. The PASS_FAIL_DO? query indicates which actions are currently selected. The command PFDO PASS, TESTING_OFF turns off Pass/ Fail testing. **Testing Off** is then displayed below the grid.

NOTATION	
BEEP	emit a beep
PULS	emit a pulse on the CAL connector
SCDP	make a hard copy
STO	store in memory or on storage media
STOP	stop acquisition

COMMAND SYNTAX

Pass_Fail_DO [<outcome>[,<act>[,<act>...]]]

<outcome> := { PASS,FAIL}

<act> := { STOP, SCDP, STO}

NOTE:

BEEP is accepted only on models equipped with the CLBZ hardware option.

PULS is accepted only on models equipped with the CKIO software option.

PFDO without arguments deletes all actions.

STO performs the store operation as described in the Operator's Manual.

After every pass or fail detected, the oscilloscope sets the INR bit 12.

QUERY SYNTAX

Pass_Fail_DO?

RESPONSE FORMAT

`Pass_Fail_DO` [<pass_fail>[,<act>[,<act>...]]]

EXAMPLE (GPIB)

This following forces the oscilloscope to stop acquiring when the test passes:

```
CMD$="PFDO PASS,STOP": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

`BUZZER, CURSOR_MEASURE, CURSOR_SET, INR, PARAMETER_VALUE, PASS_FAIL_COUNTER, PASS_FAIL_MASK`

CURSOR

PASS_FAIL_MASK, PFMS
Command

DESCRIPTION

The PASS_FAIL_MASK command generates a tolerance mask around a chosen trace and stores the mask in the selected memory.

COMMAND SYNTAX

Pass_Fail_MaSk [<trace>[,<htol>[,<vtol>[,<mask>]]]]

<trace> : = { TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3, C4}

<htol> : = 0.0 to 5.0

<vtol> : = 0.0 to 4.0

<mask> : = { M1, M2, M3, M4}

NOTE: If any arguments are missing, the previous settings will be used.

The alternative form of command:

Pass_Fail_MaSk INVT [,<mask>]

inverts the mask in the selected mask memory. If <mask> is missing, M4 is implied.

AVAILABILITY

<trace> : = { C3, C4} only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following generates a tolerance mask around the Channel 1 trace and stores it in M2:

```
CMD$="PASS_FAIL_MASK C1,0.2,0.3,M2":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PASS_FAIL_DO, PARAMETER_VALUE

CURSOR

PASS_FAIL_STATUS?, PFST?

Query

DESCRIPTION

The PASS_FAIL_STATUS query returns the status of the Pass/Fail test for a given line number.

QUERY SYNTAX

Pass_Fail_STatus? <line>

<line> : = { 1, 2, 3, 4, 5}

RESPONSE FORMAT

Pass_Fail_STatus <line>,<state>

<state> : = { TRUE, FALSE}

EXAMPLE (GPIB)

The following queries the state of the Pass/Fail test condition specified for line 3.

```
CMD$="PFST? 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PASS_FAIL_DO, PASS_FAIL_CONDITION, PARAMETER_VALUE

CURSOR

PER_CURSOR_SET, PECS

Command/ Query

DESCRIPTION

The PER_CURSOR_SET command allows you to position any one of the six independent cursors at a given screen location. The position of the cursor can be modified or queried even if the cursor is not currently displayed on the screen.

The PER_CURSOR_SET? query indicates the current position of the cursor or cursors.

The vertical cursor positions are the same as those controlled by the CURSOR_SET command.

NOTATION			
HABS	horizontal absolute	VABS	vertical absolute
HDIF	horizontal difference	VDIF	vertical difference
HREF	horizontal reference	VREF	vertical reference

COMMAND SYNTAX

<trace> : PEr_Cursor_Set <cursor>,
<position>[,<cursor>,<position>,...,<cursor>,<position>

<trace> := { TA, TB, TC, TD, C1, C2, C3, C4}

<cursor> := { HABS, HDIF, HREF, VABS, VDIF, VREF}

<position> := 0 to 10 DIV (horizontal), –29.5 to 29.5 DIV (vertical)

NOTE: Parameters are grouped in pairs. The first of the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs can be in any order and limited to those variables to be changed.

The suffix DIV is optional.

If <cursor> is not specified, ALL will be assumed. If the position of a cursor cannot be determined in a particular situation, its position will be indicated as UNDEF.

QUERY SYNTAX

<trace> : PEr_Cursor_Set?
<cursor>[,<cursor,...,<cursor>]

<cursor> : = { HABS, HDIF, HREF, VABS, VDIF, VREF, ALL}

RESPONSE FORMAT

PEr_Cursor_Set <cursor>,<position>[,<cursor>,<position>,...,<cursor>,<position>

AVAILABILITY

<trace> : = { C3, C4} only available on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following positions the HREF and HDIF cursors at +2.6 DIV and +7.4 DIV respectively, using Channel 2 as a reference:

```
CMD$="C2:PECS HREF,2.6 DIV,HDIF,7.4 DIV"
```

RELATED COMMANDS

CURSOR_MEASURE, CURSOR_SET, PERSIST, PER_CURSOR_VALUE

CURSOR

PER_CURSOR_VALUE?, PECV?

Query

DESCRIPTION

The PER_CURSOR_VALUE? query returns the values measured by the cursors specified below while in Persistence mode.

NOTATION			
HABS	horizontal absolute	VABS	vertical absolute
HREL	horizontal relative	VREL	vertical relative

QUERY SYNTAX

<trace> : PEr_Cursor_Value?
<cursor>[,<cursor>,...,<cursor>]

<trace> := { TA, TB, TC, TD, C1, C2, C3, C4}
<cursor> := { HABS, HREL, VABS, VREL, ALL}

Note: If <cursor> is not specified, ALL will be assumed.

RESPONSE FORMAT

<trace> : PEr_Cursor_Value <cursor>,
<value>[,<cursor>,<value>,...,<cursor>,<value>]

AVAILABILITY

<trace> := { C3, C4} only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following returns the value measured with the vertical relative cursor on Channel 1:

```
CMD$="C1:PECV? VREL": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
C1:PECV VREL,56 MV
```

RELATED COMMANDS

CURSOR_MEASURE, PERSIST, PER_CURSOR_SET

DISPLAY

PERSIST, PERS

Command/ Query

DESCRIPTION

The PERSIST command enables or disables the persistence display mode.

COMMAND SYNTAX

PERSist <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

PERSist?

RESPONSE FORMAT

PERSist <mode>

EXAMPLE (GPIB)

The following turns the persistence display ON:

```
CMD$="PERS ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PERSIST_COLOR, PERSIST_LAST, PERSIST_SAT, PERSIST_SETUP

DISPLAY

PERSIST_COLOR, PECL

Command/ Query

DESCRIPTION

The PERSIST_COLOR command controls the color rendering method of persistence traces.

The response to the PERSIST_COLOR? query indicates the color rendering method, Analog Persistence™ or Color Graded Persistence.

See the Operator's Manual.

COMMAND SYNTAX

PErsist_CoLor <state>

<state> : = {ANALOG, COLOR_GRADED}

QUERY SYNTAX

PErsist_CoLor?

RESPONSE FORMAT

PErsist_CoLor <state>

EXAMPLE (GPIB)

The following sets the persistence trace color to an intensity-graded range of the selected trace color:

```
CMD$="PECL ANALOG": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

COLOR, COLOR_SCHEME, PERSIST, PERSIST_LAST, PERSIST_SAT, PERSIST_SETUP

DISPLAY

PERSIST_LAST, PELT

Command/ Query

DESCRIPTION

The PERSIST_LAST command controls whether or not the last trace drawn in a persistence data map is shown.

The response to the PERSIST_LAST? query indicates whether the last trace is shown within its persistence data map.

COMMAND SYNTAX

PErsist_LasT <state>

<state> : = {ON, OFF}

QUERY SYNTAX

PErsist_LasT?

RESPONSE FORMAT

PErsist_LasT <state>

EXAMPLE (GPIB)

The following ensures the last trace is visible within its persistence data map:

```
CMD$="PELT ON": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PERSIST, PERSIST_COLOR, PERSIST_SAT, PERSIST_SETUP

DISPLAY

PERSIST_SAT, PESA

Command/ Query

DESCRIPTION

The PERSIST_SAT command sets the level at which the color spectrum of the persistence display is saturated. The level is specified in terms of percentage (PCT) of the total persistence data map population. A level of 100 PCT corresponds to the color spectrum being spread across the entire depth of the persistence data map. At lower values, the spectrum will saturate (brightest value) at the specified percentage value. The PCT is optional.

The response to the PERSIST_SAT? query indicates the saturation level of the persistence data maps.

COMMAND SYNTAX

PErsist_SAt <trace>,<value> [<trace>,<value>]

<trace> : = { C1, C2, C3, C4, TA, TB, TC, TD}

<value> : = 0 to 100 PCT

NOTE: The suffix PCT is optional.

QUERY SYNTAX

PErsist_SAt?

RESPONSE FORMAT

PErsist_SAt <trace>,<value>

AVAILABILITY

<trace> : = { C3, C4} only on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following sets the saturation level of the persistence data map for channel 3 to be 60 % — 60 % of the data points will be displayed with the color spectrum, with the remaining 40 % saturated in the brightest color:

```
CMD$="PESA C3,60": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PERSIST, PERSIST_COLOR, PERSIST_PERS, PERSIST_SETUP

DISPLAY

PERSIST_SETUP, PESU

Command/ Query

DESCRIPTION

The PERSIST_SETUP command selects the persistence duration of the display, in seconds, in persistence mode. In addition, the persistence can be set either to all traces or only the top two on the screen.

The PERSIST_SETUP? query indicates the current status of the persistence.

COMMAND SYNTAX

PErsist_SetUp <time>,<mode>

<time> := {0.5, 1, 2, 5, 10, 20, infinite}

<mode> := {TOP2, ALL}

QUERY SYNTAX

PErsist_SetUp?

RESPONSE FORMAT

PErsist_SetUp <time>,<mode>

EXAMPLE (GPIB)

The following sets the variable persistence at 10 seconds on the top two traces:

```
CMD$="PESU 20,TOP2": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PERSIST, PERSIST_COLOR, PERSIST_PERS, PERSIST_SAT

STATUS

***PRE**
Command/ Query

DESCRIPTION

The *PRE command sets the PaRallel poll Enable register (PRE). The lowest eight bits of the Parallel Poll Register (PPR) are composed of the STB bits. *PRE allows you to specify which bit(s) of the parallel poll register will affect the 'ist' individual status bit.

The *PRE? query reads the contents of the PRE register. The response is a decimal number which corresponds to the binary sum of the register bits.

COMMAND SYNTAX

PRE <value>
<value> : = 0 to 65 535

QUERY SYNTAX

*PRE?

RESPONSE FORMAT

*PRE <value>

EXAMPLE (GPIB)

The following will cause the 'ist' status bit to become 1 as soon as the MAV bit (bit 4 of STB, i.e. decimal 16) is set, and yields the PRE value 16:

```
CMD$="*PRE 16": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

*IST

PROBES

PROBE_CAL?, PRCA?

Query

DESCRIPTION

The PROBE_CAL? query performs a complete auto-calibration of a current probe connected to your *WavePro* oscilloscope. At the end of this calibration, the response indicates how the calibration has terminated, and the oscilloscope then returns to the state it was in prior to the query.

QUERY SYNTAX

<channel> : PROBE_CAL?

RESPONSE FORMAT

PROBE_CAL <diagnostics>

<diagnostics> := 0 or 1

0 = Calibration successful

EXAMPLE (GPIB)

The following forces a self-calibration:

```
CMD$="PROBE_CAL?": CALL
IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Response message (if no failure):

```
PROBE_CAL 0
```

RELATED COMMANDS

AUTO_CALIBRATE, *CAL?, PROBE_DEGAUSS?

PROBES

PROBE_DEGAUSS?, PRDG?

Query

DESCRIPTION

The PROBE_DEGAUSS? query performs the automatic degaussing of the current probe connected to your *WavePro* oscilloscope. This eliminates core saturation by use of a backing current and application of an alternating field, reduced in amplitude over time from an initial high value. After the degaussing, a probe calibration is performed.

QUERY SYNTAX

<channel> : PROBE_DEGAUSS?

RESPONSE FORMAT

PROBE_DEGAUSS <diagnostics>

<diagnostics> : = 0 or 1

0 = Degaussing and calibration successful

Note: If coupling is not DC, probe calibration will not be performed, and the <diagnostics> response will be 1.

EXAMPLE (GPIB)

The following degausses and calibrates the connected probe:

```
CMD$="PROBE_DEGAUSS?": CALL
IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Response message (if no failure):

```
PROBE_DEGAUSS 0
```

RELATED COMMANDS

PROBE_CAL?, PROBE_NAME?

PROBES — PROBE_INFOTEXT?, PRIT?

Query

DESCRIPTION

The PROBE_INFOTEXT? query returns informative text about a probe connected to your *WavePro* oscilloscope.

QUERY SYNTAX

<channel> : PROBE_INFOTEXT?

<channel> : = { C1, C2, C3, C4,EX,EX5}

RESPONSE FORMAT

<channel> : PRIT "<info>"

EXAMPLE (GPIB)

The following instruction obtains the text for the probe connected to channel 1:

```
CMD$="C1:PROBE_INFOTEXT?":
CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

RELATED COMMANDS

PROBE_CAL? PROBE_DEGAUSS?

PROBES

PROBE_NAME?, PRNA?
Query

DESCRIPTION

The PROBE_NAME? query returns the name of a probe connected to your *WavePro* oscilloscope. Passive probes are identified by their attenuation factor.

QUERY SYNTAX

<channel> : PROBE_NAME?

RESPONSE FORMAT

<channel> : PRNA "<name>"
<name> : = { name of connected probe}

EXAMPLE (GPIB)

The following obtains an identification of the connected probe:

```
CMD$="PROBE_NAME?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

RELATED COMMANDS

PROBE_CAL?, PROBE_DEGAUSS?

SAVE/RECALL SETUP

***RCL**
Command

DESCRIPTION

The *RCL command sets the state of your *WavePro* oscilloscope, using one of the five non-volatile panel setups, by recalling the complete front panel setup of the oscilloscope. Panel setup 0 corresponds to the default panel setup.

The *RCL command produces an effect the opposite of the *SAV command.

If the desired panel setup is not acceptable, the EXecution error status Register (EXR) is set and the EXE bit of the standard Event Status Register (ESR) is set.

COMMAND SYNTAX

*RCL <panel_setup>

<panel_setup> := 0 to 4

EXAMPLE (GPIB)

The following recalls your *WavePro* oscilloscope setup previously stored in panel setup 3:

```
CMD$="*RCL 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *SAV, EXR

MISCELLANEOUS

REAR_OUTPUT, ROUT

Command/ Query

DESCRIPTION

The REAR_OUTPUT command is used to set the type of signal put out at the *WavePro* DSO rear panel BNC connector. The REAR_OUTPUT? Query returns the current mode of the connector.

COMMAND SYNTAX

Rear_OUTput <mode>[,<level>[,<rate>]]

<mode> := {OFF, PF, TRIG, TRDY, PULSE}

QUERY SYNTAX

Rear_OUTput?

RESPONSE FORMAT

Rear_OUTput <mode>,<level>[,<rate>]

EXAMPLE (GPIB)

The following turns off the BNC rear output:

```
CMD$="ROUT OFF":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PASS_FAIL_DO, CAL_OUTPUT

ADDITIONAL INFORMATION

NOTATION	
OFF	Turns off rear output
PF	Pass/Fail mode
PULSE	Provides a single pulse
TRIG	Trigger Out mode
TRDY	Trigger is ready for a new acquisition

WAVEFORM TRANSFER

RECALL, REC
Command

DESCRIPTION

The RECALL command recalls a waveform file from the current directory on mass storage into any or all of the internal memories M1 to M4.

NOTE: Only waveforms stored in BINARY format can be recalled.

COMMAND SYNTAX

<memory> : RECall DISK, <device>, FILE, '<filename>'

<memory> : = { M1, M2, M3, M4, ALL}

<device> : = { CARD, FLPY, HDD}

<filename> : = An alphanumeric string of up to eight characters, followed by a dot and an extension of up to three digits.

AVAILABILITY

<device> : CARD only available with Memory Card option installed.

<device> : HDD only available with removable Hard Disk option installed.

EXAMPLE (GPIB)

The following recalls a waveform file called "SC1.001" from the memory card into Memory M1:

```
CMD$="M1:REC DISK,CARD,FILE,'SC1.001'": CALL
IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

STORE, INR?

SAVE/RECALL SETUP

RECALL_PANEL, RCPN
Command

DESCRIPTION

The RECALL_PANEL command recalls a front panel setup from the current directory on mass storage.

COMMAND SYNTAX

ReCall_PaNel DISK,<device>,FILE,'<filename>'

<device> := { CARD, FLPY, HDD}

<filename> := A string of up to eight characters with the extension .PNL.

AVAILABILITY

<device> : CARD only available with Memory Card option installed.

<device> : HDD only available with removable Hard Disk option installed.

EXAMPLE (GPIB)

The following recalls the front panel setup from file P012.PNL on the floppy disk:

```
CMD$="RCPN DISK, FLPY, FILE,'P012.PNL'":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *SAV, STORE_PANEL, *RCL

ACQUISITION

REFERENCE_CLOCK, RCLK

Command/ Query

DESCRIPTION

REFERENCE_CLOCK selects the system clock source, allowing the instrument to be phase synchronized to an external reference clock.

COMMAND SYNTAX

`Reference_CLocK` <state>

<state> : = { INT, EXT}

QUERY SYNTAX

Reference_CLocK?

RESPONSE FORMAT

`RCLK <state>`

SAVE/RECALL SETUP

***RST**
Command

DESCRIPTION

The `*RST` command initiates a device reset. `*RST` sets all eight traces to the GND line and recalls the default setup.

COMMAND SYNTAX

```
*RST
```

EXAMPLE (GPIB)

The following resets the oscilloscope:

```
CMD$="*RST": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

`*CAL, *RCL`

ACQUISITION

SAMPLE_CLOCK, SCLK

Command/ Query

DESCRIPTION

The SAMPLE_CLOCK command allows you to control an external timebase. Set the number of data points that will be acquired when your *WavePro* oscilloscope is using the external clock. When the optional suffix NUM is used with the query, the response will be returned in standard numeric format.

COMMAND SYNTAX

Sample_CLocK <state>[,<recordlength>][,<coupling>]

<state> : = { INT, ECL, LV0, TTL}

<recordlength> : = {10e+3, 10.0e+3, 11e+3…}, for example, in standard numeric format.

Or, alternatively:
= {50, 100, 200, 500, 1K, 2K, 5K, 10K, 20K, 50K, 100K, 200K, 500K, 1M, 2M, 4M}

However, values not absolutely identical to those listed immediately above will be recognized by the scope as numerical data (see the table under this heading in Chapter 1). For example, the scope will recognize 1.0M as 1 millisample. But it will recognize 1.0MA as 1 megasample.

<coupling> : = { D1M or D50}

TIP: You cannot have the record length larger than the maximum available memory of your oscilloscope. WavePro will adapt to the closest valid <recordlength>. See the Operator's Manual for maximums.

QUERY SYNTAX

Sample_CLocK? [NUM]

RESPONSE FORMAT

Sample_CLocK <state>,<recordlength>

EXAMPLE

The following sets the oscilloscope to use the external clock with 1000-data-point records.

```
CMD$="SCLK ECL,1000": CALL IBWRT(SCOPE%,CMD$)
```

SAVE/RECALL SETUP

***SAV**
Command

DESCRIPTION

The *SAV command stores the current state of your *WavePro* oscilloscope in non-volatile internal memory. The *SAV command stores the complete front panel setup of the oscilloscope at the time the command is issued.

NOTE: Neither communication parameters (those modified by the commands COMM_FORMAT, COMM_HEADER, COMM_HELP, COMM_ORDER and WAVEFORM_SETUP), nor enable registers of the status reporting system (*SRE, *PRE, *ESE, INE), are saved when *SAV is used.

COMMAND SYNTAX

*SAV <panel_setup>
<panel_setup> := 1 to 4

EXAMPLE (GPIB)

The following saves the current *WavePro* oscilloscope setup in panel setup 3:

```
CMD$="*SAV 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *RCL

DISPLAY

SCREEN
Command

DESCRIPTION

The SCREEN ON/OFF command turns the screen on or off independently of the SCREEN_SAVE command.

NOTE: When the screen is off, the oscilloscope is still fully functional.

COMMAND SYNTAX

SCREEN <state>
<state> := { ON,OFF}

HARD COPY

SCREEN_DUMP, SCDP

Command/ Query

DESCRIPTION

The SCREEN_DUMP command causes the oscilloscope to dump the screen contents onto the hardcopy device. This command will halt your *WavePro* oscilloscope's activities.

The time/ date stamp which appears on the print-out corresponds to the time at which the command was executed.

COMMAND SYNTAX

```
SCreen_DumP
```

QUERY SYNTAX

```
SCreen_DumP?
```

RESPONSE FORMAT

SCreen_DumP <status>

<status> := {OFF}

EXAMPLE (GPIB)

The following initiates a screen dump:

```
CMD$="SCDP": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

INR, HARDCOPY_SETUP, HARDCOPY_TRANSMIT

DISPLAY

SCREEN_SAVE, SCSV

Command/ Query

DESCRIPTION

The SCREEN_SAVE command controls the automatic Screen Saver, which automatically shuts down the internal color monitor after a preset time.

The response to the SCREEN_SAVE? query indicates whether the automatic screen saver feature is on or off.

NOTE: When the screen saver is in effect, the oscilloscope is still fully functional.

COMMAND SYNTAX

SCreen_SaVe <enabled>

<enabled> : = { YES, NO}

QUERY SYNTAX

SCreen_SaVe?

RESPONSE FORMAT

SCreen_SaVe <state>

EXAMPLE (GPIB)

The following enables the automatic screen saver:

```
CMD$="SCSV YES": CALL IBWRT(SCOPE%,CMD$)
```

DISPLAY

SELECT, SEL

Command/ Query

DESCRIPTION

The SELECT command selects the specified trace for manual display control. An environment error (see table on page 130) is generated if the specified trace is not displayed.

The SELECT? query returns the selection status of the specified trace.

COMMAND SYNTAX

<trace> : SELect

<trace> := {TA, TB, TC, TD}

QUERY SYNTAX

<trace> : SELect?

RESPONSE FORMAT

<trace> : SELect <mode>

<mode> := {ON, OFF}

EXAMPLE (GPIB)

The following selects Trace B (TB):

```
CMD$="TB:SEL": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRACE

ACQUISITION

SEQUENCE, SEQ

Command/ Query

DESCRIPTION

The SEQUENCE command sets the conditions for the sequence mode acquisition. The response to the SEQUENCE? query gives the conditions for the sequence mode acquisition. The argument <max_size> can be expressed either as numeric fixed point, exponential or using standard suffixes. When the optional suffix NUM is used with the query, the response will be returned in standard numeric format.

COMMAND SYNTAX

SEQuence <mode>[,<segments>[,<max_size>]]

<mode> := {OFF, ON}

<segments> := the right-hand column in the table below:

<max_size> := {...10e+3, 10.0e+3,...11e+3,...}, for example, in standard numeric format.

Or, alternatively:

= {50, 100, 250, 500, 1000, 2500, 5K, 10K, 25K, 50K, 100K, 250K, 500K, 1M}

However, values not absolutely identical to those listed immediately above will be recognized by the scope as *numerical data* (see the table under this heading in Chapter 1). For example, the scope will recognize 1.0M as 1 millisample. But it will recognize 1.0MA as 1 megasample.

NOTE: The oscilloscope will adapt the requested <max_size> to the closest valid value.

QUERY SYNTAX

SEQuence? [NUM]

RESPONSE FORMAT

SEQuence <mode>,<segments>,<max_size>

<mode> := {ON, OFF}

EXAMPLE (GPIB)

The following sets the segment count to 43, the maximum segment size to 250 samples, and turns the sequence mode ON:

```
CMD$="SEQ ON,43,250": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_MODE

DISPLAY

SKEY
Command

DESCRIPTION

For Custom DSO applications, this command allows you to assign text, borders, and files to the menu soft keys. This command must be preceded by the enable key command EKEY ON.

COMMAND SYNTAX

SKEY KEY,<n>,FILE,<filename>,TEXT,'<text string>',TYPE,<box type>

<n> : = { 1 to 7}

<filename> : = { .DSO file extension}

<text string> : = up to 3 lines, 13 ASCII characters per line; use **\n** for a new line

<box type> : = NOBOX, BOX, SHADOW (thin line box is the default if not specified)

SKEY CLEAR,<n> deletes the specified menu box

NOTE: The command SKEY CLEAR with no arguments after it deletes all menu soft keys at the same time.

RELATED COMMANDS

EKEY, KEY, STITLE

MISCELLANEOUS

SLEEP
Command

DESCRIPTION

The SLEEP command makes the scope's remote command interpreter wait the time specified in the argument before interpreting any remote commands that follow. It is typically used to let an external signal settle before the scope performs new acquisitions and processing defined by the remote commands that follow.

COMMAND SYNTAX

SLEEP <n>

<n> := time in seconds (0 to 1000.0)

EXAMPLE (GPIB)

The following command causes the scope to sleep for 3.2 seconds.

```
CMD$="SLEEP 3.2";: CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

*TRG,WAIT

STATUS

*SRE

Command/ Query

DESCRIPTION

The *SRE command sets the Service Request Enable register (SRE). This command allows you to specify which summary message bit or bits in the STB register will generate a service request. Refer to the table on page 224 for an overview of the available summary messages.

A summary message bit is enabled by writing a '1' into the corresponding bit location. Conversely, writing a '0' into a given bit location prevents the associated event from generating a service request (SRQ). Clearing the SRE register disables SRQ interrupts.

The *SRE? query returns a value that, when converted to a binary number, represents the bit settings of the SRE register. Note that bit 6 (MSS) cannot be set and its returned value is always zero.

COMMAND SYNTAX

*SRE <value>

<value> : = 0 to 255

QUERY SYNTAX

*SRE?

RESPONSE FORMAT

*SRE <value>

EXAMPLE (GPIB)

The following allows an SRQ to be generated as soon as the MAV summary bit (bit 4, i.e. decimal 16) or the INB summary bit (bit 0, i.e. decimal 1) in the STB register, or both, are set. Summing these two values yields the SRE mask 16+1 = 17.

```
CMD$="*SRE 17": CALL IBWRT(SCOPE%,CMD$)
```

STATUS

*STB?
Query

DESCRIPTION

The *STB? query reads the contents of the 488.1 defined STatus Byte register (STB), and the Master Summary Status (MSS). The response represents the values of bits 0 to 5 and 7 of the STB register and the MSS summary message.

The response to a *STB? query is identical to the response of a serial poll except that the MSS summary message appears in bit 6 in place of the RQS message. Refer to the table on page 224 for further details of the status register structure.

QUERY SYNTAX

```
*STB?
```

RESPONSE FORMAT

*STB <value>
<value> := 0 to 255

EXAMPLE (GPIB)

The following reads the status byte register:

```
CMD$="*STB?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
*STB 0
```

RELATED COMMANDS

ALL_STATUS, *CLS, *PRE, *SRE

ADDITIONAL INFORMATION

STATUS BYTE REGISTER (STB)				
Bit	**Bit Value**	**Bit Name**	**Description**	**Note**
7	128	DIO7	0 reserved for future use	
6	64	MSS/RQS MSS = 1 RQS = 1	at least 1 bit in STB masked by SRE is 1 service is requested	1. 2.
5	32	ESB	1 an ESR enabled event has occurred	3.
4	16	MAV	1 output queue is not empty	4.
3	8	DIO3	0 reserved	
2	4	VAB	1 a command data value has been adapted	5.
1	2	DIO1	0 reserved	
0	1	INB	1 an enabled internal state change has occurred	6.

NOTE: For the above table...

1. ***The Master Summary Status (MSS) indicates that the oscilloscope requests service, while the Service Request status — when set — specifies that the oscilloscope issued a service request. Bit position 6 depends on the polling method:***

   Bit 6 = MSS if an *STB? query is received,

   = RQS if serial polling is conducted.

2. ***Example: If SRE = 10 and STB = 10, then MSS = 1. If SRE = 010 and STB = 100 then MSS = 0.***
3. ***The Event Status Bit (ESB) indicates whether or not one or more of the enabled IEEE 488.2 events have occurred since the last reading or clearing of the Standard Event Status Register (ESR). ESB is set if an enabled event becomes true (1).***
4. ***The Message AVailable bit (MAV) indicates whether or not the Output queue is empty. The MAV summary bit is set true (1) whenever a data byte resides in the Output queue.***
5. ***The Value Adapted Bit (VAB) is set true (1) whenever a data value in a command has been adapted to the nearest legal value. For instance, the VAB bit would be set if the timebase is redefined as 2.5 µs/div since the adapted value is 2 µs/div.***
6. ***The INternal state Bit (INB) is set true (1) whenever certain enabled internal states are entered. For further information, refer to the INR query.***

DISPLAY **STITLE**
Command

DESCRIPTION

This command allows you to title panel of menus. Text generated by this command appears directly above the menus. The character limit is 13 ASCII characters.

This command must be preceded by the enable key command EKEY ON.

COMMAND SYNTAX

```
STITLE '<text>'
```

NOTE: To clear a title, issue the command again with nothing between the quotation marks.

RELATED COMMANDS

MSG

ACQUISITION

STOP
Command

DESCRIPTION

The STOP command immediately stops the acquisition of a signal. If the trigger mode is AUTO or NORM, STOP will place the oscilloscope in STOPPED trigger mode to prevent further acquisition.

COMMAND SYNTAX

```
STOP
```

EXAMPLE

The following stops the acquisition process:

```
CMD$ =“STOP”: CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

ARM_ACQUISITION, TRIG_MODE, WAIT

WAVEFORM TRANSFER

STORE, STO
Command

DESCRIPTION

The STORE command stores the contents of the specified trace into one of the internal memories M1 to M4 or to the current directory on mass storage.

COMMAND SYNTAX

STOre [<trace>,<dest>]

<trace> : = { TA, TB, TC, TD, C1, C2, C3, C4, ALL_DISPLAYED}

<dest> : = { M1, M2, M3, M4, CARD, FLPY, HDD}

TIP: If you send the STORE command without an argument, all traces currently enabled in the Store Setup will be stored. Modify this setup using STORE_SETUP.

AVAILABILITY

<trace> : = { C3, C4} only available on four-channel oscilloscopes.

<dest> : CARD only available with Memory Card option installed.

<dest> : HDD only available with removable Hard Disk option installed.

EXAMPLE (GPIB)

The following stores the contents of Trace A (TA) into Memory 1 (M1):

```
CMD$="STO TA,M1": CALL IBWRT(SCOPE%,CMD$)
```

The following stores all currently displayed waveforms onto the memory card:

```
CMD$="STO ALL_DISPLAYED, CARD":
CALL IBWRT(SCOPE%,CMD$)
```

The following executes the storage operation currently defined in the Storage Setup (see command STORE_SETUP):

```
CMD$="STO": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

STORE_SETUP, RECALL

SAVE/RECALL SETUP

STORE_PANEL, STPN
Command

DESCRIPTION

The STORE_PANEL command stores the complete front panel setup of your *WavePro* oscilloscope, at the time the command is issued, into a file on the current directory on mass storage.

NOTE: The communication parameters (the parameters modified by commands COMM_FORMAT, COMM_HEADER, COMM_HELP, COMM_ORDER and WAVEFORM_SETUP) and the enable registers associated with the status reporting system (commands *SRE, *PRE, *ESE, INE) are not saved by this command.

If no filename (or an empty string) is supplied, the oscilloscope generates a filename according to its internal rules.

COMMAND SYNTAX

STore_PaNel DISK, <device>, FILE, '<filename>'

<device> : = { CARD, FLPY, HDD}

<filename> : = A string of up to eight characters with the extension .PNL.

AVAILABILITY

<device> : CARD only available with Memory Card option installed.

<device> : HDD only available with removable Hard Disk option installed.

EXAMPLE (GPIB)

The following saves the current *WavePro* oscilloscope setup to the memory card in a file called "DIODE.PNL":

```
CMD$="STPN DISK, CARD, FILE,'DIODE.PNL'":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PNSU, *SAV, RECALL_PANEL, *RCL

WAVEFORM TRANSFER

STORE_SETUP, STST

Command/ Query

DESCRIPTION

The STORE_SETUP command controls the way in which traces will be stored. A single trace or all displayed traces can be enabled for storage. This applies to both auto-storing and to the STORE command. Traces can be auto-stored to mass storage after each acquisition until the mass storage device becomes full (FILL), or continuously (WRAP), replacing the oldest traces by new ones.

The STORE_SETUP? query returns the current mode of operation of AutoStore, the current trace selection, and the current destination.

NOTE: You can recall into the oscilloscope only waveforms stored in BINARY format.

COMMAND SYNTAX

STore_SeTup [<trace>, <dest>] [, AUTO, <mode>] [, FORMAT, <type>]

<trace> : = { TA, TB, TC, TD, C1, C2, C3, C4, ALL_DISPLAYED}
<dest> : = { M1, M2, M3, M4, CARD, FLPY, HDD}
<mode> : = { OFF, WRAP, FILL}
<type> : { BINARY, SPREADSHEET, MATHCAD, MATLAB }

QUERY SYNTAX

STore_SeTup?

RESPONSE FORMAT

STore_SeTup <trace>,<dest>,AUTO,<mode>

AVAILABILITY

<trace> : = { C3, C4} only available on four-channel oscilloscopes.

<dest> : CARD only available with Memory Card option installed.

<dest> : HDD only available with removable Hard Disk option installed.

EXAMPLE (GPIB)

The following Channel 1 for storage. It enables an "autostore" to the card until no more space is left on the memory card (AUTO, FILL).

```
CMD$="STST C1, CARD, AUTO,FILL":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

STORE, INR

WAVEFORM TRANSFER **STORE_TEMPLATE, STTM**
Command

DESCRIPTION

The STORE_TEMPLATE command stores your *WavePro* oscilloscope's waveform template on a mass-storage device. A filename is automatically generated in the form of LECROYvv.TPL where vv is the two-digit revision number.

For example, for revision 2.1, the file name generated will be LECROY21.TPL.

See Chapter 4 for more on the waveform template, and Appendix II for a copy of the template itself.

COMMAND SYNTAX

STore_TeMplate DISK,<device>

<device> := {CARD, FLPY, HDD, VDISK}

AVAILABILITY

<device> : CARD only available with the Memory Card option installed.

<device> : HDD only available with removable Hard Disk option installed.

<device> : VDISK = non-volatile RAM

EXAMPLE (GPIB)

The following stores the current waveform template on the memory card for future reference:

```
CMD$="STTM DISK, CARD": CALL IBWRT
(SCOPE%,CMD$)
```

RELATED COMMANDS

TEMPLATE

MISCELLANEOUS

TDISP

Command/ Query

DESCRIPTION

The TDISP command controls the presence and format of the time and date display on the oscilloscope screen. The keyword CURRENT selects the current time and date for display. The keyword NONE removes both time and date. The keyword TRIGGER selects the trigger time of the uppermost waveform currently displayed.

COMMAND SYNTAX

TDISP <state>

<state> := {CURRENT,NONE,TRIGGER}

QUERY SYNTAX

TDISP?

RESPONSE FORMAT

TDISP <state>

EXAMPLE (GPIB)

The following turns off the time and date display on the oscilloscope screen:

```
CMD$="TDISP NONE": CALL IBWRT(SCOPE%,CMD$)
```

WAVEFORM TRANSFER

TEMPLATE?, TMPL?

Query

DESCRIPTION

The TEMPLATE? query produces a copy of the template which formally describes the various logical entities making up a complete waveform. In particular, the template describes in full detail the variables contained in the descriptor part of a waveform.

See Chapter 4 for more on the waveform template, and Appendix II for a copy of the template itself.

QUERY SYNTAX

TeMPLate?

RESPONSE FORMAT

TeMPLate "<template>"

<template> := A variable length string detailing the structure of a waveform.

RELATED COMMANDS

INSPECT

ACQUISITION

TIME_DIV, TDIV

Command/ Query

DESCRIPTION

The TIME_DIV command modifies the timebase setting. The new timebase setting can be specified with suffixes: NS for nanoseconds, US for microseconds, MS for milliseconds, S for seconds, or KS for kiloseconds. An out-of-range value causes the VAB bit (bit 2) in the STB register (see table on page 224) to be set.

The TIME_DIV? query returns the current timebase setting.

COMMAND SYNTAX

Time_DIV <value>

<value> : = See the Operator's Manual for specifications.

The suffix S (seconds) is optional.

QUERY SYNTAX

Time_DIV?

RESPONSE FORMAT

Time_DIV <value>

EXAMPLE (GPIB)

The following sets the time base to 500 µs/ div:

```
CMD$="TDIV 500US": CALL IBWRT(SCOPE%,CMD$)
```

The following sets the time base to 2 msec/ div:

```
CMD$="TDIV 0.002": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_DELAY, TRIG_MODE

DISPLAY

TRACE, TRA

Command/ Query

DESCRIPTION

The TRACE command enables or disables the display of a trace. An environment error (see table on page 130) is set if an attempt is made to display more than four waveforms.

The TRACE? query indicates whether the specified trace is displayed or not.

COMMAND SYNTAX

<trace> : TRAce <mode>

<trace> : = {C1, C2, C3, C4, TA, TB, TC, TD}

<mode> : = {ON, OFF}

QUERY SYNTAX

<trace> : TRAce?

RESPONSE FORMAT

<trace> : TRAce <mode>

AVAILABILITY

<trace> : = {C3, C4} only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following displays Trace A (TA):

```
CMD$="TA:TRA ON": CALL IBWRT(SCOPE%,CMD$)
```

DISPLAY

TRACE_LABEL, TRLB

Command/ Query

DESCRIPTION

This command allows you to enter a label for a trace. The label is displayed in the channel descriptor field on the left side of the display.

COMMAND SYNTAX

<channel> : TRace_LaBel "<text>"

<channel> : = { C1, C2, C3, C4 }

QUERY SYNTAX

<channel> : TRLB?

RESPONSE FORMAT

<channel> : TRACE_LABEL "<text>"

EXAMPLE (GPIB)

The following instruction sets the trace label for channel 2 to "headsig":

```
CMD$="C2:TRACE_LABEL 'HEADSIG'"
CALL IBWRT (SCOPE%,CMD$)
```

RELATED COMMANDS

TRACE

DISPLAY

TRACE_OPACITY, TOPA

Command/ Query

DESCRIPTION

The TRACE_OPACITY command controls the opacity and the transparency of the trace color. The trace can be made either opaque (traces will overwrite and obscure each other) or transparent (overlapping traces can be distinguished from one another).

The response to the TRACE_OPACITY? query indicates whether the traces are opaque or transparent.

COMMAND SYNTAX

Trace_OPAcity <type>

<type> := {OPAQUE, TRANSPARENT}

QUERY SYNTAX

Trace_OPAcity?

RESPONSE FORMAT

Trace_OPAcity <type>

EXAMPLE (GPIB)

The following allows traces to be distinguished even when they overlap:

```
CMD$="TOPA TRANSPARENT": CALL
BWRT(SCOPE%,CMD$)
```

DISPLAY

TRACE_ORDER, TORD

Command/ Query

DESCRIPTION

This command allows you to specify the display order of traces.

COMMAND SYNTAX

Trace_ORDer <slot><trace>

<slot> : = <1, 2, 3, 4, 5, 6, 7, 8>

where 1 is at the top

<trace> : = <C1, C2, C3, C4, TA, TB, TC, TD, OFF>

QUERY SYNTAX

TORD?

RESPONSE FORMAT

TORD 1,C1,2,OFF,3,OFF,4,OFF,
5,OFF,6,OFF,7,OFF,8,OFF

EXAMPLE (GPIB)

The following command displays Trace A above channel 1:

CMD$="TORD 1,TA,2,C1"

CALL IBWRT(SCOPE%,CMD$)

RELATED COMMANDS

TRACE

WAVEFORM STORAGE

TRANSFER_FILE, TRFL

Command/ Query

DESCRIPTION

This command allows you to transfer files to and from storage media, or between scope and computer. The command format is used to transfer files from the computer to storage media. The query format is used to transfer files from storage media to computer.

COMMAND SYNTAX

TRanser_FiLe,<device>,FILE,'name.ext',#9nnnnnnnnn<data><crc>

<device> := {CARD,FLPY,VDISK,HDD}

Note: HDD is applicable only if the HD01 option is present

<n . . . n> := file size in bytes

<data> := file data (arbitrary data block)

<crc> := 32-bit cyclic redundancy check of <data>

Note: Files are read from or written into the current directory only.

QUERY SYNTAX

TRFL? DISK,<device>,FILE,'name.ext'

RESPONSE FORMAT

TRFL #9nnnnnnnnn<file content><crc>

EXAMPLE (GPIB)

The following instruction reads the file FAVORITE.DSO from the floppy disk:

```
CMD$=TRFL,DISK,FLPY,FILE,'FAVORITE.DSO'
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

DIRECTORY

ACQUISITION

***TRG**
Command

DESCRIPTION

The `*TRG` command executes an ARM command. `*TRG` is the equivalent of the 488.1 GET (Group Execute Trigger) message.

COMMAND SYNTAX

```
*TRG
```

EXAMPLE (GPIB)

The following enables signal acquisition:

```
CMD$="*TRG": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

`ARM_ACQUISITION, STOP, WAIT`

ACQUISITION

TRIG_COUPLING, TRCP

Command/ Query

DESCRIPTION

The TRIG_COUPLING command sets the coupling mode of the specified trigger source.

The TRIG_COUPLING? query returns the trigger coupling of the selected source.

NOTE: The oscilloscope automatically determines trigger slope when in HFDIV coupling. The TRIG_SLOPE command is not used in HFDIV coupling mode.

HFDIV is indicated as HF in the trigger setup menus.

COMMAND SYNTAX

<trig_source> : TRig_CouPling <trig_coupling>

<trig_source> : = { C1, C2, C3, C4, EX, EX10}

<trig_coupling> : = { AC, DC, HFREJ, LFREJ}

QUERY SYNTAX

<trig_source> : TRig_CouPling?

RESPONSE FORMAT

<trig_source> : TRig_CouPling <trig_coupling>

AVAILABILITY

<trig_source> : = { C3, C4} only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following sets the coupling mode of the trigger source Channel 2 to AC:

```
CMD$="C2:TRCP AC": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_DELAY, TRDL

Command/ Query

DESCRIPTION

The TRIG_DELAY command sets the time at which the trigger is to occur in respect of the first acquired data point (displayed at the left-hand edge of the screen).

Positive trigger delays are to be expressed as a percentage of the full horizontal screen. This mode is called pre-trigger acquisition, as data are acquired before the trigger occurs. Negative trigger delays must be given in seconds. This mode is called post-trigger acquisition, as the data are acquired after the trigger has occurred.

If a value outside the range –10 000 div × time/ div and 100 % is specified, the trigger time will be set to the nearest limit and the VAB bit (bit 2) will be set in the STB register.

The response to the TRIG_DELAY? query indicates the trigger time with respect to the first acquired data point. Positive times are expressed as a percentage of the full horizontal screen and negative times in seconds.

COMMAND SYNTAX

TRig_DeLay <value>

<value> : = 0.00 PCT to 100.00 PCT (pretrigger)

–20 PS to –10 MAS (post-trigger)

NOTE: The suffix is optional. For positive numbers, the suffix PCT is assumed. For negative numbers, the suffix S is assumed. MAS is the suffix for Ms (megaseconds), useful only for extremely large delays at very slow timebases.

QUERY SYNTAX TRig_DeLay?

RESPONSE FORMAT TRig_DeLay <value>

EXAMPLE (GPIB) The following sets the trigger delay to –20 s (post-trigger):

```
CMD$="TRDL –20S": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS TIME_DIV, TRIG_COUPLING, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_LEVEL, TRLV

Command/ Query

DESCRIPTION

The TRIG_LEVEL command adjusts the trigger level of the specified trigger source. An out-of-range value will be adjusted to the closest legal value and will cause the VAB bit (bit 2) in the STB register to be set.

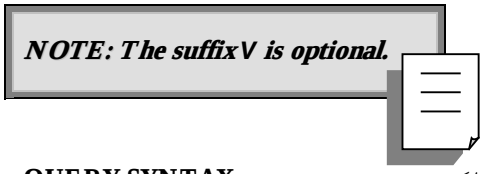
The TRIG_LEVEL? query returns the current trigger level.

COMMAND SYNTAX

<trig_source> : TRig_LeVel <trig_level>

<trig_source> : = { C1, C2, C3, C4, EX, EX10}

NOTE: The suffix V is optional.



QUERY SYNTAX

<trig_source> : TRig_LeVel?

RESPONSE FORMAT

<trig_source> : TRig_LeVel <trig_level>

AVAILABILITY

<trig_source> : = { C3, C4} only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following adjusts the trigger level of Channel 2 to –3.4 V:

```
CMD$="C2:TRLV -3.4V": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_MODE, TRMD

Command/ Query

DESCRIPTION

The TRIG_MODE command specifies the trigger mode.

The TRIG_MODE? query returns the current trigger mode.

COMMAND SYNTAX

TRig_MoDe <mode>

<mode> : = { AUTO, NORM, SINGLE, STOP}

QUERY SYNTAX

TRig_MoDe?

RESPONSE FORMAT

TRig_MoDe <mode>

EXAMPLE (GPIB)

The following selects the normal mode:

```
CMD$="TRMD NORM": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

ARM_ACQUISITION, STOP, TRIG_SELECT, SEQUENCE, TRIG_COUPLING, TRIG_LEVEL, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_PATTERN, TRPA

Command/ Query

DESCRIPTION

The TRIG_PATTERN command defines a trigger pattern. The command specifies the logic composition of the pattern sources (Channel 1, Channel 2, Channel 3 and Channel 4), as well as the conditions under which a trigger can occur. Note that this command can be used even if the complex trigger mode has not been activated.

L	**LOW**	**H**	**HIGH**
PR	**PATTERN PRESENT**	**AB**	**PATTERN ABSENT**
EN	**PATTERN ENTERED**	**EX**	**PATTERN EXITED**

The TRIG_PATTERN? query returns the current trigger pattern.

Note: PR and EN, and AB and EX are equivalent.

COMMAND SYNTAX

TRig_PAttern [<source>,<state>,...<source>,<state>],STATE, <trigger_condition>

<source> : = { C1, C2, C3, C4, EX}

<state> : = { L, H}

<trigger_condition> : = { AB, PR, EX, EN}

Note: If a source state is not specified in the command, the source will be set to the X (= don't care) state. The response sends back only the source states that are either H (= high) or L (= low), ignoring the X states.

QUERY SYNTAX

TRig_PAttern? [<source>,<state>,...<source>,<state>],STATE, <trigger_condition>

<source> : = { C1, C2, C3, C4, EX}

<state> : = { L, H}

RESPONSE FORMAT

TRig_PAttern [<source>,<state>,...<source>,<state>],STATE,<trigger_condition>

EXAMPLE (GPIB)

The following configures the logic state of the pattern as HLXH (CH 1 = H, CH 2 = L, CH 3 = X, CH 4 = H) and defines the trigger condition as pattern absent (AB).

```
CMD$="TRPA C1,H,C2,L,C4,H,STATE,AB":
CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_LEVEL_2, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_SELECT, TRSE

Command/ Query

DESCRIPTION

The TRIG_SELECT command selects the condition that will trigger the acquisition of waveforms. Depending on the trigger type, additional parameters must be specified. These additional parameters are grouped in pairs. The first in the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs can be given in any order and restricted to those variables to be changed.

The TRIG_SELECT? query returns the current trigger condition.

TRIGGER NOTATION			
DROP	Dropout	QL	Qualifier
EDGE	Edge	SNG	Single source
EV	Event	SQ	State-Qualified
GLIT	Glitch	SR	Source
HT	Hold type	STD	1Standard (Edge Trigger)
HV	Hold value	TEQ	Edge-Qualified
HV2	Second hold value	TI	Time
IL	Interval larger	TL	Time within
INTV	Interval	**TV**	**TV**
IS	Interval smaller	CHAR	Characteristics
I2	Interval-width window	FLD	Field
OFF	No hold-off on wait	FLDC	Field count
PL	Pulse larger	ILAC	Interlace
PS	Pulse smaller	LINE	Line
P2	Pulse-width window	LPIC	Lines per picture

COMMAND SYNTAX

For all but TV Trigger TRig_SElect
<trig_type>,SR,<source>,QL,<source>,

HT,<hold_type>,HV,<hold_value>,HV2,<hold value>

1 HT and HV do not apply to the Standard Trigger.

<trig_type> : = { DROP, EDGE, GLIT, INTV, STD, SNG, SQ, TEQ}

<source> : = { C1, C2, C3, C4, LINE, EX, EX10, PA}

<hold_type> : = { TI, TL, EV, PS, PL, IS, IL, P2, I2, OFF}

<hold_value> : = See Operator's Manual for valid values

NOTE: The suffix S (seconds) is optional.

QUERY SYNTAX

TRig_SElect?

RESPONSE FORMAT

TRig_SElect
<trig_type>,SR,<source>,HT,<hold_type>,HV,

<hold_value>,<hold_value>

HV2 only returned if <hold_type> is P2 or I2

AVAILABILITY

<source> : { C3, C4} only available on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following selects the single-source trigger with Channel 1 as trigger source. Hold type and hold value are chosen as "pulse smaller" than 20 ns:

```
CMD$="TRSE SNG,SR,C1,HT,PS,HV,20 NS":
CALL IBWRT(SCOPE%,CMD$)
```

TV COMMAND SYNTAX

TRig_SElect
<trig_type>,SR,<source>,FLDC,<field_count>,FLD,<field>,
CHAR,<characteristics>,LPIC,<lpic>,ILAC,<ilace>,LINE,<line>

<trig_type> : = { TVP,TVN} TVP=TV Pos, TVN=TV Neg

<source> : = { C1, C2, C3, C4, NE, EX, EX10}

<field_count> : = { 1, 2}

<field> : = 1 to field_count

<characteristics> : = { NTSC, PALSEC, CUST50, CUST60}

<lpic> : = 1 to 1500

<ilace> : = { 1, 2}

<line> : = 1 to 1500 or 0 for any line

The FLD value is interpreted with the current FLDC value. The LINE value is interpreted with the current FLD and CHAR values.

QUERY SYNTAX

TRig_SElect?

RESPONSE FORMAT

TRig_SElect
TVP,SR,<source>,FLDC,<field_count>,FLD,<field>,
CHAR,<characteristic>,LINE,<line>

AVAILABILITY

<source> : = { C3, C4} only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following sets up the trigger system to trigger on the 3rd field, line 17, of the eight-field PAL / SECAM TV signal applied to the external input.

```
CMD$="TRSE TVN,SR,EX,FLDC,8,FLD,3,CHAR,PALSEC,
LINE,17": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL,
TRIG_MODE, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_SLOPE, TRSL

Command/ Query

DESCRIPTION

The TRIG_SLOPE command sets the trigger slope of the specified trigger source. An environment error (see table on page 130) will be generated when an incompatible TRSL order is received while the trigger coupling is set to HFDIV (see TRIG_COUPLING).

The TRIG_SLOPE? query returns the trigger slope of the selected source.

COMMAND SYNTAX

<trig_source> : TRig_SLope <trig_slope>

<trig_source> : = {C1, C2, C3, C4, LINE, EX, EX10}

<trig_slope> : = {NEG, POS, WINDOW}

QUERY SYNTAX

<trig_source> : TRig_SLope?

RESPONSE FORMAT

<trig_source> : TRig_SLope <trig_slope>

AVAILABILITY

<trig_source> : = {C3, C4} only available on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following sets the trigger slope of Channel 2 to negative:

```
CMD$="C2:TRSL NEG": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION

TRIG_WINDOW, TRWI

Command/ Query

DESCRIPTION

The TRIG_WINDOW command sets the window amplitude in volts on the current Edge trigger source. The window is centered around the Edge trigger level.

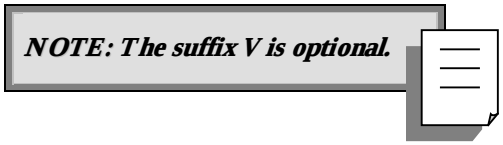
The TRIG_WINDOW? query returns the current window amplitude.

COMMAND SYNTAX

TRig_WIndow <value>

<value> : = 0 to 25 V (maximum range)

NOTE: The suffix V is optional.



QUERY SYNTAX

TRig_WIndow?

RESPONSE FORMAT

TRig_WIndow <trig_level>

EXAMPLE

The following adjusts the window size to +0.5 V:

```
CMD$="TRWI 0.5V": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE

MISCELLANEOUS

*TST?
Query

DESCRIPTION

The *TST? query performs an internal self-test, the response indicating whether the self-test has detected any errors. The self-test includes testing the hardware of all channels, the timebase and the trigger circuits.

Hardware failures are identified by a unique binary code in the returned <status> number. A "0" response indicates that no failures occurred.

QUERY SYNTAX

```
*TST?
```

RESPONSE FORMAT

```
*TST <status>
```

<status> := 0 self-test successful

EXAMPLE (GPIB)

The following causes a self-test to be performed:

```
CMD$="*TST?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RD$): PRINT RD$
```

Response message (if no failure):

```
*TST 0
```

RELATED COMMANDS

*CAL

STATUS

URR?
Query

DESCRIPTION

The URR? query reads and clears the contents of the User Request status Register (URR). The URR register specifies which button in the menu field was pressed.

In remote mode, the URR register indicates the last button was pressed, provided it was activated with a KEY command (see page 161). In local mode, the URR register indicates whether the CALL HOST button has been pressed. If no menu button has been pressed since the last URR? query, the value 0 is returned.

USER REQUEST STATUS REGISTER STRUCTURE (URR)

Value	Description
0	No button has been pressed
1	The top menu button has been pressed
2	The second-from-top menu button has been pressed
3	The third-from-top menu button has been pressed
4	The fourth-from-top menu button has been pressed
5	The fifth-from-top menu button has been pressed
100	When the "Call Host" command is "On" (the bottom button for the root, or primary, menu has been pressed)

QUERY SYNTAX

URR?

RESPONSE FORMAT

URR <value>
<value> := 0 to 5, 100

EXAMPLE (GPIB)

The following reads the contents of the URR register:

```
CMD$="URR?": CALL IBWRT(SCOPE%,CMD$):
CALL IBRD(SCOPE%,RSP$): PRINT RSP$
```

Response message:

```
URR 0
```

RELATED COMMANDS

CALL_HOST, KEY, ALL_STATUS, *CLS

DISPLAY

VERT_MAGNIFY, VMAG

Command/ Query

DESCRIPTION

The VERT_MAGNIFY command vertically expands the specified trace. The command is executed even if the trace is not displayed.

The maximum magnification allowed depends on the number of significant bits associated with the data of the trace.

The VERT_MAGNIFY? query returns the magnification factor of the specified trace.

COMMAND SYNTAX

<trace> : Vert_MAGnify <factor>

<trace> : = { TA, TB, TC, TD}

<factor> : = 4.0E-3 to 50 (maximum)

QUERY SYNTAX

<trace> : Vert_MAGnify?

RESPONSE FORMAT

<trace> : Vert_MAGnify <factor>

EXAMPLE

The following enlarges the vertical amplitude of Trace A by a factor of 3.45 with respect to its original amplitude:

```
CMD$="TA:VMAG 3.45": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

VERT_POSITION

DISPLAY

VERT_POSITION, VPOS

Command/ Query

DESCRIPTION

The VERT_POSITION command adjusts the vertical position of the specified trace on the screen. It does not affect the original offset value obtained at acquisition time.

The VERT_POSITION? query returns the current vertical position of the specified trace.

COMMAND SYNTAX

<trace> : Vert_POSITION <display_offset>

<trace> : = { TA, TB, TC, TD}

<display_offset> : = –5900 to +5900 DIV

NOTE: The suffix DIV is optional. The limits depend on the current magnification factor, the number of grids on the display, and the initial position of the trace.

QUERY SYNTAX

<trace> : Vert_POSition?

RESPONSE FORMAT

<trace> : Vert_POSITION <display_offset>

EXAMPLE

The following shifts Trace A (TA) upwards by +3 divisions relative to the position at the time of acquisition:

```
CMD$="TA:VPOS 3DIV": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

VERT_MAGNIFY

ACQUISITION

VOLT_DIV, VDIV

Command/ Query

DESCRIPTION

The VOLT_DIV command sets the vertical sensitivity in Volts/ div. The VAB bit (bit 2) in the STB register (see table on page 224) is set if an out-of-range value is entered.

The probe attenuation factor is not taken into account for adjusting vertical sensitivity.

The VOLT_DIV? query returns the vertical sensitivity of the specified channel.

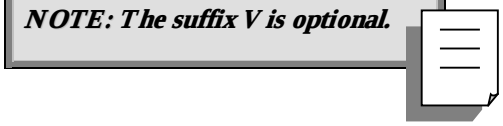
COMMAND SYNTAX

<channel>: Volt_DIV <v_gain>

<channel> : = { C1, C2, C3, C4}

<v_gain> : = See Operator's Manual for specifications.

NOTE: The suffix V is optional.



QUERY SYNTAX

<channel> : Volt_DIV?

RESPONSE FORMAT

<channel> : Volt_DIV <v_gain>

AVAILABILITY

<channel> : = { C3, C4} only available on four-channel oscilloscopes.

EXAMPLE

The following sets the vertical sensitivity of channel 1 to 50 mV/ div:

```
CMD$="C1:VDIV 50MV": CALL IBWRT(SCOPE%,CMD$)
```

STATUS

***WAI**
Command

DESCRIPTION

The *WAI (WAIt to continue) command, required by the IEEE 488.2 standard, has no effect on the oscilloscope, as *WavePro* DSO only starts processing a command when the previous command has been entirely executed.

COMMAND SYNTAX

*WAI

RELATED COMMANDS

*OPC

ACQUISITION

WAIT
Command

DESCRIPTION

The WAIT command prevents your *WavePro* oscilloscope from analyzing new commands until the oscilloscope has completed the current acquisition. The optional argument specifies the timeout (in seconds) after which the scope will stop waiting for new acquisitions. If <t> is not given, or if <t> = 0.0, the scope will wait indefinitely.

COMMAND SYNTAX

```
WAIT [<t>]
```

<t> := timeout in seconds (0.0 to 1000.0, default is indefinite)

EXAMPLE (GPIB)

```
send: "TRMD SINGLE"
loop {send:"ARM;WAIT;C1:PAVA?MAX"
read response
process response
}
```

This example finds the maximum amplitudes of several signals acquired one after another. ARM starts a new data acquisition. The WAIT command ensures that the maximum is evaluated for the newly acquired waveform.

C1:PAVA?MAX instructs the oscilloscope to evaluate the maximum data value in the Channel 1 waveform.

RELATED COMMANDS

*TRG SLEEP

WAVEFORM TRANSFER

WAVEFORM, WF
Command/ Query

DESCRIPTION

A WAVEFORM command transfers a waveform from the controller to the oscilloscope, whereas a WAVEFORM? query transfers a waveform from the oscilloscope to the controller.

WAVEFORM stores an external waveform back into the oscilloscope's internal memory. A waveform consists of several distinct entities:

the descriptor (DESC)

the user text (TEXT)

the time (TIME) descriptor

the data (DAT1) block, and, optionally

a second block of data (DAT2).

See Chapter 4 for further information on waveform structure.

NOTE: You can restore to the oscilloscope only complete waveforms queried with WAVEFORM? ALL.

The WAVEFORM? query instructs the oscilloscope to transmit a waveform to the controller. The entities can be queried independently. If the "ALL" parameter is specified, all four or five entities are transmitted in one block in the order enumerated above.

NOTE: The format of the waveform data depends on the current settings specified by the last WAVEFORM_SETUP, COMM_ORDER and COMM_FORMAT commands.

COMMAND SYNTAX

<memory> : WaveForm ALL <waveform_data_block>

<memory> : = { M1, M2, M3, M4}

<waveform_data_block> : = Arbitrary data block (see Chapter 5).

QUERY SYNTAX

<trace> :WaveForm? <block>

<trace> : = { TA, TB, TC,TD, M1, M2, M3, M4, C1, C2, C3, C4}

<block> : = { DESC, TEXT, TIME, DAT1, DAT2, ALL}

If you do not give a parameter, ALL will be assumed.

RESPONSE FORMAT

<trace> :WaveForm <block>,<waveform_data_block>

TIP: It may be convenient to disable the response header if the waveform is to be restored. See the COMM_HEADER command for further details.

AVAILABILITY

<trace> : = { C3, C4} only available on four-channel oscilloscopes.

EXAMPLES (GPIB)

The following reads the block DAT1 from Memory 1 and saves it in the file "MEM1.DAT". The path header "M1:" is saved together with the data.

```
FILE$ = "MEM1.DAT"
CMD$ = "M1:WF? DAT1"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRDF(SCOPE%,FILE$)
```

In the following example, the entire contents of Channel 1 are saved in the file "CHAN1.DAT". The path header "C1:" is skipped to ensure that the data can later be recalled into the oscilloscope.

```
FILE$="CHAN1.DAT":RD$=SPACE$(3)
CMD$="CHDR SHORT; C1:WF?"
CALL IBWRT(SCOPE%,CMD$)
CALL IBRD(SCOPE%,RD$) Skip first 3 characters "C1:"
CALL IBRDF(SCOPE%,FILE$) Save data in file "CHAN1.DAT"
```

The following illustrates how the waveform data saved in the preceding example can be recalled into Memory 1:

```
FILE$ = "CHAN1.DAT"
CMD$ ="M1:"
CALL IBEOT(SCOPE%,0)disable EOI
CALL IBWRT(SCOPE%,CMD$)
CALL IBEOT(SCOPE%,1)re-enable EOI
CALL IBWRTF(SCOPE%,FILE$)
```

The “M1:” command ensures that the active waveform is “M1”. When the data file is sent to the oscilloscope, it first sees the header “WF” (the characters “C1:” having been skipped when reading the file) and assumes the default destination “M1”.

RELATED COMMANDS

INSPECT, COMM_FORMAT, COMM_ORDER, FUNCTION_STATE, TEMPLATE, WAVEFORM_SETUP, WAVEFORM_TEXT

WAVEFORM TRANSFER

WAVEFORM_SETUP, WFSU

Command/ Query

DESCRIPTION

The WAVEFORM_SETUP command specifies the amount of data in a waveform to be transmitted to the controller. The command controls the settings of the parameters listed below.

NOTATION			
FP	first point	NP	number of points
SN	segment number	SP	sparsing

Sparsing (SP):

The sparsing parameter defines the interval between data points. For example:

SP = 0	sends all data points
SP = 1	sends all data points
SP = 4	sends every 4th data point

Number of points (NP):

The number of points parameter indicates how many points should be transmitted. For example:

NP = 0	sends all data points
NP = 1	sends 1 data point
NP = 50	sends a maximum of 50 data points
NP = 1001	sends a maximum of 1001 data points

First point (FP):

The first point parameter specifies the address of the first data point to be sent. For waveforms acquired in sequence mode, this refers to the relative address in the given segment. For example:

FP = 0	corresponds to the first data point
FP = 1	corresponds to the second data point
FP = 5000	corresponds to data point 5001

Segment number (SN): The segment number parameter indicates which segment should be sent if the waveform was acquired in sequence mode. This parameter is ignored for non-segmented waveforms. For example:

`SN = 0` all segments

`SN = 1` first segment

`SN = 23` segment 23

The `WAVEFORM_SETUP?` query returns the transfer parameters currently in use.

COMMAND SYNTAX

`WaveForm_SetUp`
`SP,`<sparsing>`,NP,`<number>`,FP,`<point>`,SN,`

NOTE: After power-on, all values are set to 0 (i.e. entire waveforms will be transmitted without sparsing).

Parameters are grouped in pairs. The first of the pair names the variable to be modified, while the second gives the new value to be assigned. Pairs can be given in any order and restricted to those variables to be changed.

QUERY SYNTAX

`WaveForm_SetUp?`

RESPONSE FORMAT

`WaveForm_SetUp`
`SP,`<sparsing>`,NP,`<number>`,FP,`<point>`,SN,`

EXAMPLE (GPIB) The following instructs every 3rd data point (SP=3) starting at address 200 to be transferred:

```
CMD$="WFSU SP,3,FP,200": CALL
IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS `INSPECT, WAVEFORM, TEMPLATE`

WAVEFORM TRANSFER

WAVEFORM_TEXT, WFTX

Command/ Query

DESCRIPTION

The WAVEFORM_TEXT command is used to document the conditions under which a waveform has been acquired. The text buffer is limited to 160 characters.

The WAVEFORM_TEXT? query returns the text section of the specified trace.

COMMAND SYNTAX

<trace> : WaveForm_TeXt '<text>'

<trace> : = { TA, TB, TC, TD, M1, M2, M3, M4, C1, C2, C3, C4}

<text> : = An ASCII message (max. 160 characters long)

QUERY SYNTAX

<trace> : WaveForm_TeXt?

RESPONSE FORMAT

<trace> : WaveForm_TeXt "<text>"

AVAILABILITY

<trace> : = { C3, C4} only on four-channel *WavePro* oscilloscopes.

EXAMPLE (GPIB)

The following documents Trace A (TA):

```
MSG$= "'Averaged pressure signal. Experiment
carried out Jan.15, 98'"
CMD$= "TA:WFTX"+ MSG$: CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMAND

INSPECT, WAVEFORM, TEMPLATE

DISPLAY

XY_ASSIGN?, XYAS?

Query

DESCRIPTION

The XY_ASSIGN? query returns the traces currently assigned to the XY display. If there is no trace assigned to the X or Y axis the value UNDEF will be returned instead of the trace name.

QUERY SYNTAX

XY_ASsign?

RESPONSE FORMAT

XY_ASsign <X_source>,<Y_source>

<X_source> := { UNDEF, TA, TB, TC, TD, C1, C2, C3, C4}
<Y_source> := { UNDEF, TA, TB, TC, TD, C1, C2, C3, C4}

AVAILABILITY

<X_source> := { C3, C4} only available on four-channel oscilloscopes.

<Y_source> := { C3, C4} only available on four-channel oscilloscopes.

EXAMPLE (GPIB)

The following query finds the traces assigned to the X axis and the Y axis respectively:

```
CMDS$="XYAS?": CALL IBWRT(SCOPE%,CMDS$)
```

Example of response message:

```
XYAS C1,C2
```

RELATED COMMANDS

TRACE

CURSOR

XY_CURSOR_ORIGIN, XYCO

Command/ Query

DESCRIPTION

The XY_CURSOR_ORIGIN command sets the position of the origin for absolute cursor measurements on the XY display.

Absolute cursor values can be measured either with respect to the point (0,0) volts (OFF) or with respect to the center of the XY grid (ON).

The XY_CURSOR_ORIGIN? query returns the current assignment of the origin for absolute cursor measurements.

COMMAND SYNTAX

XY_Cursor_Origin <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

XY_Cursor_Origin?

RESPONSE FORMAT

XY_Cursor_Origin <mode>

EXAMPLE (GPIB)

The following sets the origin for absolute cursor measurements to the center of the XY grid.

```
CMDS$="XYCO ON": CALL IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

XY_CURSOR_VALUE

CURSOR

XY_CURSOR_SET, XYCS

Command/ Query

DESCRIPTION

The XY_CURSOR_SET command allows you to position any one of the six independent XY voltage cursors at a given screen location. The positions of the cursors can be modified or queried even if the required cursor is not currently displayed or if the XY display mode is OFF.

The XY_CURSOR_SET? query indicates the current position of the cursor or cursors.

The CURSOR_SET command is used to position the time cursors.

NOTATION	
XABS	vertical absolute on X axis
XREF	vertical reference on X axis
XDIF	vertical difference on X axis
YABS	vertical absolute on Y axis
YREF	vertical reference on Y axis
YDIF	vertical difference on Y axis

COMMAND SYNTAX

XY_Cursor_Set
<cursor>,<position>[,<cursor>,<position>,...<cursor>,<position>]

<cursor> : = {XABS, XREF, XDIF, YABS, YREF, YDIF}

<position> : = –4 to 4 DIV

NOTE: The suffix DIV is optional.

Parameters are grouped in pairs. The first of the pair names the cursor to be modified, while the second indicates its new value. Pairs can be given in any order and restricted to those items to be changed.

QUERY SYNTAX

`XY_Cursor_Set?` [<cursor,...<cursor>]

<cursor> : = {XABS, XREF, XDIF, YABS, YREF, YDIF, ALL}

If <cursor> is not specified, ALL will be assumed.

RESPONSE FORMAT

`XY_Cursor_Set`
<cursor>,<position>[,<cursor>,<position>...,<cursor>,
<position>]

EXAMPLE (GPIB)

The following positions the XREF and YDIF at +3 DIV and –2 DIV respectively.

```
CMDS$="XYCS XREF,3DIV,YDIF,–2DIV": CALL
IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

`XY_CURSOR_VALUE, CURSOR_MEASURE, CURSOR_SET`

CURSOR

XY_CURSOR_VALUE?, XYCV?

Query

DESCRIPTION

The XY_CURSOR_VALUE? query returns the current values of the X versus Y cursors. The X versus Y trace does not need to be displayed to obtain these parameters, but valid sources must be assigned to the X and Y axes.

NOTATION	
<cursor type> : = [HABS, HREL, VABS, VREL]	
<cursor type>_X	X
<cursor type>_Y	Y
<cursor type>_RATIO	ΔY/ΔX
<cursor type>_PROD	ΔY*ΔX
<cursor type>_ANGLE	arc tan(ΔY/ΔX)
<cursor type>_RADIUS	sqrt(ΔX*ΔX + ΔY*ΔY)

QUERY SYNTAX

XY_Cursor_Value? [<parameter>,...<parameter>]

<parameter> : = { HABS_X, HABS_Y, HABS_RATIO, HABS_PROD, HABS_ANGLE, HABS_RADIUS, HREL_X, HREL_Y, HREL_RATIO, HREL_PROD, HREL_ANGLE, HREL_RADIUS, VABS_X, VABS_Y, VABS_RATIO, VABS_PROD, VABS_ANGLE, VABS_RADIUS, VREL_X, VREL_Y, VREL_RATIO, VREL_PROD, VREL_ANGLE, VREL_RADIUS, ALL}

NOTE: If <parameter> is not specified or equals ALL, all the measured cursor values are returned. If the value of a cursor could not be determined in the current environment, the value UNDEF will be returned. If no trace has been assigned to either the X axis or the Y axis, an environment error will be generated.

RESPONSE FORMAT

`XY_Cursor_Value` <parameter>,<value>[,...<parameter>,<value>]

<value> := A decimal value or UNDEF

EXAMPLE (GPIB)

The following query reads the ratio of the absolute horizontal cursor, the angle of the relative horizontal cursor, and the product of the absolute vertical cursors:

```
CMDS$="XYCV? HABS_RATIO,HREL_ANGLE,VABS_PROD"
CALL IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

`CURSOR_MEASURE, CURSOR_VALUE, XY_CURSOR_ORIGIN`

DISPLAY

XY_DISPLAY, XYDS

Command/ Query

DESCRIPTION

The XY_DISPLAY command turns the XY display mode on or off. (When off, the scope will be in standard display mode.)

The XY_DISPLAY? query returns the current mode of the XY display.

COMMAND SYNTAX

XY_DiSplay <mode>

<mode> := {ON, OFF}

QUERY SYNTAX

XY_DiSplay?

RESPONSE FORMAT

XY_DiSplay <mode>

EXAMPLE (GPIB)

The following turns on the XY display:

```
CMDS$="XYDS ON": CALL IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

GRID

DISPLAY

XY_RENDER, XYRD

Command/ Query

DESCRIPTION

The XY_RENDER command controls the rendering of the XY plot on screen. In Smooth mode, the dots representing XY pairs are connected. In Sharp mode, they are unconnected.

COMMAND SYNTAX

XY_RENDER <state>

<state> : = { SHARP,SMOOTH}

QUERY SYNTAX

XY_RENDER?

RESPONSE FORMAT

XY_RENDER <state>

EXAMPLE (GPIB)

The following sets the rendering to sharp:

```
CMD$="XY_RENDER SHARP": CALL
IBWRT(SCOPE%,CMD$)
```

DISPLAY

XY_SATURATION, XYSA

Command/ Query

DESCRIPTION

The XY_SATURATION command sets the level at which the color spectrum of the persistence display is saturated in XY display mode. The level is specified in terms of percentage (PCT) of the total persistence data map population. A level of 100 PCT corresponds to the color spectrum being spread across the entire depth of the XY persistence data map. At lower values, the spectrum will saturate (brightest value) at the specified percentage value. The PCT is optional.

The response to the XY_SAT? query indicates the saturation level of the XY persistence data map.

COMMAND SYNTAX

XY_SAturation <value>

<value> : = 0 to 100 PCT

NOTE: The suffix PCT is optional.

QUERY SYNTAX

XY_SAturation?

RESPONSE FORMAT

XY_SAturation <value>

EXAMPLE (GPIB)

The following sets the saturation level of the XY persistence data map at 60% (60% of the data points will be displayed with the color spectrum, with the remaining 40% saturated in the brightest color):

```
CMD$="XYSA 60": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PERSIST_SAT

Example 1

USE THE INTERACTIVE GPIB PROGRAM "IBIC"

This example assumes the use of an IBM PC or compatible equipped with a National Instruments GPIB interface card. The GPIB driver is left in default state so that the device name "dev4" corresponds to the GPIB address 4, the oscilloscope address. All `text` is entered by the user.

`IBIC<cr>`

program announces itself

`: ibfind<CR>`

enter board/ device name: `dev4<CR>`

dev4: `ibwrt<CR>`

enter string: `"tdiv?"<CR>`

[0100] (cmpl)

count: 5

dev4: `ibrd<CR>`

enter byte count: `10<CR>`

[0100] (cmpl)

count: 10

54 44 49 56 20 35 30 45	T D I V 5 0 E
2D 39	– 9

dev4: `ibwrt<CR>`

enter string: `"c1:cpl?"<CR>`

[0100] (cmpl)

count: 7

dev4: `ibrd<CR>`

enter byte count: `20<CR>`

[2100] (end cmpl)

count: 11

43 31 3A 43 50 4C 20 44	C 1 : C P L D
35 30 0A	5 0 z

dev4: `q<CR>` to quit the program.

Example 2

USE THE GPIB PROGRAM FOR IBM PC (HIGH-LEVEL FUNCTION CALLS)

The following BASICA program allows full interactive control of the oscilloscope using an IBM PC as GPIB controller. As in Example 1, it is assumed that the controller is equipped with a National Instruments GPIB interface card. All commands can be used following this example simply by entering the text string of the command. For example, `"C1:VDIV 50 MV"`, without the quotation marks. The program automatically displays the information sent back by the oscilloscope in response to queries.

In addition, a few utilities have been provided for convenience. The commands ST and RC enable waveform data to be stored on, or retrieved from, a disk if the correct drive and file names are provided. The command LC returns the oscilloscope to local mode. Responses sent back by the oscilloscope are interpreted as character strings and are thus limited to a maximum of 255 characters.

```
1-99<DECL.BAS>
100CLS
110PRINT "Control of the 9300 via GPIB and IBM PC"
115PRINT ""
120PRINT "Options :EX to exitLC local mode"
125PRINT "ST store dataRC recall data"
130PRINT ""
140LINE INPUT "GPIB-address of oscilloscope (1...16)? :",ADDR$
145DEV$ = "DEV" + ADDR$
150CALL IBFIND(DEV$,SCOPE%)
155IF SCOPE% < 0 THEN GOTO 830
160TMO% = 10  'timeout = 300 msec (rather than default 10 sec)
165CALL IBTMO(SCOPE%,TMO%)
170'
200LOOP% = 1
205WHILE LOOP%
210LINE INPUT "Enter command (EX --> Exit) : ",CMD$
220IF CMD$ = "ex" OR CMD$ = "EX" THEN LOOP% = 0 : GOTO 310
230IF CMD$ = "st" OR CMD$ = "ST" THEN GOSUB 600 : GOTO 300
240IF CMD$ = "rc" OR CMD$ = "RC" THEN GOSUB 700 : GOTO 300
250IF CMD$ = "lc" OR CMD$ = "LC" THEN GOSUB 400 : GOTO 300
```

```
260 IF CMD$ = "" THEN GOTO 300
270 CALL IBWRT(SCOPE%,CMD$)
275 IF IBSTA% < 0 THEN GOTO 840
280 GOSUB 500
300 WEND
310 GOSUB 400
320 END
400 '
405 'SUBROUTINE LOCAL_MODE
410 '
420 CALL IBLOC(SCOPE%)
425 PRINT ""
430 RETURN
500 '
505 'SUBROUTINE GET_DATA
510 'If there are no data to read, simply wait until timeout occurs
515 '
520 CALL IBRD(SCOPE%,RD$)
525 I = IBCNT% 'IBCNT% is the number of characters read
530 FOR J = 1 TO I
535 PRINT MID$(RD$,J,1);
540 NEXT J
545 PRINT ""
550 RETURN
600 '
605 'SUBROUTINE STORE_DATA
610 '
615 RD1$=SPACE$(3)
620 LINE INPUT "Specify trace (TA...TD,M1...M4,C1...C4): ",TRACE$
625 LINE INPUT "Enter filename : ",FILE$
630 CMD$="WFSU NP,0,SP,0,FP,0,SN,0; CHDR SHORT"
```

```
640CALL IBWRT(SCOPE%,CMD$)
645CMD$=TRACE$+":WF?"
650CALL IBWRT(SCOPE%,CMD$)
660CALL IBRD(SCOPE%,RD1$)      'Discard first 3 chars of response
665CALL IBRDF(SCOPE%,FILE$)
670IF IBSTA% < 0 THEN GOTO 840
675PRINT ""
680RETURN
700'
705'SUBROUTINE RECALL_DATA
710'
715LINE INPUT "Specify target memory (M1...M4):",MEM$
720LINE INPUT "Enter filename : ",FILE$
730CMD$=MEM$+":"
735CALL IBWRT(SCOPE%,CMD$)
740CALL IBWRTF(SCOPE%,FILE$)
745IF IBSTA% < 0 THEN GOTO 840
750PRINT ""
755RETURN
800'
810'ERROR HANDLER
820'
830PRINT "IBFIND ERROR"
835END
840PRINT "GPIB ERROR -- IBERR: ";IBERR%;"IBSTA: ";HEX$(IBSTA%)
END
```

NOTE:

It is assumed that the National Instruments GPIB driver GPIB.COM is in its default state. This means that the interface board can be referred to by its symbolic name 'GPIB0' and that devices on the GPIB with addresses 1 to 16 can be called by the symbolic name 'DEV1' to 'DEV16'.

Lines 1–99 are a copy of the file DECL.BAS supplied by National Instruments. The first six lines are required for the initialization of the GPIB handler. DECL.BAS requires access to the file BIB.M during the GPIB initialization. BIB.M is one of the files supplied by National Instruments, and must exist in the directory currently in use.

The first two lines of DECL.BAS each contain a string "XXXXX" which must be replaced by the number of bytes which determine the maximum workspace for BASICA (computed by subtracting the size of BIB.M from the currently available space in BASICA). For example, if the size of BIB.M is 1200 bytes and when BASICA is loaded it reports "60200 bytes free", "XXXXX" would be replaced by the value 59000 or less.

The default timeout of 10 seconds is modified to 300 ms during the execution of this program. However, the default value of the GPIB handler remains unchanged. Whenever a remote command is entered by the user, the program sends it to the instrument with the function call IBWRT. Afterwards, it always executes an IBRD call, regardless of whether or not a response is expected. If a response is received it is immediately displayed. If there is no response, the program waits until time-out and then asks for the next command.

Example 3

USE GPIB PROGRAM FOR IBM PC (LOW-LEVEL FUNCTION CALLS)

This example has the same function as Example 2, but is written with low-level function calls. The program assumes that the controller (board) and oscilloscope (device) are at addresses 0 and 4, respectively, and the decimal addresses are:

	Listener Address	Talker Address
CONTROLLER	32(ASCII <space>)	64 (ASCII @)
DEVICE	32+4=36 (ASCII $)	64+4=68 (ASCII D)

```
1-99<DECL.BAS>
100CLS
110PRINT "Control of the 9300 (address 4) via GPIB and IBM PC"
115PRINT "": PRINT "Options :  EX to exit        LC local mode"
120PRINT "           ST store data     RC recall data": PRINT""
125LOOP=1
130CMD1$ = "?_@$" 'Unlisten, Untalk, Board talker, Device listener
135CMD2$ = "?_ D" 'Unlisten, Untalk, Board listener, Device talker
140BDNAME$= "GPIB0": CALL IBFIND(BDNAME$,BRD0%)
145IF BRD0% < 0 THEN GOTO 420
150CALL IBSIC(BRD0%): IF IBSTA% < 0 THEN GOTO 425
155WHILE LOOP
160LINE INPUT "Enter command (EX --> Exit) : ",CMD$
165V% = 1: CALL IBSRE(BRD0%,V%)
170IF CMD$ = "ex" OR CMD$ = "EX" THEN LOOP = FALSE: GOTO 205
175IF CMD$ = "st" OR CMD$ = "ST" THEN GOSUB 285: GOTO 200
180IF CMD$ = "rc" OR CMD$ = "RC" THEN GOSUB 365: GOTO 200
185IF CMD$ = "lc" OR CMD$ = "LC" THEN  GOSUB 240: GOTO 200
190IF CMD$ = "" THEN GOTO 200
195CALL IBCMD(BRD0%,CMD1$): CALL IBWRT(BRD0%,CMD$): GOSUB 270
200WEND
```

```
205CALL IBSIC(BRD0%): V%=0: CALL IBSRE(BRD0%,V%)
210CALL IBSIC(BRD0%)
215END
220'
230'LOCAL MODE
235'
240V% = 0: CALL IBSRE(BRD0%,V%): PRINT ""
245RETURN
250'
260'SUBROUTINE GET_DATA
265'
270CALL IBCMD(BRD0%,CMD2$): CALL IBRD(BRD0%,RD$): I=IBCNT%
275FOR J=1 TO I: PRINT MID$(RD$,J,1);: NEXT J: PRINT ""
280RETURN
285'
290'SUBROUTINE STORE_DATA
295'
300RD1$=SPACE$(3)
305LINE INPUT "Specify trace (TA...TD,M1...M4,C1...C4):  ",TRACE$
310LINE INPUT "Enter filename : ",FILE$
315CALL IBCMD(BRD0%,CMD1$)
320CMD$="WFSU NP,0,SP,0,FP,0,SN,0;CHDR SHORT"
321CALL IBWRT(BRD0%,CMD$)
325CMD$=TRACE$+":WF?": CALL IBWRT(BRD0%,CMD$)
330CALL IBCMD(BRD0%,CMD2$): CALL IBRD(BRD0%,RD1$)
335CALL IBRDF(BRD0%,FILE$)
340IF IBSTA% < 0 THEN GOTO 430
345PRINT ""
350RETURN
355'
360'SUBROUTINE RECALL_DATA
```

```
365'
370LINE INPUT "Specify target memory (M1...M4): ",MEM$
375LINE INPUT "Enter filename : ",FILE$
380CALL IBCMD(BRD0%,CMD1$)
385CMD$=MEM$+":": CALL IBWRT(BRD0%,CMD$)
390CALL IBWRTF(BRD0%,FILE$)
395IF IBSTA% < 0 THEN GOTO 430
400PRINT ""
405RETURN
410'
415'ERROR HANDLER
420'
425PRINT "IBFIND ERROR": STOP
430PRINT "GPIB ERROR -- IBERR : ";IBERR%;"IBSTA : ";HEX$(IBSTA%)
435STOP
440END
```

NOTE: The Template also describes an array named DUAL This is simply a way to allow you to use the INSPECT? query to examine the two data arrays together.

.

Waveform Template

This template is the oscilloscope's response to a TMPL? query:

```
/00
000000              LECROY_2_3:  TEMPLATE
                    8 66 111
;
; Explanation of the formats of waveforms and their descriptors on the
; LeCroy Digital Oscilloscopes,
;     Software Release 8.1.0, 98/09/29.
;
; A descriptor and/or a waveform consists of one or several logical data blocks
; whose formats are explained below.
; Usually, complete waveforms are read: at the minimum they consist of
;       the basic descriptor block WAVEDESC
;       a data array block.
; Some more complex waveforms, e.g. Extrema data or the results of a Fourier
; transform, may contain several data array blocks.
; When there are more blocks, they are in the following sequence:
;       the basic descriptor block WAVEDESC
;       the history text descriptor block USERTEXT (may or may not be present)
;       the time array block (for RIS and sequence acquisitions only)
;       data array block
;       auxiliary or second data array block
;
; In the following explanation, every element of a block is described by a
; single line in the form
;
; <byte position>   <variable name>: <variable type> ; <comment>
;
;  where
;
;   <byte position> = position in bytes (decimal offset) of the variable,
;                     relative to the beginning of the block.
;
;   <variable name> = name of the variable.
;
;   <variable type> = string        up to 16-character name
;                                   terminated with a null byte
;                       byte        08-bit signed data value
;                       word        16-bit signed data value
;                       long        32-bit signed data value
;                      float        32-bit IEEE floating point value
;  with the format shown below
```

```
;                                   31  30 .. 23   22 ... 0   bit position
;                                   s   exponent   fraction
;                                   where
;                                   s = sign of the fraction
;                                   exponent = 8 bit exponent e
;                                   fraction = 23 bit fraction f
;                                   and the final value is
;                                   (-1)**s * 2**(e-127) * 1.f
;                       double      64-bit IEEE floating point value
;                                   with the format shown below
;                                   63  62 .. 52   51 ... 0   bit position
;                                   s   exponent   fraction
;                                   where
;                                   s = sign of the fraction
;                                   exponent = 11 bit exponent e
;                                   fraction = 52 bit fraction f
;                                   and the final value is
;                                   (-1)**s * 2**(e-1023) * 1.f
;                         enum      enumerated value in the range 0 to N
;                                   represented as a 16-bit data value.
;                                   The list of values follows immediately.
;                                   The integer is preceded by an _.
;                   time_stamp      double precision floating point number,
;                                   for the number of seconds and some bytes
;                                   for minutes, hours, days, months and year.
;
;                                   double  seconds     (0 to 59)
;                                   byte    minutes     (0 to 59)
;                                   byte    hours       (0 to 23)
;                                   byte    days        (1 to 31)
;                                   byte    months      (1 to 12)
;                                   word    year        (0 to 16000)
;                                   word    unused
;                                   There are 16 bytes in a time field.
;                         data      byte, word or float, depending on the
;                                   read-out mode reflected by the WAVEDESC
;                                   variable COMM_TYPE, modifiable via the
;                                   remote command COMM_FORMAT.
;                         text      arbitrary length text string
;                                   (maximum 160)
;               unit_definition     a unit definition consists of a 48 character
;                                   ASCII string terminated with a null byte
;                                   for the unit name.
;
;===============================================================================
;
WAVEDESC: BLOCK
;
; Explanation of the wave descriptor block WAVEDESC;
;
```

```
;
<  0>          DESCRIPTOR_NAME: string  ; the first 8 chars are always WAVEDESC
;
< 16>          TEMPLATE_NAME: string
;
< 32>          COMM_TYPE: enum          ; chosen by remote command COMM_FORMAT
               _0      byte
               _1      word
               endenum
;
< 34>          COMM_ORDER: enum
               _0      HIFIRST
               _1      LOFIRST
               endenum
;
;
; The following variables of this basic wave descriptor block specify
; the block lengths of all blocks of which the entire waveform (as it is
; currently being read) is composed. If a block length is zero, this
; block is (currently) not present.
;
; Blocks and arrays that are present will be found in the same order
; as their descriptions below.
;
;BLOCKS :
;
< 36>          WAVE_DESCRIPTOR: long    ; length in bytes of block WAVEDESC
< 40>          USER_TEXT: long          ; length in bytes of block USERTEXT
< 44>          RES_DESC1: long          ;
;
;ARRAYS :
;
< 48>          TRIGTIME_ARRAY: long     ; length in bytes of TRIGTIME array
;
< 52>          RIS_TIME_ARRAY: long     ; length in bytes of RIS_TIME array
;
< 56>          RES_ARRAY1: long         ; an expansion entry is reserved
;
< 60>          WAVE_ARRAY_1: long       ; length in bytes of 1st simple
                                        ; data array. In transmitted waveform,
                                        ; represent the number of transmitted
                                        ; bytes in accordance with the NP
                                        ; parameter of the WFSU remote command
                                        ; and the used format (see COMM_TYPE).
;
< 64>          WAVE_ARRAY_2: long       ; length in bytes of 2nd simple
                                        ; data array
;
< 68>          RES_ARRAY2: long
< 72>          RES_ARRAY3: long         ; 2 expansion entries are reserved
```

```
;
; The following variables identify the instrument
;
< 76>          INSTRUMENT_NAME: string
;
< 92>          INSTRUMENT_NUMBER: long
;
< 96>          TRACE_LABEL: string      ; identifies the waveform.
;
<112>          RESERVED1: word
<114>          RESERVED2: word          ; 2 expansion entries
;
; The following variables describe the waveform and the time at
; which the waveform was generated.
;
<116>          WAVE_ARRAY_COUNT: long   ; number of data points in the data
                                        ; array. If there are two data
                                        ; arrays (FFT or Extrema), this number
                                        ; applies to each array separately.
;
<120>          PNTS_PER_SCREEN: long    ; nominal number of data points
                                        ; on the screen
;
<124>          FIRST_VALID_PNT: long    ; count of number of points to skip
                                        ; before first good point
                                        ; FIRST_VALID_POINT = 0
                                        ; for normal waveforms.
;
<128>          LAST_VALID_PNT: long     ; index of last good data point
                                        ; in record before padding (blanking)
                                        ; was started.
                                        ; LAST_VALID_POINT = WAVE_ARRAY_COUNT-1
                                        ; except for aborted sequence
                                        ; and rollmode acquisitions
;
<132>          FIRST_POINT: long        ; for input and output, indicates
                                        ; the offset relative to the
                                        ; beginning of the trace buffer.
                                        ; Value is the same as the FP parameter
                                        ; of the WFSU remote command.
;
<136>          SPARSING_FACTOR: long    ; for input and output, indicates
                                        ; the sparsing into the transmitted
                                        ; data block.
                                        ; Value is the same as the SP parameter
                                        ; of the WFSU remote command.
;
<140>          SEGMENT_INDEX: long      ; for input and output, indicates the
                                        ; index of the transmitted segment.
                                        ; Value is the same as the SN parameter
```

```
                                        ; of the WFSU remote command.
;
<144>          SUBARRAY_COUNT: long     ; for Sequence, acquired segment count,
                                        ; between 0 and NOM_SUBARRAY_COUNT
;
<148>          SWEEPS_PER_ACQ: long     ; for Average or Extrema,
                                        ; number of sweeps accumulated
                                        ; else 1
;
<152>          POINTS_PER_PAIR: word    ; for Peak Dectect waveforms (which
always
                                        ; include data points in DATA_ARRAY_1
and
                                        ; min/max pairs in DATA_ARRAY_2).
                                        ; Value is the number of data points
for
                                        ; each min/max pair.
;
<154>          PAIR_OFFSET: word        ; for Peak Dectect waveforms only
                                        ; Value is the number of data points by
                                        ; which the first min/max pair in
                                        ; DATA_ARRAY_2 is offset relative to
the
                                        ; first data value in DATA_ARRAY_1.
;
<156>          VERTICAL_GAIN: float
;
<160>          VERTICAL_OFFSET: float   ; to get floating values from raw data
:
                                        ; VERTICAL_GAIN * data -
VERTICAL_OFFSET
;
<164>          MAX_VALUE: float         ; maximum allowed value. It corresponds
                                        ; to the upper edge of the grid.
;
<168>          MIN_VALUE: float         ; minimum allowed value. It corresponds
                                        ; to the lower edge of the grid.
;
<172>          NOMINAL_BITS: word       ; a measure of the intrinsic precision
                                        ; of the observation: ADC data is 8 bit
                                        ;    averaged data is 10-12 bit, etc.
;
<174>          NOM_SUBARRAY_COUNT: word ; for Sequence, nominal segment count
                                        ; else 1
;
<176>          HORIZ_INTERVAL: float    ; sampling interval for time domain
                                        ;  waveforms
;
<180>          HORIZ_OFFSET: double     ; trigger offset for the first sweep of
                                        ; the trigger, seconds between the
```

```
                                    ; trigger and the first data point
;
<188>          PIXEL_OFFSET: double     ; needed to know how to display the
                                        ; waveform
;
<196>          VERTUNIT: unit_definition  ; units of the vertical axis
;
<244>          HORUNIT: unit_definition   ; units of the horizontal axis
;
<292>          HORIZ_UNCERTAINTY: float ; uncertainty from one acquisition to the
                                        ; next, of the horizontal offset in seconds
;
<296>          TRIGGER_TIME: time_stamp ; time of the trigger
;
<312>          ACQ_DURATION: float      ; duration of the acquisition (in sec)
                                        ; in multi-trigger waveforms.
                                        ; (e.g. sequence, RIS,  or averaging)
;
<316>          RECORD_TYPE: enum
               _0      single_sweep
               _1      interleaved
               _2      histogram
               _3      graph
               _4      filter_coefficient
               _5      complex
               _6      extrema
               _7      sequence_obsolete
               _8      centered_RIS
               _9      peak_detect
               endenum
;
<318>          PROCESSING_DONE: enum
               _0       no_processing
               _1       fir_filter
               _2       interpolated
               _3       sparsed
               _4       autoscaled
               _5       no_result
               _6       rolling
               _7       cumulative
               endenum
;
<320>          RESERVED5: word          ; expansion entry
;
<322>          RIS_SWEEPS: word         ; for RIS, the number of sweeps
                                        ; else 1
;
; The following variables describe the basic acquisition
; conditions used when the waveform was acquired
;
```

```
<324>          TIMEBASE: enum
               _0    1_ps/div
               _1    2_ps/div
               _2    5_ps/div
               _3    10_ps/div
               _4    20_ps/div
               _5    50_ps/div
               _6    100_ps/div
               _7    200_ps/div
               _8    500_ps/div
               _9    1_ns/div
               _10   2_ns/div
               _11   5_ns/div
               _12   10_ns/div
               _13   20_ns/div
               _14   50_ns/div
               _15   100_ns/div
               _16   200_ns/div
               _17   500_ns/div
               _18   1_us/div
               _19   2_us/div
               _20   5_us/div
               _21   10_us/div
               _22   20_us/div
               _23   50_us/div
               _24   100_us/div
               _25   200_us/div
               _26   500_us/div
               _27   1_ms/div
               _28   2_ms/div
               _29   5_ms/div
               _30   10_ms/div
               _31   20_ms/div
               _32   50_ms/div
               _33   100_ms/div
               _34   200_ms/div
               _35   500_ms/div
               _36   1_s/div
               _37   2_s/div
               _38   5_s/div
               _39   10_s/div
               _40   20_s/div
               _41   50_s/div
               _42   100_s/div
               _43   200_s/div
               _44   500_s/div
               _45   1_ks/div
               _46   2_ks/div
               _47   5_ks/div
               _100  EXTERNAL
```

```
                endenum
;
<326>           VERT_COUPLING: enum
                _0      DC_50_Ohms
                _1      ground
                _2      DC_1MOhm
                _3      ground
                _4      AC,_1MOhm
                endenum
;
<328>           PROBE_ATT: float
;
<332>           FIXED_VERT_GAIN: enum
                _0   1_uV/div
                _1   2_uV/div
                _2   5_uV/div
                _3   10_uV/div
                _4   20_uV/div
                _5   50_uV/div
                _6   100_uV/div
                _7   200_uV/div
                _8   500_uV/div
                _9   1_mV/div
                _10  2_mV/div
                _11  5_mV/div
                _12  10_mV/div
                _13  20_mV/div
                _14  50_mV/div
                _15  100_mV/div
                _16  200_mV/div
                _17  500_mV/div
                _18  1_V/div
                _19  2_V/div
                _20  5_V/div
                _21  10_V/div
                _22  20_V/div
                _23  50_V/div
                _24  100_V/div
                _25  200_V/div
                _26  500_V/div
                _27  1_kV/div
                endenum
;
<334>           BANDWIDTH_LIMIT: enum
                _0      off
                _1      on
                endenum
;
<336>           VERTICAL_VERNIER: float
;
```

```
<340>          ACQ_VERT_OFFSET: float
;
<344>          WAVE_SOURCE: enum
               _0      CHANNEL_1
               _1      CHANNEL_2
               _2      CHANNEL_3
               _3      CHANNEL_4
               _9      UNKNOWN
               endenum
;
/00              ENDBLOCK
;
;===============================================================================
;
USERTEXT: BLOCK
;
; Explanation of the descriptor block USERTEXT at most 160 bytes long.
;
;
<  0>          TEXT: text               ; a list of ASCII characters
;
/00              ENDBLOCK
;
;===============================================================================
;
TRIGTIME: ARRAY
;
; Explanation of the trigger time array TRIGTIME.
; This optional time array is only present with SEQNCE waveforms.
; The following data block is repeated for each segment which makes up
; the acquired sequence record.
;
<  0>          TRIGGER_TIME: double     ; for sequence acquisitions,
                                        ; time in seconds from first
                                        ; trigger to this one
;
<  8>          TRIGGER_OFFSET: double   ; the trigger offset is in seconds
                                        ; from trigger to zeroth data point
;
/00              ENDARRAY
;
;===============================================================================
;
RISTIME: ARRAY
;
; Explanation of the random-interleaved-sampling (RIS) time array RISTIME.
; This optional time array is only present with RIS waveforms.
; This data block is repeated for each sweep which makes up the RIS record
;
<  0>          RIS_OFFSET: double       ; seconds from trigger to zeroth
```

```
                                        ; point of segment
;
/00              ENDARRAY
;
;===============================================================================
;
DATA_ARRAY_1: ARRAY
;
; Explanation of the data array DATA_ARRAY_1.
; This main data array is always present. It is the only data array for
; most waveforms.
; The data item is repeated for each acquired or computed data point
; of the first data array of any waveform.
;
<  0>           MEASUREMENT: data        ; the actual format of a data is
                                        ; given in the WAVEDESC descriptor
                                        ; by the COMM_TYPE variable.
;
/00              ENDARRAY
;
;===============================================================================
;
DATA_ARRAY_2: ARRAY
;
; Explanation of the data array DATA_ARRAY_2.
; This is an optional secondary data array for special types of waveforms:
;       Complex FFT     imaginary part      (real part in DATA_ARRAY_1)
;       Extrema         floor trace         (roof trace in DATA_ARRAY_1)
;       Peak Detect     min/max pairs       (data values in DATA_ARRAY_1)
; In the first 2 cases, there is exactly one data item in DATA_ARRAY_2 for
; each data item in DATA_ARRAY_1.
; In Peak Detect waveforms, there may be fewer data values in DATA_ARRAY_2,
; as described by the variable POINTS_PER_PAIR.
;
<  0>           MEASUREMENT: data        ; the actual format of a data is
                                        ; given in the WAVEDESC descriptor
                                        ; by the COMM_TYPE variable.
;
/00              ENDARRAY
;
;===============================================================================
;
SIMPLE: ARRAY
;
; Explanation of the data array SIMPLE.
; This data array is identical to DATA_ARRAY_1. SIMPLE is an accepted
; alias name for DATA_ARRAY_1.
;
<  0>           MEASUREMENT: data        ; the actual format of a data is
                                        ; given in the WAVEDESC descriptor
```

```
                                        ; by the COMM_TYPE variable.
;
/00             ENDARRAY
;
;===============================================================================
;
DUAL: ARRAY
;
; Explanation of the DUAL array.
; This data array is identical to DATA_ARRAY_1, followed by DATA_ARRAY_2.
; DUAL is an accepted alias name for the combined arrays DATA_ARRAY_1 and
; DATA_ARRAY_2 (e.g. real and imaginary parts of an FFT).
;
<  0>          MEASUREMENT_1: data      ; data in DATA_ARRAY_1.
;
<  0>          MEASUREMENT_2: data      ; data in DATA_ARRAY_2.
;
/00               ENDARRAY
;
;
000000               ENDTEMPLATE
```

BLANK PAGE

A

Addresses, 14
ALL_STATUS?, ALST?, Query, 65
ARM_ACQUISITION, ARM, Command, 66
ASCII
  waveform storage, 10
ATTENUATION, ATTN, Command/ Query, 67
AUTO_CALIBRATE, ACAL, Command/ Query, 68
AUTO_SCROLL, ASCR, Command/ Query, 69
AUTO_SETUP, ASET, Command, 70

B

BANDWIDTH_LIMIT, BWL, Command/ Query, 71
BASICA, 17, 18, 37
Binary blocks, 39
Block Data, 10
Buffers, 14
BUZZER, BUZZ, Command, 73

C

*CAL?, Query, 74
CAL_OUTPUT, COUT, Command/ Query, 75
CALL_HOST, CHST, Command/ Query, 77
Character data, 9
CLEAR_MEMORY, CLM, Command, 78
CLEAR_SWEEPS, CLSW, Command, 79
*CLS, Command, 80
CMR (Command Error Status Register), 47, 49
CMR?, Query, 81
COLOR, COLR, Command/ Query, 83
COLOR_SCHEME, CSCH, Command/ Query, 86
Colors
  list of colors and their short form names, 84
COMBINE_CHANNELS, COMB, Command/ Query, 87
COMM_FORMAT, CFMT, Command/ Query, 88
COMM_HEADER, CHDR, Command/ Query, 90
COMM_HELP, CHLP, Command/ Query, 91
COMM_HELP_LOG?, CHL?, Query, 92
COMM_NET, CONET, Command/ Query, 93
COMM_ORDER, CORD, Command/ Query, 94
COMM_RS232, CORS, Command/ Query, 96
Commands and Queries, 6
  How they are described, 53
  Notation, 53
  Overview, 53
Configuring
  Printing, 26
Continuous Polling, 21
Controller Timeout, 7, 14, 18, 20
COUPLING, CPL, Command/ Query, 99
CURSOR_MEASURE, CRMS, Command/ Query, 100
CURSOR_READOUT, CRRD, Command/ Query, 103
CURSOR_SET, CRST, Command/ Query, 104
CURSOR_VALUE?, CRVA?, Query, 106

D

Data
  Arrays, 36
  Formatting, 37, 42
  HEX mode, 39, 42
  Horizontal position, 40
  Interpretation, 37, 39
  Sparsing, 40,42
  Values, 36, 39
  Vertical reading, 39
DATA_POINTS, DPNT, Command/ Query, 108
DATE, Command/ Query, 109
DDR (Device Dependent Error Status Register), 49
DDR?, Query, 110
DEFINE, DEF, Command/ Query, 112
DELETE_FILE, DELF, Command, 118

Descriptor
Block, 35
Values, 36, 39
Device Dependent Error Status Register. see DDR
DIRECTORY, DIR, Command/ Query, 119
DISPLAY, DISP, Command/ Query, 121
DOT_JOIN, DTJN, Command/ Query, 122
Dual Array, 36, 37
DUAL_ZOOM, DZOM, Command/ Query, 123

E

ENABLE_KEY, Command/ Query, 124
Error Messages, 6
ESE (Standard Event Status Enable Register), 19, 47, 49
*ESE, Command/ Query, 125
ESR (Standard Event Status Register), 19, 47, 49
*ESR?, Query, 126
EXR (Execution Error Status Register), 49
EXR?, Query, 129
External Monitor port, 5, 6

F

FAT_CURSOR, FATC, Command/ Query, 131
FILENAME, FLNM, Command/ Query, 132
FIND_CTR_RANGE, FCR, Command, 133
FORMAT_CARD, FCRD, Command/ Query, 134
FORMAT_FLOPPY, FFLP, Command/ Query, 136
FORMAT_HDD, FHDD, Command/ Query, 138
FORMAT_VDISK, FVDISK, Command/ Query, 140
FULL_SCREEN, FSCR, Command/ Query, 141
FUNCTION_RESET, FRST, Command, 142

G

GLOBAL_BWL, GBWL, Command/ Query, 143
GPIB
Addresses, 14
ATN (ATteNtion), 14
Data lines, 14
DCL (Device CLear), 14, 15
EOI (End Or Identify), 6, 14
GET (Group Execute Trigger), 15, 19
GTL (Go To Local), 15, 16, 19
Handshake lines, 14
Hard copy, 25, 26
Hardware configuration, 14
IEEE 488.1, 13, 15
IEEE 488.2, 6, 47, 49
IFC (InterFace Clear), 14, 16
INE (INternal State Change Enable Register), 20, 22, 23
INR (INternal State Change Status Register), 21, 22, 24
Listener address, 24, 26
LLO (Local LOckout), 15, 16
MLA (Listen address), 14
MTA (Talker address), 14
Polling, 21
Port, 5
PRE (PaRallel Poll Enable Register), 22
Printing, 26
Programming service requests, 19, 20
Programming transfers, 17, 18
REN (Remote ENable), 15, 16
RQS (ReQuest for Service), 20
SDC (Selected Device Clear), 15, 19
Signals, 14
Software configuration, 16
SRE (Service Request Enable Register), 20
SRQ (Service ReQuest), 14, 19, 20
Standard, 6
Structure, 13
Talker address, 24, 26
Transfers, 17
UNL (Universal unlisten), 14, 24, 26, 27
UNT (Universal untalk), 14, 24, 26, 27
GRID, Command/ Query, 144

H

Hard copies. see GPIB
HARDCOPY_SETUP, HCSU, 145
HARDCOPY_TRANSMIT, HCTR, Command, 148
Header, 7, 8
Header Path, 8

Help Messages, 6
HOR_MAGNIFY, HMAG, Command/ Query, 149
HOR_POSITION, HPOS, Command/ Query, 150

I

*IDN?, Query, 152
IEEE 488.1, 13, 15
IEEE 488.2, 6, 45, 47, 49
IEEE Standards. see GPIB
INE (INternal State Change Enable Register), 20, 47, 49
INE, Command/ Query, 153
INR (INnternal State Change Status Register), 20, 48
INR?, Query, 154
INSPECT? Queries, 36
INSPECT?, INSP?, Query, 156
INTENSITY, INTS, Command/ Query, 158
Interface capabilities, 13
Interface messages, 13
INTERLEAVED, ILVD, Command/ Query, 159
Internal State Change Enable Register. see INE
Internal State Change Status Register. see INR
IST Polling, 24, 45, 48
*IST?, Query, 160

K

KEY, Command, 161

L

Line Splitting. see RS-232-C
Logical Data Blocks, 35
LOGO, Command/ Query, 162

M

MASK, Command/ Query, 163
Math functions
  CLEAR_MEMORY, CLM, Command, 78
  CLEAR_ SWEEPS, CLSW, Command, 79
  DEFINE, DEF, Command/ Query, 112
  FIND_CTR_RANGE, FCR, Command, 133
  FUNCTION_RESET, FRST, Command, 142
MathCad®, 216
MATH_LIMITS, MLIM, Command/ Query, 165
MEASURE_GATE, MGAT, Command/ Query, 166
MEMORY_SIZE, MSIZ, Command/ Query, 167
MESSAGE, MSG, Command/ Query, 168
MULTI_ZOOM, MZOM, Command/ Query, 169
Multipliers, 9

N

Numeric Data, 9

O

OFFSET, OFST, Command/ Query, 170
OFFSET_CONSTANT, OFCT, Command/ Query, 171
*OPC, Command/ Query, 172
*OPT?, Query, 173
OPTIMIZE, OPMZ, Command/ Query, 175

P

PANEL_SETUP, PNSU, Command/ Query, 176
Parallel Poll Enable Register. see PRE
Parallel Polling, 22
Parameter measurements, 100
PARAMETER_CLR, PACL, Command, 177
PARAMETER_CUSTOM, PACU, Command/ Query, 178
PARAMETER_DELETE, PADL, Command, 181
PARAMETER_STATISTICS?, PAST?, Query, 182
PARAMETER_VALUE?, PAVA?, Query, 183
PASS_FAIL_CONDITION, PFCO, Command/ Query, 186
PASS_FAIL_COUNTER, PFCT, Command/ Query, 188
PASS_FAIL_DO, PFDO, Command/ Query, 189
PASS_FAIL_MASK, PFMS, Command, 191
PASS_FAIL_STATUS?, PFST?, Query, 192

PER_CURSOR_SET, PECS, Command/ Query, 193
PER_CURSOR_VALUE?, PECV?, Query, 195
PERSIST, PERS, Command/ Query, 196
PERSIST_COLOR, PECL, Command/ Query, 197
PERSIST_LAST, PELT, Command/ Query, 198
PERSIST_SAT, PESA, Command/ Query, 199
PERSIST_SETUP, PESU, Command/ Query, 200
Polling, 21
  Continuous, 21
  IST Polling, 24
  Parallel, 22
  Serial, 21
PRE (Parallel Poll Enable Register), 21, 48
*PRE, Command/ Query, 201
PROBE_CAL?, PRCA?, Query, 202
PROBE_DEGAUSS?, PRDG?, Query, 203
PROBE_INFOTEXT?, PRIT?, Query, 204
PROBE_NAME?, PRNA?, Query, 205
Program Messages, 6

Q

Quotation marks
  their use in command notation, 10

R

*RCL, Command, 206
REAR_OUTPUT, ROUT, Command/ Query, 207
RECALL, REC, Command, 208
RECALL_PANEL, RCPN, Command, 209
Response Messages, 10
RIS Acquisition Times (RISTIME), 35, 41
RQS (ReQuest for Service), 20
RS-232-C
  Configuration, 29
  Echo, 29
  Editing, 30
  Handshake control, 29
  Immediate commands, 29
  Line splitting, 31
  Message terminators, 30
  Port, 5, 6
  Simulating GPIB Commands, 31
  SRQ (Service ReQuest), 31
RS-232-C computer cabling, 11
*RST, Command, 210

S

SAMPLE_CLOCK, SCLK, Command/ Query, 211
*SAV, Command, 212
ScopeExplorer™, 11
SCREEN Command, 213
SCREEN_DUMP, SCDP, Command/ Query, 214
SCREEN_SAVE, SCSV, Command/ Query, 215
SELECT, SEL, Command/ Query, 216
SEQUENCE, SEQ, Command/ Query, 217
Serial Polling, 21
Service Request Enable Register. see SRE
Service Request Reporting, 48
Service requests, 31
SIMPLE, 36
SKEY Command, 219
SLEEP, Command, 220
SRE (Service Request Enable Register), 18, 45, 46, 48
*SRE, Command/ Query, 221
SRQ (Service Request), 18, 19, 30, 49
Standard Event Status Register. see ESR
Status Byte Register. see STB
Status Register Reporting, 47
STB (Status Byte Register), 18, 49
*STB?, Query, 222
STITLE Command, 224
STOP, Command, 225
STORE, STO, Command, 226
STORE_PANEL, STPN, Command, 227
STORE_SETUP, STST, Command/ Query, 228
STORE_TEMPLATE, STTM, Command, 229
String Data, 10

T

Talker, 13, 14, 24, 26
TDISP, Command/ Query, 230
Template, 35, 36, 39, 40

TEMPLATE?, TMPL?, Query, 231
Terminators, 6, 30, 39
TIME_DIV, TDIV, Command/ Query, 232
TRACE, TRA, Command/ Query, 233
TRACE_LABEL, TRBL, Command/ Query, 234
TRACE_OPACITY, TOPA, Command/ Query, 235
TRACE_ORDER, TORD, Command/ Query, 236
TRANSFER_FILE, TRFL, Command/ Query, 237
*TRG, Command, 238
TRIG_COUPLING, TRCP, Command/ Query, 239
TRIG_DELAY, TRDL, Command/ Query, 240
TRIG_LEVEL, TRLV, Command/ Query, 242
TRIG_MODE, TRMD, Command/ Query, 243
TRIG_PATTERN, TRPA, Command/ Query, 244
TRIG_SELECT, TRSE, Command/ Query, 246
TRIG_SLOPE, TRSL, Command/ Query, 249
TRIG_WINDOW, TRWI, Command/ Query, 250
Trigger Times (TRIGTIME), 35, 41
*TST?, Query, 251

U

URR (User Request Status Register), 49
URR?, Query, 252
User Request Status Register. see URR
USERTEXT, 35

V

VERT_MAGNIFY, VMAG, Command/ Query, 254
VERT_POSITION, VPOS, Command/ Query, 255
VOLT_DIV, VDIV, Command/ Query, 256

W

*WAI, Command, 257
WAIT, Command, 258
Warning Messages, 6
WAVEDESC. see Descriptor
WAVEFORM
  Command, 42
  Query, 37, 42
  Transfer optimization, 43
WAVEFORM, WF, Command/ Query, 259
WAVEFORM_SETUP, WFSU, Command/ Query, 262
WAVEFORM_TEXT, WFTX, Command/ Query, 264

X

XY_ASSIGN?, XYAS?, Query, 265
XY_CURSOR_ORIGIN, XYCO, Command/ Query, 266
XY_CURSOR_SET, XYCS, Command/ Query, 267
XY_CURSOR_VALUE?, XYCV?, Query, 269
XY_DISPLAY, XYDS, Command/ Query, 271
XY_RENDER, XYRD, Command/ Query, 272
XY_SATURATION, XYSA, Command/ Query, 273