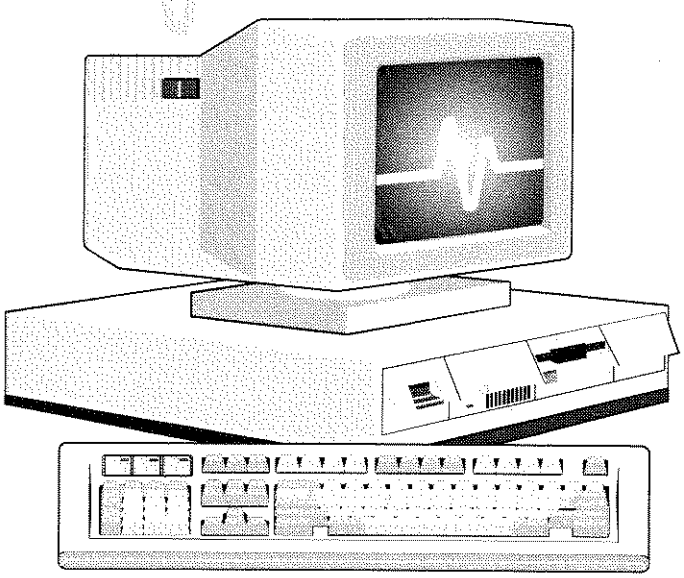
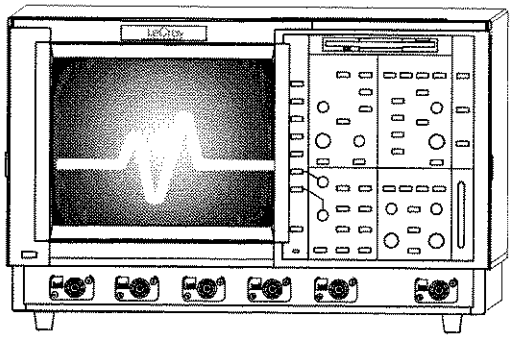


Remote Control Manual



LeCroy Digital Oscilloscopes

Revision O — April 1999

LeCroy Corporation
700 Chestnut Ridge Road
Chestnut Ridge, NY 10977–6499
Tel: (914) 578 6020, Fax: (914) 578 5985

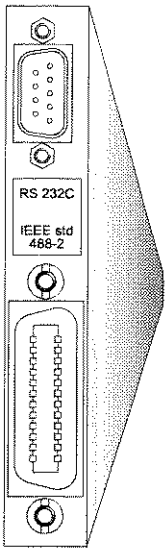
LeCroy SA
2, rue du Pré-de-la-Fontaine
1217 Meyrin 1/Geneva, Switzerland
Tel: (41) 22 719 21 11, Fax: (41) 22 782 39 15

Internet: www.lecroy.com

LCXXX-RM-E Rev O 0399

Contents

How to Operate the Oscilloscope Remotely



Illustrated above and below-right, the GPIB (IEEE Std 488-2)and RS-232-C ports found on the back of LeCroy oscilloscopes, used for connecting the instrument to an external controller. They are vertically or horizontally arranged on the back panel according to model.

Your LeCroy oscilloscope can of course be operated *manually*, using the front-panel controls (*see the accompanying Operator's Manual*). But it can also be operated *remotely* by means of an external controller. Normally, this controller will be a computer. However, it may be a simple terminal.

The present manual describes how to remotely control the oscilloscope. Its main section provides the system commands for executing the instrument's functions from an external controller.

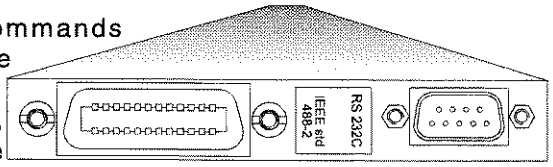
Remote control is done using either the GPIB (General Purpose Interface Bus) — labeled "IEEE Std 488-2" — or the RS-232-C communication port on the rear panel of the oscilloscope. The instrument can be fully controlled in remote mode, the only actions not able to be performed being the powering-on of the oscilloscope and the setting of remote addresses.

In this chapter, the basic remote control concepts common to both GPIB and RS-232-C are introduced. Also presented is a brief description of the remote control messages. The following two chapters set out how to send program messages over the GPIB and RS-232-C interfaces, respectively. Chapter 4 offers a detailed description and run-through of the transfer and formatting of waveforms. While Chapter 5 explains the use of status bytes for error reporting.

The special *System Commands* section provides a complete directory and description of the system commands. And the Appendices offer GPIB Program Examples (Appendix A) and a Waveform Template (Appendix B).

GPIB Standard

The remote commands conform to the GPIB IEEE 488.2* standard, which may be



*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.

considered as an extension of the IEEE 488.1 standard, dealing mainly with electrical and mechanical issues. The IEEE 488.2 recommendations have also been adopted for RS-232-C communications wherever applicable.

Program Messages

To control the oscilloscope remotely, the program messages sent from the external controller must conform to precise format structures. The oscilloscope will execute all program messages sent in the correct form, but will ignore those where errors are detected.

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.

During the development of the control program it is possible to observe all remote control transactions, including error messages, on an external monitor connected to the RS-232-C port. *Refer to the command "COMM_HELP" for further details.*

Commands and Queries

Program messages consist of either one or several commands or queries.

Whereas the command directs the instrument to change its state — its timebase or vertical sensitivity, for example — the query asks the instrument about that state. Very often, the same mnemonic is used 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, the controller sends the following command to the instrument:

```
TIME_DIV 2 MS
```

To ask the instrument about its timebase, this query should be sent:

```
TIME_DIV?
```

A query causes the instrument 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 query above 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 may be suppressed with the command "COMM_HEADER".

Depending on the state of the instrument and the computation to be done, up to several seconds may pass before a response is received. Command interpretation does not have priority over other oscilloscope activities. It is therefore judicious to set the controller IO timeout conditions to three or more seconds. In addition, it should always be remembered that an incorrect query message will not generate a response message.

Program Message Form

An instrument is remotely controlled with program messages that consist of one or several commands or queries, separated by semicolons <;> and ending in a terminator:

<command/query>;.....;<command/query>
<terminator>

Upper or lower-case characters or both can be used in program messages.

The instrument does not decode incoming program messages before receiving a terminator. The exception to this is when the program message is longer than the 256 byte input buffer: 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.

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.

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

In RS-232-C, the terminator may be defined by the user with the command "COMM_RS232". The default value is <CR>, i.e. the ASCII carriage return character, the decimal value of which is 13.

Examples

```
GRID DUAL
```

This program message consists of a single command that instructs the instrument to display a dual grid. The terminator is not shown, as it is usually automatically added by the interface driver routine writing to the GPIB (or RS-232).

```
DZOM ON; DISPLAY OFF; DATE?
```

This program message consists of two commands, followed by a query. They instruct the instrument to turn on the multi-zoom mode, turn off the display, and then ask for the current date. Again, the terminator is not shown.

Command/Query Form

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). The detailed listing of all commands in *System Commands* indicates which may also be queries. There is a space between the header and the first parameter. There are commas between parameters.

Example

```
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.

Header

The header is the mnemonic form of the operation to be performed by the oscilloscope. All command mnemonics are listed in alphabetic order in the System Commands section.

The majority of the command/query headers have a long form for optimum legibility and a short form for better transfer and decoding speed. The two forms are fully equivalent and can be

used interchangeably. For example, the following two commands for switching to the automatic trigger mode are fully equivalent:

`TRIG_MODE AUTO` and `TRMD AUTO`

Some command/query mnemonics are imposed by the IEEE 488.2 standard. They are standardized so that different instruments 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 instrument, whereas "*TST?" instructs the instrument to perform an internal self-test and to report the outcome.

Header path

Some 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, the header must be preceded 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 (refer to the individual commands listed in *System Commands* for details of the values applying to given command headers):

`C1, C2`	Channels 1 and 2
`C3, C4`	Channels 3 and 4†
`M1, M2, M3, M4`	Memories 1, 2, 3, 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
```

Commands to set the offset of Channel 1 to –300 mV.

Header paths need only be specified once. Subsequent commands with header destinations not indicated are assumed to refer to the last defined path. For example, the following commands are identical:

† *On four-channel instruments only.*

`C2:VDIV?; C2:OFST?`	What is the vertical sensitivity and the offset of channel 2?
`C2:VDIV?; OFST?`	Same as above, without repeating the path.

Data

Whenever a command/query uses additional data values, the values are expressed in terms of ASCII characters. There is a single exception: the transfer of waveforms with the command/query "WAVEFORM", where the waveform may be expressed as a sequence of binary data values. *Chapter 4 gives a detailed explanation of waveform format.*

ASCII data can have the form of character, numeric, string or block data.

Character data

These are simple words or abbreviations for the indication of a specific action.

Example

```
DUAL_ZOOM ON
```

Here, the data value "ON" commands that the dual-zoom mode be turned on (the data value "OFF" in such a case will obviously have the opposite effect).

However, this can become more complex. In some commands, where as many as a dozen different parameters are able to be specified, 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 indicating:

Example

```
HARDCOPY_SETUP DEV,EPSON,PORT,GPIB
```

Here, two pairs of parameters are specified. The first specifies the device as the EPSON printer (or compatible) and the second indicates the GPIB port. While the command "HARDCOPY_SETUP" allows many more parameters, they are either not relevant for printers or 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.

***Note:** Numeric values may be followed by multipliers and units, modifying the value of the numerical expression. The following mnemonics are recognized:*

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-

Examples

There are many ways of setting the timebase of the instrument 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
```

String Data

This data type enables the transfer of a (long) string of characters as a single parameter. String data are formed by simply enclosing any sequence of ASCII characters between single or double quotation marks:

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

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

Block Data

These are binary data values coded in hexadecimal ASCII, i.e. 4-bit nibbles translated into the digits 0,...9, A,...F and transmitted as ASCII characters. They are used only for the transfer of waveforms (Command "WAVEFORM") and of the instrument configuration (Command "PANEL_SETUP")

Response Message Form

The instrument sends a response message to the controller, as an answer to a query. The format of such messages is the same as that of program messages, i.e. individual responses in the format of commands, separated by semicolons <;> and ending with a terminator. They can be sent back to the instrument in the form in which they are received, to be accepted as valid commands. In GPIB response messages, the <NL> <EOI> terminator is always used.

For instance, if the controller sends the program message:

```
TIME_DIV?;TRIG_MODE NORM;C1:COUPLING?
```

(terminator not shown).

The instrument might respond as follows:

```
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 instrument, 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 a response is expected from the instrument, the control program must instruct the GPIB or RS-232-C interface to read from the instrument. If the controller sends another program message without reading the response to the previous one, the response message in the output buffer of the instrument is discarded.

The instrument uses somewhat stricter rules for response messages than for the acceptance of program messages. Whereas the controller may send program messages in upper or lower case characters, response messages are always returned

in upper case. Program messages may contain extraneous spaces or tabs (white space): response messages will not. And while program messages may contain a mixture of short and long command/query headers, response messages always use short headers by default.

However, the instrument can be forced, using the command "COMM_HEADER", to use long headers, or no headers at all. If the response header is omitted, the response transfer time is minimized, but such a response will not be able to be sent back to the instrument. Suffix units are also suppressed in the response.

If the trigger slope of Channel 1 is set 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

Waveforms which are obtained from the instrument using the query "WAVEFORM?" constitute a special kind of response message. Their exact format can be controlled via the "COMM_FORMAT" and "COMM_ORDER" commands.

Communication via the GPIB Bus

This chapter describes how to remotely control the oscilloscope using the General Purpose Interface Bus (GPIB). Discussed are interface capabilities, addressing, standard bus commands, and polling schemes. *See also the "Utilities" chapter in the accompanying Operator's Manual and "Hands-On Guide".*

GPIB Structure

GPIB is similar to a standard computer bus. But whereas a computer interconnects circuit cards via a backplane bus, the GPIB interconnects independent devices 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, instrument status and waveform data. *Their general form is described in the previous chapter.*
- **Interface messages** manage the bus itself. They perform functions such as its initialization, the addressing and "unaddressing" of devices, and the setting of remote and local modes.

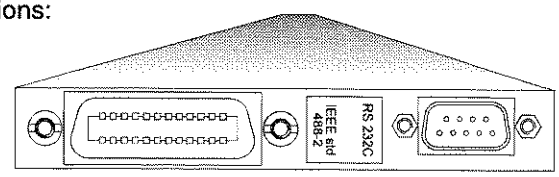
Talkers and Listeners

Devices connected by GPIB can be listeners, talkers, or controllers. A talker sends program messages to one or more listeners, while the controller manages the flow of information on the bus by sending interface messages to the devices.

The oscilloscope can be a talker or listener, but *not* a controller. The host computer must be able to play all three roles.*

Interface Capabilities

The interface capabilities of the oscilloscope include the following IEEE 488.1 definitions:



* *For details of how the controller configures the GPIB for specific functions, refer to the GPIB interface manufacturer's manual.*

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

Addressing

Every device on the GPIB has an address. When the remote control port is set to "GPIB", using the oscilloscope's front-panel-operated "UTILITIES" menus, the instrument can be controlled via GPIB. When the remote control port is set to "RS-232" by the same means, the instrument will execute solely "talk-only" operations over the GPIB, such as driving a printer. Setting the oscilloscope to "RS-232" enables the instrument to be controlled via the RS-232-C port (*see next chapter*).

If the oscilloscope is addressed to talk, it will remain thus configured until receiving a universal untalk command (UNT), its own listen address (MLA), or another instrument's talk address.

Similarly, if the oscilloscope is addressed 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 bus system consists of 16 signal lines and eight ground or shield lines. The signal lines are divided into three groups:

1. **Data Lines:** These eight lines, usually called DI01 through to DI08, carry both program and interface messages. Most of the messages use the 7-bit ASCII code, in which case DI08 is unused.

2. **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.

3. **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 instrument 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 instrument 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.

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

***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.*

them in the remote state. Such interface messages are executed with the interface management lines of the GPIB, usually with ATN true.

All of these messages (except GET) are executed immediately upon reception and not, with normal commands, in chronological order.

The command list in *System Commands* does not contain a command for clearing the input or output buffers, nor for setting the instrument to the remote state. 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 which go beyond mere reconfiguration of the bus and have an effect on the operation of the instrument.

Device CLear

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

Group Execute Trigger

The Group Execute Trigger message (GET) causes the oscilloscope to arm the trigger system. It 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 and configures the instrument 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 instrument is set to the remote mode. Whenever the controller returns the REN line to false, all instruments on the bus return to GO TO LOCAL. Individual instruments can be returned to LOCAL with the Go To Local message (*see below*).

Local front-panel control may be regained by pressing the GO TO LOCAL menu button, unless the instrument has been placed in Local LOckout (LLO) mode.

Local LOckout

The Local LOckout command (LLO) causes the GO TO LOCAL menu to disappear. The LLO command can be sent in local or remote mode but only becomes effective once the instrument has been set to the remote mode.

Go To Local

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

When Local Lockout is activated the scope can only be returned to its local state by the controller returning the LLO to false. And whenever the instrument returns to the remote state the local lockout mode will immediately become effective again.

InterFace Clear

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

Programming GPIB Transfers

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 to this.

If using another type of computer or GPIB interface, refer to the interface manual for installation procedures and subroutine calls.

---

†*National Instruments Corporation, 12109 Technology Boulevard, Austin, Texas 78727, USA.*

Configuring the GPIB Driver Hardware

Check that the GPIB interface is properly installed in the computer. If found that 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 these program examples, default positions are assumed.

Connect the oscilloscope to the computer with a GPIB interface cable. Set the GPIB address to the required value. The program examples assume a setting of '4'.

Configuring the GPIB Driver Software

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 copies the GPIB handler GPIB.COM into the boot directory, modifies the DOS system configuration file CONFIG.SYS to declare the presence of the GPIB handler, creates a sub-directory GPIB-PC, and installs 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 via functions entered at the keyboard. Use of this program is highly recommended to anyone unfamiliar with GPIB programming or the oscilloscope'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.*

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'.

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.***

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 the oscilloscope 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.

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 which 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.***

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 GPIB address 4.

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

Lines 140 and 150 reads the response of the instrument and places it into the character string RD$.

Line 170 displays the response on the terminal.

When running this sample program, the oscilloscope 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 the oscilloscope to local mode if the GPIB handler was modified to inhibit Local LOckout (LLO).

Here is a slightly modified version of the sample program which 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 instrument was set to a value other then 4. Line 180 resets the instrument to local with a call to the GPIB routine IBLOC.

Additional Driver Calls

IBLOC is used to execute the IEEE 488.1 standard message Go To Local (GTL), i.e. it returns the instrument 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**, respectively, 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 function calls IBRD and IBWRT are far less limited in terms of 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 instrument 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 the oscilloscope to arm the trigger system.

National Instruments supply 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.

Service Requests

When an oscilloscope is used in a remote application, events often occur asynchronously — at times that are unpredictable for the host computer. The most common example of this is the wait of a trigger after the arming of the instrument: 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 bytes refer to Chapter 5.*

A potentially more efficient way of detecting events occurring in the instrument 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 instrument. 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 the user 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. The SRQ is enabled by setting the Service Request Enable register with the command "*SRE" and specifying which event should generate an SRQ. The oscilloscope 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, the controller may be required to identify which instrument caused the interrupt by serial polling the various devices.

***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 may be disabled by clearing the SRE register ("*SRE 0").*

Example 1

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 the 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:

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

```
CMD$="INE 5;*SRE 1"
```

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

Example 2

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

Instrument Polls

State transitions occurring within the instrument can be remotely monitored by polling selected internal status registers (*see, too, Chapter 5*). Four basic polling methods can be used to detect the occurrence of a given event. These are continuous, serial, parallel and *IST.

By far the simplest of these is continuous polling. The other three are appropriate only when interrupt service routines (servicing the SRQ line) are supported, or multiple devices on GPIB require constant monitoring. Emphasizing the differences between these methods, which are described below, the same example — determining whether a new acquisition has taken place — is presented in respect of each.

Continuous Poll

Here, a status register is continuously monitored until a transition is observed. This is the most straightforward method for detecting state changes, but may be impracticable 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, obviating the need for an additional clearing action after a non-zero value has been detected. The command "CHDR OFF" instructs the instrument to omit any command headers when responding to a query, simplifying the decoding of the response. The instrument 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
```

Serial Poll

Serial polling takes place once the SRQ interrupt line has been asserted, and is only advantageous when there are several instruments involved. The controller finds which device of a number has generated the interrupt by inspecting the SRQ bit in the STB register of each instrument. 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 instrument generated the interrupt. It will only exit if there was a timeout or if the instrument "SCOPE%" generated SRQ. The additional function call IBRSP fetches the value of the status byte which may be further interpreted. For this to function 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 instruments 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 instrument is reset. If these bits are not cleared, they cannot generate another interrupt.*

Parallel Poll

Like serial polling, this is only advantageous when there are several instruments. In parallel polling, the controller simultaneously reads the Individual STatus bit (IST) of all the instruments to determine which needs service. Because this method allows up to eight different instruments to be polled at the same time, it is the fastest way to identify state changes in instruments with this capability.

When a parallel poll is initiated, each instrument 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 DI02 takes place.

Stage 1: Enable the INE and PRE registers, configure the controller for parallel poll and instruct the oscilloscope to respond on data line 2 (DI02)

```
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: Parallel poll the instrument until DI02 is set

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

Stage 3: Disable parallel polling (hex 15) and clear the parallel poll register

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

***Note:** In the example above, board-level GPIB function calls are used. It is assumed that the controller (board) and scope (device) are respectively located at addresses 0 and 4. The listener and talker addresses for the controller and oscilloscope are*

Logic Device	Listener Address	Talker Address
External Controller	32 ASCII<space>)	64 (ASCII @)
Oscilloscope	32+4=36 (ASCII $)	64+4=68 (ASCII D)

***IST Poll**

The state of the Individual STatus bit (IST) returned in parallel polling can also be read by sending the "*IST?" query. To enable this poll mode, the oscilloscope must be initialized as for parallel polling by writing into the PRE register. Since *IST polling emulates parallel polling, this method is applicable in all instances where 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 instrument's response, simplifying the interpretation. The status of the IST bit is then continuously monitored until set by the instrument:

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. Since 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.***

```
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
```

Driving Hardcopy Devices on the GPIB

The oscilloscope can be interfaced with a wide range of hardcopy devices for copying the screen contents to them. The devices supported are listed with the command "HARDCOPY_SETUP" (*see System Commands*).

With a hardcopy device connected to the GPIB, one of two basic configurations can be used. When only the oscilloscope and a hardcopy device such as a printer are connected to the GPIB, the oscilloscope must be configured as talker-only, and the hardcopy device as listener-only, to ensure proper data transfer (*see below*).

However, when an external controller is connected to the GPIB, it must be used to supervise the data transfers. A variety of schemes can then be used to transfer the oscilloscope screen contents (*see next page*).

DSO & Printer Only

The oscilloscope is configured as talker-only using the "GPIB & RS232" menus selected using the instrument's front-panel controls (*see the sections "GPIB/RS232 Setup" and "Hardcopy Setup" in the UTILITIES Chapter of the Operator's Manual*). The hardcopy device manufacturer usually specifies an address that forces the instrument into listening mode, and this as well as the other necessary settings can be selected using the same menus.

To configure the oscilloscope as talker-only:

1. **Press  to select "Remote Control from RS-232" from the ("UTILITIES" menu).**
2. **Select "GPIB" from the "output to" menu in the "HARDCOPY" group.**
3. **Put the hardcopy device in listener-only mode.**
4. **Press** 

Printing by GPIB Controller

The following schemes can be used for driving hardcopy devices by remote control using the GPIB.

Data read by controller The controller reads the data into internal memory, then sends them to the printer. This alternative can be done 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$)
CALL IBWRTF(PRINTER%,FILE$)
```

DSO sends data to both The oscilloscope 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 later be deleted.

Stage 1: Controller talker, oscilloscope listener. Issue the screen dump command

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

Stage 2: Oscilloscope talker, controller and printer listeners. Print data while storing data in scratch file SCRATCH.DAT

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

DSO talks directly to printer The controller goes into a standby state. The oscilloscope becomes a talker and sends data directly to the printer. The controller goes into stand-by and resumes GPIB operations once the data have been printed, i.e. when an EOI is detected:

Stage 1: Controller talker, oscilloscope listener. Issue the screen dump command

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

Stage 2: Oscilloscope talker, printer listener. Put controller in stand-by

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

Note:

- ***In Schemes 2 and 3, board-level GPIB function calls are used. It is assumed that the controller (board), the scope and the printer are respectively located at addresses 0, 4 and 5. The listener and talker addresses for the controller, oscilloscope and printer are***

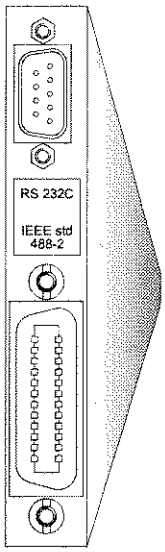
Logic Device	Listener Address	Talker Address
Controller	32 (ASCII<space>)	64 (ASCII @)
Oscilloscope	32+4=36 (ASCII $)	64+4=68 (ASCII D)
Printer	32+5=37 (ASCII %)	64+5=69 (ASCII E)

- ***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, PC talker, DSO listener

CMD2$ = "?_ D" Unlisten, Untalk, PC listener, DSO talker

Using the RS-232-C Port



This chapter describes remote control of the oscilloscope when the RS-232-C port is used. It concludes with definitions of the special RS-232-C commands used for configuring the oscilloscope, or for simulating GPIB 488.1 messages such as those for placing the oscilloscope in either remote or local mode.

The port supports all commands described in the *System Commands* section of this manual. However, waveform transfer is only possible in HEX mode, while the default value for COMM_FORMAT is set appropriately. The syntax of the response to WF? is identical to that for GPIB.

Notation

In this chapter, characters that cannot be printed in ASCII are 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 means that the control key and the U key are pressed simultaneously.

Connector Pin Assignments

See the section "GPIB/RS232 Setup" in the UTILITIES Chapter of the oscilloscope Operator's Manual.

RS-232-C Configuration

The RS-232-C port is full-duplex configured. This means that both sides, the external controller *and* the oscilloscope, can send and receive messages at the same time.

Nevertheless, when the oscilloscope receives a new command, it stops outputting.

Transmission of long messages to the oscilloscope should be done while it is in a triggered mode, and *not* while an acquisition is in progress. This is especially important when sending waveforms or front-panel setups into the oscilloscope.

The behavior of the RS-232-C port may be set according to the operator's needs. In addition to the basic setup on the front-panel

menu, there are "immediate commands", as well as a special command "COMM_RS232". 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.

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

Echo

The serial port may echo the received characters. Echo is useful if the oscilloscope is attached to a terminal. Echoing can be turned on or off by sending the two-character sequence <ESC>] or <ESC>[respectively. Echoing is on by default.

***Note:** 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 be sent. These handshake 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 RTS handshake.

The flow of characters coming from the oscilloscope may be controlled either by a signal level on the CTS line or by the <XON>/<XOFF> pair of characters.

Editing Features

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

<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.

On input to the oscilloscope, the Program Message Terminator is a character that could be selected. A good choice is a character never used for anything else. Such a character is chosen 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 is chosen by the user. This string must not be empty. The default Response Message Terminator is "\n\r", which means <LF><CR>.

Examples

```
COMM_RS232 EI,3
```

This command informs the oscilloscope that each message it receives will be terminated with the ASCII character <ETX> which corresponds to 3 in decimal.

```
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 these settings, a host command will look like:

```
TDIV?<ETX>
```

The oscilloscope responds:

```
TDIV 1.S
END
```

Note: ***Having sent a COMM_RS232 command, the host must wait for the oscilloscope to change its behavior before sending a command in the new mode. A sure way to do this is to include a query on the line that contains the COMM_RS232 command and wait until the response is received. For example:*** `COMM_RS232 EI,3;*STB?`

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
a <CR> followed by one <LF>

and 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:

```
PNSU#9000001496
AAAA5555000655AA4030005800190000000000001
00000000000000000000000000C1B0100580000
0000000000000000000000000000000000000000
...
```

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> and/or <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.

Simulating GPIB Commands with RS-232-C

These special RS-232-C commands can be used for simulating GPIB 488.1 messages.

<ESC>C or <ESC>c Device Clear Command

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 Set to Remote Command (REN)

Places the oscilloscope into the 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 Set to Local Command

Places the oscilloscope into local mode. The command clears local lockout (*see below*). It has the same function as GPIB setting the REN line to false.

<ESC>F or <ESC>f Set to Local Lockout Command

Disables the front-panel "LOCAL" button, either immediately, if the oscilloscope is already in remote mode, or later, when the oscilloscope is next set to remote.

This disabling of the menu "LOCAL" button is called "Local Lockout" and can only be cancelled with the <ESC>L command. <ESC>F has the same meaning as the GPIB LLO interface message.

<ESC>T or <ESC>t Trigger Command (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, as well as having the same meaning as the GPIB GET interface message.

Understanding Waveforms

This chapter covers the reading and writing of waveforms in remote control, and attempts to explain their structure and content.

Basic Structure

Waveforms can be divided into two basic entities. One is the basic data array: the raw data values from the oscilloscope's ADCs (Analog–to–Digital Converters) in the acquisition. The other is the accompanying descriptive information, such as vertical and horizontal scale, and time of day, necessary for a full understanding of the information contained in the waveform.

This information can be accessed by remote control using the INSPECT? Query, which interprets it in an easily understood ASCII text form. It can be more rapidly transferred using the WAVEFORM? query, or written back into the instrument with the WAVEFORM command.

The oscilloscope itself contains a data structure, or *template*, which provides a detailed description of how the waveform's information is organized.

Waveforms can also be stored in pre-formatted ASCII output*, for popular spreadsheets and math processing packages, using the STORE and STORE_SETUP commands.

Waveform Template

This gives a detailed description of the form and content of the logical data blocks of a waveform, and is provided as a reference. Although a sample template is given elsewhere in this manual (*see Appendix B*), it is suggested that the TEMPLATE? query and the actual instrument template be used. The template may change as the instrument's firmware is enhanced, and it will help provide backward compatibility for the interpretation of waveforms.

Logical Data Blocks

A waveform normally contains a waveform descriptor block and a data array block. However, in more complicated cases, one or more other blocks will be present.

---

* *See Appendix E of the* Operator's Manual *for these formats.*

- Waveform Descriptor block (WAVEDESC): This block includes all the information necessary to reconstitute the display of the waveform from the data. This includes:
  - hardware settings at the time of acquisition
  - the exact time of the event
  - the kinds of processing that have been performed
  - the name and serial number of the instrument
  - the encoding format used for the data blocks
  - miscellaneous constants.
- Optional User-provided Text block (USERTEXT): The WFTX command can be used to put a title or description of a waveform into this block. The WFTX? query command gives an alternative way to read it. This text block can hold up to 160 characters. They can be displayed in the TEXT + TIMES status menu as four lines of 40 characters.
- Sequence Acquisition Times block (TRIGTIME): This block 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 block is needed 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 second data array is needed to hold the results of processing functions such as the Extrema (WP01 option) or Complex FFT (WP02 option). In such cases, the data arrays contain:

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

Note: *The Template also describes an array named DUAL. This is simply a way to allow the INSPECT? command to examine the two data arrays together.*

Using the INSPECT? Query

The query INSPECT? is a simple way to examine the contents of a waveform in remote control.

Usable on both the data and descriptive parts, its most basic form is:

```
INSPECT? "name"
```

where the template gives the name of a descriptor item or data block. The answer is returned as a single string, but may span many lines. Some typical dialogue:

question
response
question
response

```
C1:INSPECT? "VERTICAL_OFFSET"
C1:INSP "VERTICAL_OFFSET: 1.5625e-03"
C1:INSPECT? "TRIGGER_TIME"
C1:INSP "TRIGGER_TIME: Date = FEB 17, 1994,
Time =  4: 4:29.5580"
```

INSPECT? can also be used to provide a readable translation of the full waveform descriptor block with:

```
INSPECT? "WAVEDESC"
```

The template dump will give details of the interpretation of each of the parameters. INSPECT? is also used to examine the measured data values of a waveform using:

```
INSPECT? "SIMPLE"
```

For example, for an acquisition with 52 points:

```
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
 "
```

These numbers are the fully converted measurements in volts. Of course, when the data block contains thousands of items the string will contain a great many lines.

Depending on the application, the data may be preferred in its raw form as either a BYTE (8 bits) or a WORD (16 bits) for each data value. In this case the relations given below must be used in association with WAVEFORM? to interpret the measurement. It might then say:

```
INSPECT? "SIMPLE",BYTE
```

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.

Finally...

INSPECT? is useful, but it is also a rather verbose way to send information. As a query form only, INSPECT? cannot be used to send a waveform back into the oscilloscope. Users who require this capability or speed or both should instead use the WAVEFORM query or commands. It is possible to examine just a part of the waveform or a sparsed form of it, using the WAVEFORM_SETUP command covered later in this chapter.

BASIC users might find it convenient, too, to combine the capabilities of the inspect facility with the waveform query command in order to construct files containing a human and BASIC readable version of the waveform descriptor together with the full waveform in a format suitable for retransmission to the instrument. This can be done for a waveform in a memory location by sending the command

```
MC:INSPECT? "WAVEDESC";WAVEFORM?
```

and putting the response directly into a disk file.

WAVEFORM?, Related Commands, and Blocks

Using the WAVEFORM? query is an effective way to transfer waveform data using the block formats defined in the IEEE-488.2 standard. Responses can then be downloaded back into the instrument using the WAVEFORM command.

All of a waveform's logical blocks can be read with the single query:

```
C1:WAVEFORM?
```

This is the preferred form for most applications due to its completeness. Time and space are the advantages when reading many waveforms with the same acquisition conditions, or when the interest is only in large amounts of raw integer data.

And any single block can be chosen for reading with a query such as:

```
C1:WAVEFORM? DAT1
```

The description In the *System Commands* section provides the various block names.

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.

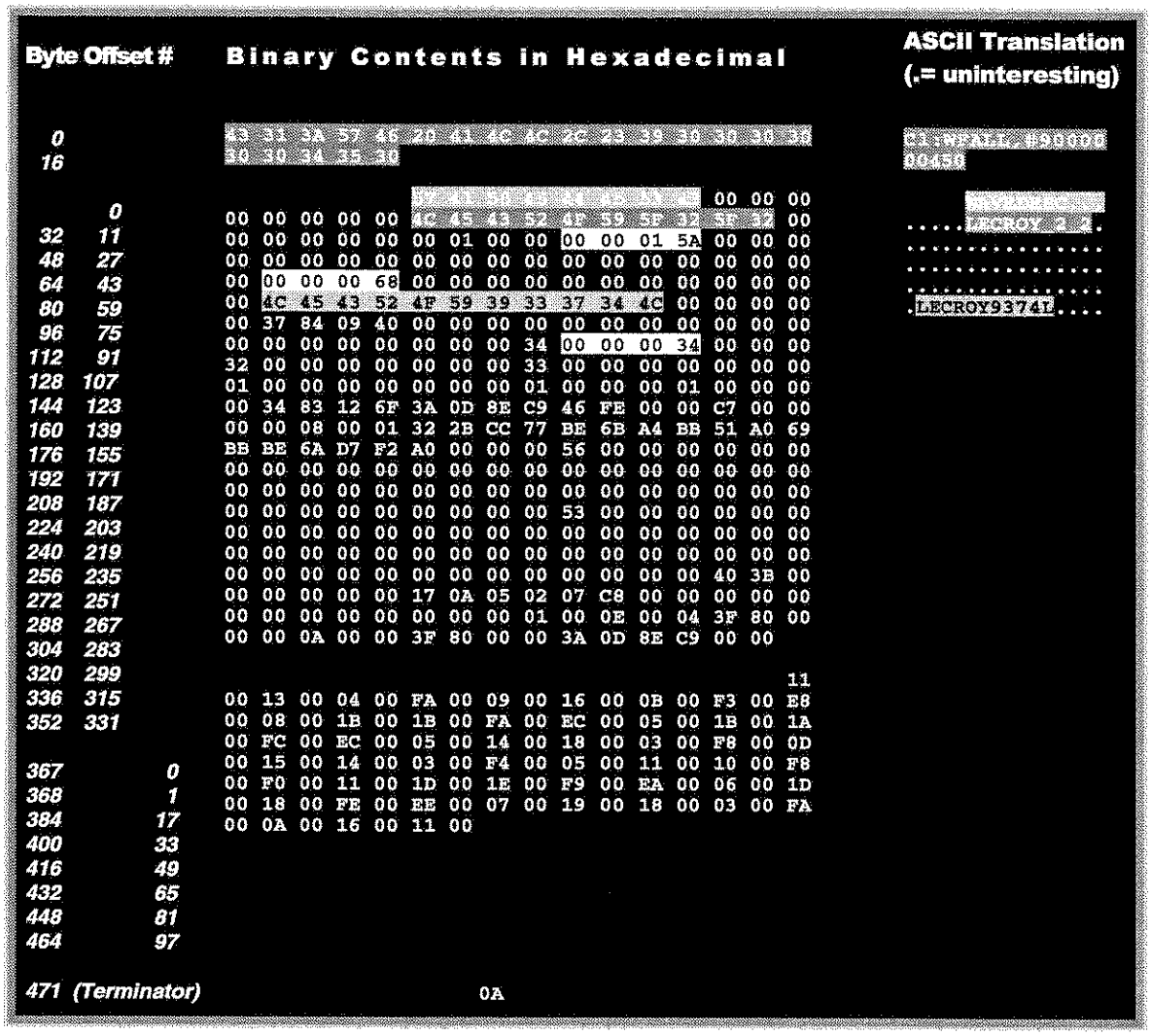
Interpreting the Waveform Descriptor

The binary response to a query of the form:

```
C1:WAVEFORM? or C1:WAVEFORM? ALL
```

can be placed in a disk file and then dumped to show the following hexadecimal and ASCII form†:

† *Done using the GPIB bus with default settings.*



Here, in order to illustrate the contents of the logical blocks, the relevant parts (see explanations next page) have been separated. In addition, to facilitate counting, 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, highlighted in the same fashion.

On the preceding 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 number 21. The very first byte is Byte #0, as it is for the first byte in each block (at the head of each of the three Byte Offset columns illustrated).

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 (or at Byte #37, counting from the very start and referring to the numbers in the first Byte Offset column), we find the beginning of the next string: the TEMPLATE_NAME with the value LECROY_2_2.

Several other parameters follow. The INSTRUMENT_NAME, 76 bytes from the descriptor start (Byte #97), is easily recognizable. On the preceding line, at 38 bytes after the descriptor (Byte #59), a four-byte-long integer gives the length of the descriptor:

WAVE_DESCRIPTOR = 00 00 01 5A (hex) = 346.

At 60 bytes from the descriptor start (or 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>.

---

‡ *The waveform descriptor can be deciphered with the aid of the waveform template (see Appendix B).*

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). These 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.

Interpreting Vertical Data

Now that we know how to decipher the data it would be useful 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 respective of the descriptor start 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

In the case of the data shown above we find:

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

and 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, a description is to be found 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.

Calculating 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 being examined. We will treat separately the single sweep, the sequence, and the interleaved (RIS) 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 × 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

which gives:

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

Sequence Waveforms

Since sequence waveforms are really many independent acquisitions, each segment will have its own horizontal offset. These can be found in the TRIGTIME array. For the n'th segment

x[i,n] = HORIZ_INTERVAL × i + 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.

Interleaved (RIS) Waveforms

These 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] = HORIZ_INTERVAL × j + 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:

RIS_OFFSET[i] ≅ PIXEL_OFFSET + (i – 0.5) × 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.
...

WAVEFORM Commands

Waveforms that have been read in their entirety with the WAVEFORM? query can be sent back into the instrument using WAVEFORM and other, related commands. Since the descriptor contains all of the necessary information, care need not be taken with any of the communication format parameters. The instrument can learn all it needs to know from the waveform.

When synthesizing waveforms for display or comparison, in order

Note: Waveforms can only be sent back to memory traces (M1, M2, M3, M4). This means possibly removing or changing the prefix (C1 or CHANNEL_1) in the response to the WF? query. See the System Commands for examples.

to ensure that the descriptor is coherent, read out a waveform of

the appropriate size and then replace the data with the desired values.

There are many ways to use WAVEFORM and related commands to simplify or speed up work. Among them:

- **Partial Waveform Readout:** The WAVEFORM_SETUP command allows specification of a short part of a waveform for readout, as well as selection of a sparsing factor for reading only every n'th data point.
- **Byte Swapping:** The COMM_ORDER command allows the swapping of the two bytes of data presented in 16-bit word format (can be in the descriptor or in the the data/time arrays), when sending the data over the GPIB or RS-232-C remote-control ports. This allows easier data interpretation, depending on the computer system used:
  - Intel-based computers — the data should be sent with the LSB first, and the command should be CORD LO.
  - Motorola-based computers — the data should be sent with the MSB first (CORD HI). This is the default at power-up.

***Note:** Data written to the scope's Hard Disk Drive, Floppy Drive, or to Memory Card,will* **always** *remain in LSB first format, as this is the default DOS format. The CORD command cannot be applied here, as it is only applicable for data sent over the GPIB or RS-232-C ports*

- **Data Length, Block Format, and Encoding**: The COMM_FORMAT command gives control over these parameters. If the extra precision of the lower order byte of the standard data value is not needed, the BYTE option allows a saving of a factor of two on the amount of data to be transmitted or stored. If the computer being used is unable to read binary data, the HEX option allows a response form where the value of each byte is given by a pair of hexadecimal digits.

- **Data-Only Transfers**: The COMM_HEADER OFF mode enables a response to WF? DAT1 with the data only (the C1:WF DAT1 will disappear).

  If COMM_FORMAT OFF,BYTE,BIN has also been specified, the response will be mere data bytes (the #90000nnnnn will disappear).

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

High-Speed Waveform Transfer

Several important factors need to taken into account for achieving maximum continuous-data-transfer rates from oscilloscope to controller.

The single most important of these is the limiting of work done in the computer. This effectively means avoiding writing data to disk wherever possible, as well as minimizing operations such as per-data-point computations and reducing the number of calls to the IO system. Ways of doing this include:

- Reducing 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.
- Attempting 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 the oscilloscope will be able to acquire events will be considerably increased if it is obliged to wait for triggers (livetime).
- Minimizing the number of waveform transfers by using Sequence Mode to accumulate many triggers for each transfer. This is preferable to using the WAVEFORM_SETUP command to reduce the number of data points to be transferred. It also significantly reduces oscilloscope transfer overhead.

Example

The desirable type of command is:

```
ARM; WAIT;C1:WF?
```

to wait for the event, transfer the data, and then start a new acquisition.

This line can be "looped" in the program as soon as it has finished reading the waveform.

Using Status Registers

A wide range of status registers allows the oscilloscope's internal processing status to be determined quickly at any time. These registers and the instrument's status reporting system are designed to comply with IEEE 488.2 recommendations. *Following an overview, starting this page, each of the registers and their roles are described.*

Related functions are grouped together in common status registers. Some, such as the Status Byte Register (STB) or the Standard Event Status Register (ESR), are required by the IEEE 488.2 Standard. Other registers are device-specific, and include 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 <*>.

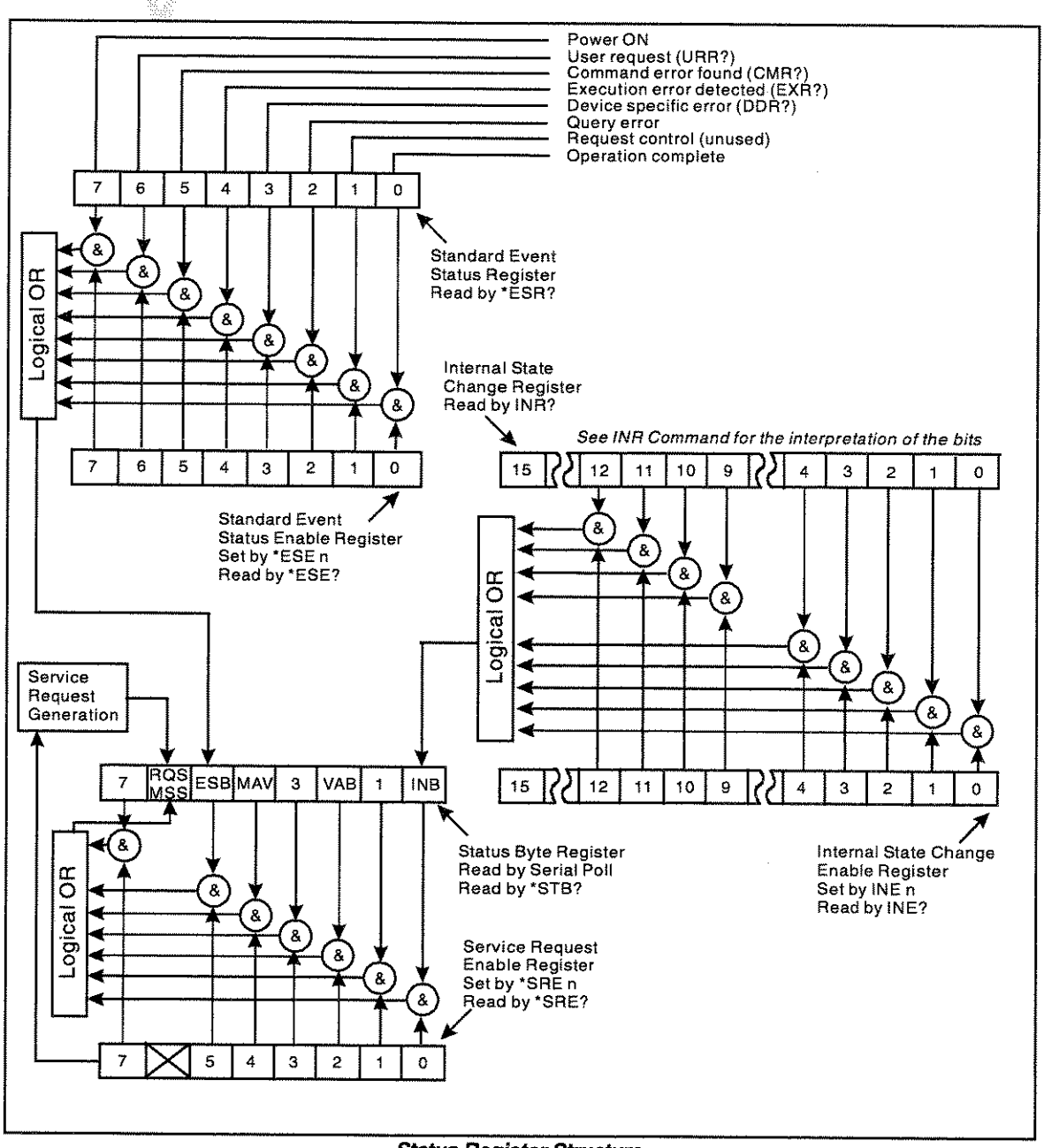
Overview

The Standard Event Status Bit (ESB) and the Internal Status Change Bit (INB) in the Status Byte Register are summary bits of the Standard Event Status Register (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" is received, the timebase is set to 2 ms/div along with the VAB bit).

The Master Summary Status bit (MSS) indicates a request for service from the instrument. The MSS bit can only be set if one or more of the other bits of STB are enabled 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 Standard Event Status Register (ESR) primarily summarizes errors, whereas the Internal State Change Register (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?".



Status Register Structure

The register structure contains one additional register, not shown in the figure on the previous page. This is the Parallel Poll Enable Register (PRE), which behaves exactly like the Service Request Enable Register (SRE), but sets the "ist" bit (also not shown in the figure), used in the Parallel Poll. The "ist" bit can also be read with the "*IST?" query.

Example

If an erroneous remote command — "TRIG_MAKE SINGLE", for example — is transmitted to the instrument, it rejects the command and sets the Command Error Register (CMR) to the value 1 (unrecognized command/query header). The non-zero value of CMR is reported to Bit 5 of the Standard Event Status Register (ESR), which is then set.

Nothing further occurs unless the corresponding Bit 5 of the Standard Event Status Enable Register (ESE) is set (with the command "*ESE 32"), enabling Bit 5 of ESR to be set for reporting to the summary bit ESB of the Status Byte Register (STB).

If setting of the ESB summary bit in STB is enabled, again nothing occurs unless further reporting is enabled by setting the corresponding bit in the Service Request Enable Register (with the command "*SRE 32"). In this case, the generation of a non-zero value of CMR ripples through to the Master Summary Status bit (MSS), generating a Service Request (SRQ).

The value of CMR can be read 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 several types of potential errors must be surveyed, it is usually far more efficient to enable propagation of the errors of interest into the STB with the enable registers ESE and INE.

Summary

A command error (CMR) sets Bit 5 of ESR if:

- Bit 5 of ESE is set, ESB of STB is also set, or
- Bit 5 of SRE is set, MSS/RQS of STB is also set and a Service Request is generated.

Status Byte Register (STB)

The Status Byte Register (STB) is the instrument's central reporting structure. The STB is composed of eight single-bit summary messages (of which three are unused), which reflect the current status of the associated data structures implemented in the instrument:

- **Bit 0** is the summary bit INB of the Internal State Change Register. It is set if any of the bits of the INR are set, provided they are enabled by the corresponding bit of the INE register.
- **Bit 2** is the Value Adapted bit, indicating that a parameter value was adapted during a previous command interpretation.
- **Bit 4** is the Message Available (MAV) bit, indicating that the interface output queue is not empty.
- **Bit 5** is the summary bit ESB of the Standard Event Status Register. 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 Master Summary Status bit (MSS) or the Request for Service bit (RQS), owing to the STB being able to be read in two different ways. The command "*STB?" reads and clears the STB in the query mode, in which case Bit 6 is the MSS bit, and indicates whether the instrument has any reason for requesting service.

The Status Byte Register can be read using the query "*STB?". The response represents the binary weighted sum of the register bits. The register is cleared by "*STB?", "ALST?", "*CLS", or after the instrument has been powered up.

Another way of reading the STB is using the serial poll (*see "Instrument Polls", 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 only clears the RQS bit. Therefore, the MSS bit of the STB (and any other bits which caused MSS to be set) will stay set after a serial poll. These bits must be reset.

Standard Event Status Register (ESR)

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

The ESR is read using the query "*ESR?". The response is the binary weighted sum of the register bits. The register is cleared with an "*ESR?" or "ALST?" query, a "*CLS" command or after power-on.

Example

The response message "*ESR 160" indicates 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 on the same page as the ESR command description for the conditions corresponding to the bits set.*

The "Power ON" bit appears only on the first "*ESR?" query after power-on because the query clears the register. This type of command error can be determined by reading the Command Error Status Register with the query "CMR?". Note that it is not necessary to read (nor 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)

The Standard Event Status Enable Register (ESE) allows one or more events in the Standard Event Status Register to be reported to the ESB summary bit in the STB.

The ESE is modified with the command "*ESE" and cleared with the command "*ESE 0", or after power-on. It is read with the query "*ESE?".

Example

"*ESE 4" sets bit 2 (binary 4) of the ESE Register, enabling query errors to be reported.

Service Request Enable Register (SRE)

The Service Request Enable Register (SRE) specifies which summary bit(s) in the Status Byte Register will bring about a service request. The SRE consists of eight bits. Setting a bit in this register allows the summary bit located at the same bit position in the Status Byte Register 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 the query "*SRE?".

The SRE is modified with the command "*SRE" and cleared with the command "*SRE 0", or after power-on. It may be read with the query "*SRE?".

Parallel Poll Enable Register (PRE)

The Parallel Poll Enable Register (PRE) specifies which summary bit(s) in the Status Byte Register will set the "ist" individual local message. This register is quite similar to the Service Request Enable Register (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).

The PRE is modified with the command "*PRE" and cleared with the command "*PRE 0", or after power-on. It is read with the query "*PRE?". (*See Chapter 2 "Instrument Polls".*)

Example

"*PRE 5" sets bits 2 and 0 (decimal 4 and 1) of the Parallel Poll Enable Register.

Internal State Change Status Register (INR)

The Internal State Change Status Register (INR) reports the completion of a number of internal operations (*the events tracked by this 16-bit-wide register are listed with the "INR?" query in the System Commands section*).

The INR is read using the query "INR?". The response is the binary-weighted sum of the register bits. The register is cleared with an "INR?" or "ALST?" query, a "*CLS" command, or after power-on.

Internal State Change Enable Register (INE)

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

The INE is modified with the command "INE" and cleared with the command "INE 0", or after power-on. It is read with the query "INE?".

Command Error Status Register (CMR)

The Command Error Status register contains the code of the last command error detected by the instrument. Command error codes are listed with the command "CMR?".

The Command Error Status Register may be read using the query "CMR?". The response is the error code. The register is cleared with a "CMR?" or "ALST?" query, a "*CLS" command, or after power-on.

Device Dependent Error Status Register (DDR)

The Device Dependent Error Status Register (DDR) indicates the type of hardware errors affecting the instrument. Individual bits in this register report specific hardware failures. They are listed with the command "DDR?".

The DDR is read using the "DDR?" query. The response is the binary weighted sum of the error bits. The register is cleared with a "DDR?" or "ALST?" query, a "*CLS" command, or after power-on.

Execution Error Status Register (EXR)

The Execution Error Status Register (EXR) contains the code of the last execution error detected by the instrument. Execution error codes are listed with the command "EXR?".

The EXR is read using the "EXR?" query. The response is the error code. The register is cleared with an "EXR?" or "ALST?" query, a "*CLS" command, or after power-on.

User Request Status Register (URR)

The User Request Status Register (URR) contains the identification code of the last menu button pressed. The codes are listed with the command "URR?".

The URR is read using the query "URR?". The response is the decimal code associated with the selected menu button. The register is cleared with a "URR?" or "ALST?" query, a "*CLS" command, or after power-on.

About These Commands & Queries

This section lists and describes the remote control commands and queries recognized by the instrument. All commands and queries can be executed in either local or remote state. Where not included here, those for special options can be found in the options' dedicated Operator's Manuals.

The description for each command or query, with syntax and other information, begins on a new page. The name (header) is given in both long and short form at the top of the page, and the subject is indicated as a command or query or both. Queries perform actions such as obtaining information, and are recognized by the question mark (?) following the header.

How They are Listed

The descriptions are listed in alphabetical order according to their *long* form. Thus the description of ATTENUATION, whose short form is ATTN, is listed before that of AUTO SETUP, whose short form is ASET. The two special indexes at the beginning of this section are designed as reference aids for quickly finding commands and queries. One lists the commands and queries in alphabetical order according to short form, while the other groups them according to subsystem or category.

How They are Described

In the descriptions themselves, a brief explanation of the function performed is given. This is followed by a presentation of the formal syntax, with the header given in `Upper-and-Lower-Case` characters and the short form derived from it in `ALL UPPER-CASE` characters. 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 shows calls to the related interface subroutines in BASIC. The device name of the instrument is defined as SCOPE%.

When Can They be Used?

The commands and queries listed here can be used with all LeCroy digital instruments. However, the raised hand symbol ☝ indicates a note on availability for a particular model or option.

Commands & Queries

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 may be used in a command instead of the placeholder.
{ }	Braces enclose a list of choices, one of which one must be made.
[ ]	Square brackets enclose optional items.
...	An ellipsis indicates that the items both to its left and right may be repeated a number of times.

As an example, consider the syntax notation for the command to set the vertical input sensitivity:

<channel> : VOLT_DIV <v_gain>

<channel> : = {C1, C2}

<v_gain> : = 5.0 mV to 2.5 V

The first line shows the formal appearance of the command, with <channel> denoting the placeholder for the header path and <v_gain> the placeholder for the data parameter specifying the desired vertical gain value. The second line indicates that either C1 or C2 must be chosen for the header path. And the third explains that the actual vertical gain can be set to any value between 5 mV and 2.5 V.

Refer to Chapter 1 for an overview of the command functions and notation used.

Commands & Queries Tabled by Short Form

Page	Short	Long Form	Subsystem	What the Command/Query Does
15	ACAL	AUTO_CALIBRATE	MISCELLANEOUS	Enables or disables automatic calibration.
12	ALST?	ALL_STATUS?	STATUS	Reads and clears the contents of all status registers.
13	ARM	ARM_ACQUISITION	ACQUISITION	Changes acquisition state from "stopped" to "single".
16	ASCR	AUTO_SCROLL	DISPLAY	Controls the Auto Scroll viewing feature.
17	ASET	AUTO_SETUP	ACQUISITION	Adjusts vertical, timebase and trigger parameters.
14	ATTN	ATTENUATION	ACQUISITION	Selects the vertical attenuation factor of the probe.
20	BUZZ	BUZZER	MISCELLANEOUS	Controls the built-in piezo-electric buzzer.
18	BWL	BANDWIDTH_LIMIT	ACQUISITION	Enables/disables the bandwidth-limiting low-pass filter.
21	*CAL?	*CAL?	MISCELLANEOUS	Performs complete internal calibration of the instrument.
34	CFMT	COMM_FORMAT	COMMUNICATION	Selects the format for sending waveform data.
36	CHDR	COMM_HEADER	COMMUNICATION	Controls formatting of query responses.
37	CHLP	COMM_HELP	COMMUNICATION	Controls operational level of the RC Assistant.
38	CHL	COMM_HELP_LOG	COMMUNICATION	Returns the contents of the RC Assistant log.
24	CHST	CALL_HOST	DISPLAY	Allows manual generation of a service request (SRQ).
25	CLM	CLEAR_MEMORY	FUNCTION	Clears the specified memory.
27	*CLS	*CLS	STATUS	Clears all status data registers.
26	CLSW	CLEAR_SWEEPS	FUNCTION	Restarts the cumulative processing functions.
28	CMR?	CMR?	STATUS	Reads and clears the Command error Register (CMR).
30	COLR	COLOR	DISPLAY	Selects color of individual on-screen objects
33	COMB	COMBINE_CHANNELS	ACQUISITION	Controls the channel interleaving function.
39	CORD	COMM_ORDER	COMMUNICATION	Controls the byte order of waveform data transfers.
40	CORS	COMM_RS232	COMMUNICATION	Sets remote control parameters of the RS-232-C port.
22	COUT	CAL_OUTPUT	MISCELLANEOUS	Sets signal type put out at the CAL BNC connector.
43	CPL	COUPLING	ACQUISITION	Selects the specified input channel's coupling mode.
44	CRMS	CURSOR_MEASURE	CURSOR	Specifies the type of cursor/parameter measurement.
47	CRST?	CURSOR_SET?	CURSOR	Allows positioning of any one of eight cursors.
49	CRVA?	CURSOR_VALUE?	CURSOR	Returns trace values measured by specified cursors.
32	CSCH	COLOR_SCHEME	DISPLAY	Selects the display color scheme.
51	DATE	DATE	MISCELLANEOUS	Changes the date/time of the internal real-time clock.
52	DDR?	DDR?	STATUS	Reads, clears the Device Dependent Register (DDR).
53	DEF	DEFINE	FUNCTION	Specifies math expression for function evaluation.
59	DELF	DELETE_FILE	MASS STORAGE	Deletes files from mass storage.
60	DIR	DIRECTORY	MASS STORAGE	Creates and deletes file directories.
62	DISP	DISPLAY	DISPLAY	Controls the display screen.

Commands & Queries

Page	Short	Long Form	Subsystem	What the Command/Query Does
50	DPNT	DATA_POINTS	DISPLAY	Controls bold/single pixel display of sample points.
63	DTJN	DOT_JOIN	DISPLAY	Controls the interpolation lines between data points.
64	DZOM	DUAL_ZOOM	DISPLAY	Sets horizontal magnification and positioning.
65	*ESE	*ESE	STATUS	Sets the Standard Event Status Enable register(ESE).
66	*ESR?	*ESR?	STATUS	Reads, clears the Event Status Register (ESR).
69	EXR?	EXR?	STATUS	Reads, clears the EXecution error Register (EXR).
73	FCR	FIND_CTR_RANGE	FUNCTION	Automatically sets the center and width of a histogram.
74	FCRD	FORMAT_CARD	MISCELLANEOUS	Formats the memory card.
76	FFLP	FORMAT_FLOPPY	MISCELLANEOUS	Formats a floppy disk.
78	FHDD	FORMAT_HDD	MASS STORAGE	Formats the removable hard disk.
72	FLNM	FILENAME	MASS STORAGE	Changes default filenames.
81	FRST	FUNCTION_RESET	FUNCTION	Resets a waveform-processing function.
80	FSCR	FULL_SCREEN	DISPLAY	Selects magnified view format for the grid.
82	GBWL	GLOBAL_BWL	ACQUISITION	Enables/disables the Global Bandwidth Limit.
83	GRID	GRID	DISPLAY	Specifies single-, dual- or quad-mode grid display.
84	HCSU	HARDCOPY_SETUP	HARD COPY	Configures the hard-copy driver.
87	HCTR	HARDCOPY_TRANSMIT	HARD COPY	Sends string of ASCII characters to hard-copy unit.
88	HMAG	HOR_MAGNIFY	DISPLAY	Horizontally expands the selected expansion trace.
89	HPOS	HOR_POSITION	DISPLAY	Horizontally positions intensified zone's center.
91	*IDN?	*IDN?	MISCELLANEOUS	For identification purposes.
98	ILVD	INTERLEAVED	ACQUISITION	Enables/disables random interleaved sampling (RIS).
92	INE	INE	STATUS	Sets the Internal state change Enable register (INE).
93	INR?	INR?	STATUS	Reads, clears INternal state change Register (INR).
95	INSP?	INSPECT?	WAVEFORM TRANS.	Allows acquired waveform parts to be read.
97	INTS	INTENSITY	DISPLAY	Sets the grid or trace/text intensity level.
99	IST?	IST?	STATUS	Reads the current state of the IEEE 488.
100	KEY	KEY	DISPLAY	Displays a string in the menu field.
101	MGAT	MEASURE_GATE	DISPLAY	Controls highlighting of the measurement gate region.
103	MSG	MESSAGE	DISPLAY	Displays a string of characters in the message field.
102	MSIZ	MEMORY_SIZE	ACQUISITION	Selects max. memory length.
104	MZOM	MULTI_ZOOM	DISPLAY	Sets horizontal magnification and positioning.
105	OFST	OFFSET	ACQUISITION	Allows output channel vertical offset adjustment.
106	*OPC	*OPC	STATUS	Sets the OPC bit in the Event Status Register (ESR).
107	*OPT?	*OPT?	MISCELLANEOUS	Identifies oscilloscope options.
110	PACL	PARAMETER_CLR	CURSOR	Clears all current parameters in Custom, Pass/Fail.
111	PACU	PARAMETER_CUSTOM	CURSOR	Controls parameters with customizable qualifiers.
115	PADL	PARAMETER_DELETE	CURSOR	Deletes a specified parameter in Custom, Pass/Fail.

Page	Short	Long Form	Subsystem	What the Command/Query Does
116	PAST?	PARAMETER_STATISTICS?	CURSOR	Returns current statistics parameter values.
117	PAVA?	PARAMETER_VALUE?	CURSOR	Returns current parameter, mask test values.
127	PDET	PEAK_DETECT	ACQUISITION	Switches the peak detector ON and OFF.
128	PECS	PER_CURSOR_SET	CURSOR	Positions independent cursors.
130	PECV?	PER_CURSOR_VALUE?	CURSOR	Returns values measured by cursors.
133	PELT	PERSIST_LAST	DISPLAY	Shows the last trace drawn in a persistence data map.
131	PERS	PERSIST	DISPLAY	Enables or disables the persistence display mode.
132	PECL	PERSIST_COLOR	DISPLAY	Controls color rendering method of persistence traces.
134	PESA	PERSIST_SAT	DISPLAY	Sets the color saturation level in persistence.
135	PESU	PERSIST_SETUP	DISPLAY	Selects display persistence duration.
120	PFCO	PASS_FAIL_CONDITION	CURSOR	Adds a Pass/Fail test condition or custom parameter.
122	PFCT	PASS_FAIL_COUNTER	CURSOR	Resets the Pass/Fail acquisition counters.
123	PFDO	PASS_FAIL_DO	CURSOR	Defines desired outcome, actions after Pass/Fail test.
125	PFMS	PASS_FAIL_MASK	CURSOR	Generates tolerance mask on a trace and stores it.
126	PFST?	PASS_FAIL_STATUS?	CURSOR	Returns the Pass/Fail test for a given line number.
109	PNSU	PANEL_SETUP	SAVE/RECALL	Complements the *SAV/*RST commands.
136	*PRE	*PRE	STATUS	Sets the PaRallel poll Enable register (PRE).
137	PRCA?	PROBE_CAL?	PROBES	Performs auto-calibration of connected current probe.
138	PRDG?	PROBE_DEGAUSS?	PROBES	Degausses and calibrates the connected current probe.
139	PRNA	PROBE_NAME?	PROBES	Names the probe connected to the instrument.
140	*RCL	*RCL	SAVE/RECALL	Recalls one of five non-volatile panel setups.
142	RCPN	RECALL_PANEL	SAVE/RECALL	Recalls a front-panel setup from mass storage.
141	REC	RECALL	WAVEFORM TRANS.	Recalls a file from mass storage to internal memory.
143	*RST	*RST	SAVE/RECALL	The *RST command initiates a device reset.
145	*SAV	*SAV	SAVE/RECALL	Stores current state in non-volatile internal memory.
146	SCDP	SCREEN_DUMP	HARD COPY	Causes a screen dump to the hard-copy device.
144	SCLK	SAMPLE_CLOCK	ACQUISITION	Allows control of an external timebase.
147	SCSV	SCREEN_SAVE	DISPLAY	Controls the automatic screen saver.
148	SEL	SELECT	DISPLAY	Selects the specified trace for manual display control.
149	SEQ	SEQUENCE	ACQUISITION	Sets the conditions for the sequence mode acquisition.
151	*SRE	*SRE	STATUS	Sets the Service Request Enable register (SRE).
152	*STB?	*STB?	STATUS	Reads the contents of the IEEE 488.
155	STO	STORE	WAVEFORM TRANS.	Stores a trace in internal memory or mass storage.
154	STOP	STOP	ACQUISITION	Immediately stops signal acquisition.
156	STPN	STORE_PANEL	SAVE/RECALL	Stores front-panel setup to mass storage.
157	STST	STORE_SETUP	WAVEFORM TRANS.	Controls the way in which traces are stored.
158	STTM	STORE_TEMPLATE	WAVEFORM TRANS.	Stores the waveform template to mass storage.

Commands & Queries

Page	Short	Long Form	Subsystem	What the Command/Query Does
160	TDIV	TIME_DIV	ACQUISITION	Modifies the timebase setting.
159	TMPL?	TEMPLATE?	WAVEFORM TRANS.	Produces a complete waveform template copy.
161	TRA	TRACE	DISPLAY	Enables or disables the display of a trace.
162	TOPA	TRACE_OPACITY	DISPLAY	Controls the opacity of the trace color.
164	TRCP	TRIG_COUPLING	ACQUISITION	Sets the coupling mode of the specified trigger source.
165	TRDL	TRIG_DELAY	ACQUISITION	Sets the time at which the trigger is to occur.
163	*TRG	*TRG	ACQUISITION	Executes an ARM command.
166	TRLV	TRIG_LEVEL	ACQUISITION	Adjusts the trigger level of the specified trigger source.
167	TRLV2	TRIG_LEVEL_2	ACQUISITION	Adjusts the second trigger threshold reference level.
168	TRMD	TRIG_MODE	ACQUISITION	Specifies the trigger mode.
169	TRPA	TRIG_PATTERN	ACQUISITION	Defines a trigger pattern.
171	TRSE	TRIG_SELECT	ACQUISITION	Selects the condition that will trigger acquisition.
174	TRSL	TRIG_SLOPE	ACQUISITION	Sets the trigger slope of the specified trigger source.
175	TRWI	TRIG_WINDOW	ACQUISITION	Sets window amplitude on current Edge trigger source.
176	*TST?	*TST?	MISCELLANEOUS	Performs an internal self-test.
177	URR?	URR?	STATUS	Reads, clears User Request status Register (URR).
180	VDIV	VOLT_DIV	ACQUISITION	Sets the vertical sensitivity.
178	VMAG	VERT_MAGNIFY	DISPLAY	Vertically expands the specified trace.
179	VPOS	VERT_POSITION	DISPLAY	Adjusts the vertical position of the specified trace.
181	*WAI	*WAI	STATUS	Required by the IEEE 488.
182	WAIT	WAIT	ACQUISITION	Prevents new analysis until current is completed.
183	WF	WAVEFORM	WAVEFORM TRANS.	Transfers a waveform from controller to scope.
185	WFSU	WAVEFORM_SETUP	WAVEFORM TRANS.	Specifies amount of waveform data to go to controller.
187	WFTX	WAVEFORM_TEXT	WAVEFORM TRANS.	Documents acquisition conditions.
188	XYAS?	XY_ASSIGN?	DISPLAY	Returns traces currently assigned to the XY display.
189	XYCO	XY_CURSOR_ORIGIN	CURSOR	Sets origin position of absolute cursor measurements.
190	XYCS	XY_CURSOR_SET	CURSOR	Allows positioning of XY voltage cursors.
192	XYCV?	XY_CURSOR_VALUE?	CURSOR	Returns the current values of the X vs Y cursors.
194	XYDS	XY_DISPLAY	DISPLAY	Enables or disables the XY display mode.
195	XYSA	XY_SATURATION	DISPLAY	Sets persistence color saturation level in XY display.

Commands & Queries Tabled by Subsystem

Page	Short	Long Form	What the Command/Query Does
ACQUISITION — Controlling Waveform Acquisition			
13	ARM	ARM_ACQUISITION	Changes acquisition state from "stopped" to "single".
17	ASET	AUTO_SETUP	Adjusts vertical, timebase and trigger parameters for display.
14	ATTN	ATTENUATION	Selects the vertical attenuation factor of the probe.
18	BWL	BANDWIDTH_LIMIT	Enables or disables the bandwidth-limiting low-pass filter.
33	COMB	COMBINE_CHANNELS	Controls the acquisition system's channel-interleaving function.
43	CPL	COUPLING	Selects the coupling mode of the specified input channel.
82	GBWL	GLOBAL_BWL	Enables/disables the Global Bandwidth Limit
98	ILVD	INTERLEAVED	Enables or disables random interleaved sampling (RIS).
102	MSIZ	MEMORY_SIZE	Allows selection of maximum memory length (M-, L-models only).
105	OFST	OFFSET	Allows vertical offset adjustment of the specified input channel.
127	PDET	PEAK_DETECT	Switches the peak detector ON and OFF.
144	SCLK	SAMPLE_CLOCK	Allows control of an external timebase.
149	SEQ	SEQUENCE	Sets the conditions for Sequence-mode acquisition.
154	STOP	STOP	Immediately stops signal acquisition.
160	TDIV	TIME_DIV	Modifies the timebase setting.
164	TRCP	TRIG_COUPLING	Sets the coupling mode of the specified trigger source.
165	TRDL	TRIG_DELAY	Sets the time at which the trigger is to occur.
163	*TRG	*TRG	Executes an ARM command.
166	TRLV	TRIG_LEVEL	Adjusts the level of the specified trigger source.
167	TRLV2	TRIG_LEVEL_2	Adjusts the second trigger threshold reference level.
168	TRMD	TRIG_MODE	Specifies Trigger mode.
169	TRPA	TRIG_PATTERN	Defines a trigger pattern.
171	TRSE	TRIG_SELECT	Selects the condition that will trigger acquisition.
174	TRSL	TRIG_SLOPE	Sets the slope of the specified trigger source.
175	TRWI	TRIG_WINDOW	Sets the window amplitude in volts on current Edge trigger source.
180	VDIV	VOLT_DIV	Sets the vertical sensitivity in volts/div.
182	WAIT	WAIT	Prevents new command analysis until current acquisition completion.
COMMUNICATION — Setting Communication Characteristics			
34	CFMT	COMM_FORMAT	Selects the format to be used for sending waveform data.
36	CHDR	COMM_HEADER	Controls formatting of query responses.
37	CHLP	COMM_HELP	Controls operational level of the RC Assistant.
38	CHL	COMM_HELP_LOG	Returns the contents of the RC Assistant log.

Commands & Queries

Page	Short	Long Form	What the Command/Query Does
39	CORD	COMM_ORDER	Controls the byte order of waveform data transfers.
40	CORS	COMM_RS232	Sets remote control parameters of the RS-232-C port.
CURSOR — Performing Measurements			
44	CRMS	CURSOR_MEASURE	Specifies type of cursor or parameter measurement for display.
47	CRST?	CURSOR_SET?	Allows positioning of any one of eight independent cursors.
49	CRVA?	CURSOR_VALUE?	Returns values measured by specified cursors for a given trace.
110	PACL	PARAMETER_CLR	Clears all current parameters in Custom and Pass/Fail modes.
115	PADL	PARAMETER_DELETE	Deletes a specified parameter in Custom and Pass/Fail modes.
116	PAST?	PARAMETER_STATISTICS?	Returns current statistics values for specified pulse parameter.
117	PAVA?	PARAMETER_VALUE?	Returns current value(s) of parameter(s) and mask tests.
128	PECS	PER_CURSOR_SET	Allows positioning of any one of six independent cursors.
130	PECV?	PER_CURSOR_VALUE?	Returns values measured by specified cursors for a given trace.
120	PFCO	PASS_FAIL_CONDITION	Adds a Pass/Fail test condition or custom parameter to display.
122	PFCT	PASS_FAIL_COUNTER	Resets the Pass/Fail acquisition counters.
123	PFDO	PASS_FAIL_DO	Defines desired outcome and actions following a Pass/Fail test.
125	PFMS	PASS_FAIL_MASK	Generates a tolerance mask around a chosen trace and stores it.
126	PFST?	PASS_FAIL_STATUS?	Returns the Pass/Fail test for a given line number.
189	XYCO	XY_CURSOR_ORIGIN	Sets position of origin for absolute cursor measurements on XY display.
190	XYCS	XY_CURSOR_SET	Allows positioning of any of six independent XY voltage cursors.
192	XYCV?	XY_CURSOR_VALUE?	Returns current values of X vs Y cursors.
DISPLAY — Displaying Waveforms			
16	ASCR	AUTO_SCROLL	Controls the Auto Scroll viewing feature.
24	CHST	CALL_HOST	Allows manual generation of a service request (SRQ).
30	COLR	COLOR	Selects color of individual objects: traces, grids or cursors.
32	CSCH	COLOR_SCHEME	Selects the display color scheme.
50	DPNT	DATA_POINTS	Controls display of sample points in single display pixels or bold.
62	DISP	DISPLAY	Controls the oscilloscope display screen.
63	DTJN	DOT_JOIN	Controls the interpolation lines between data points.
64	DZOM	DUAL_ZOOM	Sets horiz. magnification and positioning for all expanded traces.
80	FSCR	FULL_SCREEN	Selects magnified view format for the grid.
83	GRID	GRID	Specifies grid display in single, dual or quad mode.
88	HMAG	HOR_MAGNIFY	Horizontally expands selected expansion trace.
89	HPOS	HOR_POSITION	Horizontally positions intensified zone's center on source trace.
97	INTS	INTENSITY	Sets grid or trace/text intensity level.
100	KEY	KEY	Displays a string in the menu field.
101	MGAT	MEASURE_GATE	Controls highlighting of region between parameter cursors.

Page	Short	Long Form	What the Command/Query Does
103	MSG	MESSAGE	Displays a string of characters in the message field.
104	MZOM	MULTI_ZOOM	Sets horiz. magnification and positioning for all expanded traces.
131	PERS	PERSIST	Enables or disables the Persistence Display mode.
132	PECL	PERSIST_COLOR	Controls color rendering method of persistence traces.
133	PELT	PERSIST_LAST	Shows the last trace drawn in a persistence data map.
134	PESA	PERSIST_SAT	Sets the color saturation level in persistence.
135	PESU	PERSIST_SETUP	Selects display persistence duration in Persistence mode.
147	SCSV	SCREEN_SAVE	Controls the automatic screen saver.
148	SEL	SELECT	Selects the specified trace for manual display control.
161	TRA	TRACE	Enables or disables the display of a trace.
162	TOPA	TRACE_OPACITY	Controls the opacity of the trace color.
178	VMAG	VERT_MAGNIFY	Vertically expands the specified trace.
179	VPOS	VERT_POSITION	Adjusts the vertical position of the specified trace.
188	XYAS?	XY_ASSIGN?	Returns the traces currently assigned to the XY display.
194	XYDS	XY_DISPLAY	Enables or disables the XY display mode.
195	XYSA	XY_SATURATION	Sets persistence color saturation level in XY display.
FUNCTION — Performing Waveform Mathematical Operations			
25	CLM	CLEAR_MEMORY	Clears the specified memory.
26	CLSW	CLEAR_SWEEPS	Restarts the cumulative processing functions.
53	DEF	DEFINE	Specifies the math expression to be evaluated by a function.
81	FRST	FUNCTION_RESET	Resets a waveform processing function.
HARD COPY — Printing the Contents of the Display Screen			
84	HCSU	HARDCOPY_SETUP	Configures the hard-copy driver.
87	HCTR	HARDCOPY_TRANSMIT	Sends string of unmodified ASCII characters to the hard-copy unit.
146	SCDP	SCREEN_DUMP	Causes a screen dump to the hard-copy device.
MASS STORAGE — Creating and Deleting File Directories			
59	DELF	DELETE_FILE	Deletes files from currently selected directory on mass storage.
60	DIR	DIRECTORY	Creates and deletes file directories on mass-storage devices.
74	FCRD	FORMAT_CARD	Formats the memory card.
73	FCR	FIND_CTR_RANGE	Automatically sets the center and width of a histogram.
76	FFLP	FORMAT_FLOPPY	Formats a floppy disk in the Double- or High-Density format.
78	FHDD	FORMAT_HDD	Formats the removable hard disk.
72	FLNM	FILENAME	Changes default filename of any stored trace, setup or hard copy.
MISCELLANEOUS — Calibration and Testing			
15	ACAL	AUTO_CALIBRATE	Enables or disables automatic calibration.
20	BUZZ	BUZZER	Controls the built-in piezo-electric buzzer.

Commands & Queries

Page	Short	Long Form	What the Command/Query Does
21	*CAL?	*CAL?	Performs a complete internal calibration of the instrument.
22	COUT	CAL_OUTPUT	Sets the type of signal put out at the CAL BNC connector.
51	DATE	DATE	Changes the date/time of the internal real-time clock.
91	*IDN?	*IDN?	Used for identification purposes.
107	*OPT?	*OPT?	Identifies oscilloscope options.
176	*TST?	*TST?	Performs an internal self-test.
PROBES — Using Probes			
137	PRCA?	PROBE_CAL?	Performs a complete calibration of the connected current probe.
138	PRDG	PROBE_DEGAUSSS?	Degausses and calibrates the connected current probe.
139	PRNA	PROBE_NAME?	Gives an identification of the probe connected to the instrument.
SAVE/RECALL SETUP — Preserving and Restoring Front-Panel Settings			
109	PNSU	PANEL_SETUP	Complements the *SAV/*RST commands.
140	*RCL	*RCL	Recalls one of five non-volatile panel setups.
142	RCPN	RECALL_PANEL	Recalls a front-panel setup from mass storage.
143	*RST	*RST	Initiates a device reset.
145	*SAV	*SAV	Stores the current state in non-volatile internal memory.
156	STPN	STORE_PANEL	Stores the complete front-panel setup on a mass-storage file.
STATUS — Obtaining Status Information and Setting Up Service Requests			
12	ALST?	ALL_STATUS?	Reads, clears contents of all (but one) of the status registers.
27	*CLS	*CLS	Clears all the status data registers.
28	CMR?	CMR?	Reads and clears contents of CoMmand error Register (CMR).
52	DDR?	DDR?	Reads and clears the Device-Dependent error Register (DDR).
65	*ESE	*ESE	Sets the standard Event Status Enable (ESE) register.
66	*ESR?	*ESR?	Reads and clears the Event Status Register (ESR).
69	EXR?	EXR?	Reads and clears the EXecution error Register (EXR).
92	INE	INE	Sets the INternal state change Enable register (INE).
93	INR?	INR?	Reads and clears the INternal state change Register (INR).
99	IST?	IST?	Individual STatus reads the current state of IEEE 488.
106	*OPC	*OPC	Sets to true the OPC bit (0) in the Event Status Register (ESR).
136	*PRE	*PRE	Sets the PaRallel poll Enable register (PRE).
151	*SRE	*SRE	Sets the Service Request Enable register (SRE).
152	*STB?	*STB?	Reads the contents of IEEE 488.
177	URR?	URR?	Reads and clears the User Request status Register (URR).
181	*WAI	*WAI	WAIt to continue — required by IEEE 488.
WAVEFORM TRANSFER —Preserving and Restoring Waveforms			
95	INSP?	INSPECT?	Allows acquired waveform parts to be read.

Page	Short	Long Form	What the Command/Query Does
141	REC	RECALL	Recalls waveform file from mass storage to internal memories.
155	STO	STORE	Stores a trace in one of the internal memories or mass storage.
157	STST	STORE_SETUP	Controls the way in which traces are stored.
158	STTM	STORE_TEMPLATE	Stores the waveform template in a mass-storage device.
159	TMPL?	TEMPLATE?	Produces a copy of the template describing a complete waveform.
183	WF	WAVEFORM	Transfers a waveform from the controller to the oscilloscope.
185	WFSU	WAVEFORM_SETUP	Specifies amount of waveform data for transmission to controller.
187	WFTX	WAVEFORM_TEXT	Documents waveform acquisition conditions.

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 ALL_STATUS? query is useful in a complete overview of the state of the instrument.

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 instruction 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 command 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 may 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`☝, `EX5`☝}
<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 instruments.
<channel> : {EX10} available on all models except those in the LC564 and LC584 Series.
<channel> : {EX5} available only on LC564 and LC584 Series models.

EXAMPLE (GPIB)

The following command 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 the instrument. 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 may be disabled by issuing the command ACAL OFF. Whenever it is 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 instruction 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 <rate>`

<rate> : = scroll rate in divisions per second in the range -10.0 to 10.00. A positive value causes it to scroll to the right while a negative value scrolls to the left. 0 will stop the scrolling.

QUERY SYNTAX

`Auto_SCRoll?`

RESPONSE FORMAT

`Auto_SCRoll <rate>`

EXAMPLE (GPIB)

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

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

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 instruments.

EXAMPLE

The following command 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 82*) 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**

Not available on some models. And the 200 MHz setting is available only on certain models.

{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 instruction turns *off* Global_BWL):

```
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 piezo-electric 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 selected by the user by turning the buzzer ON or OFF. This command is only usable in oscilloscopes fitted with the CLBZ hard option.

COMMAND SYNTAX

`BUZZer` <state>

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

EXAMPLE (GPIB)

Sending the following code 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 cause the oscilloscope to perform an internal self-calibration and generates a response that indicates whether or not the instrument 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 instrument 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 instruction 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 CAL BNC connector.

COMMAND SYNTAX

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

`Cal_OUTput PULSE`[,<width>]

<mode> : = {`OFF`, `CALSQ`, `CALPU`, `PF`, `TRIG`, `LEVEL`☝, `PULSE`☝, `TRDY`}
<level> : = 0.0 to 1.00 V into 1 MΩ
<rate> : = 500 to 2 000 000 Hz

<width> : = 10 µs 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 command 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	
CALSQ	Provides a square signal
CALPU	Provides a pulse signal
PF	PASS/FAIL mode
TRIG	Trigger Out mode
LEVEL	Provides a DC signal at the requested level
OFF	Provides no signal (ground level)
PULSE	Provides a single pulse
TRDY	Trigger is ready for a new acquisition

DISPLAY — CALL_HOST, CHST

Command/Query

DESCRIPTION

The CALL_HOST command allows the user 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 code 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 command 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 the status data registers.

COMMAND SYNTAX

```
*CLS
```

EXAMPLE (GPIB)

The following command 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 the instrument.

QUERY SYNTAX

```
CMR?
```

RESPONSE FORMAT

CMR <value>

<value> : = 0 to 13

EXAMPLE (GPIB)

The following instruction reads the contents of the CMR register:

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

Response message:

```
CMR 0
```

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 U1...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>]

AVAILABILITY

Available on color instruments only.

EXAMPLE (GPIB)

The following instruction 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 (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>`

AVAILABILITY

This command and query is available only on color instruments.

EXAMPLE (GPIB)

The following instruction 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

On most models whose channels can be combined, the COMBINE_CHANNELS command controls the channel interleaving function of the acquisition system (*see the Operator's Manual for an explanation of which models combine channels and how*). 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>`

☝ **AVAILABILITY**

<state>1 and AUTO are only available on LC584 Series instruments.

<state>4 can be used on the four-channel oscilloscopes of the 9344C, LC374 and LC584 Series, as well as on models in the 9354C, 9374C, 9384C, LC334, LC534 and LC574 Series using the supplied adapter.

EXAMPLE (GPIB)

The following instruction engages channel interleaving between Channels 1 and 2, and Channels 3 and 4 on four-channel instruments:

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

Only on LC584 Series models, 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 code 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 8-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. The instrument provides three response formats: LONG format, in which responses start with the long form of the header word; SHORT format, 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. Until the user requests otherwise, the SHORT response format is used.

This command does not affect the interpretation of messages sent to the oscilloscope. Headers may 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`}

Note: The default mode, i.e. the mode just after power-on, is SHORT.

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 oscilloscope Operator's Manual*), which assists in debugging remote control programs. Selected when using the instrument's front-panel via the "UTILITIES" and "SPECIAL MODES" menus, Remote Control Assistant can log all message transactions occurring between the external controller and the oscilloscope. The log may be viewed at any time in the provided menu on the screen and has four levels to choose from:

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 code reads the remote control log and prints it:

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

RELATED COMMANDS

COMM_HELP

COMMUNICATION — COMM_ORDER, CORD

Command/Query

DESCRIPTION

The COMM_ORDER command controls the byte order of waveform data transfers. Waveform data may be sent with the most significant byte (MSB) or the least significant byte (LSB) in the first position. The default mode is 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 selected, the most significant byte is sent first. When "COMM_ORDER LO" is specified, the least significant byte is sent first.

The COMM_ORDER? query returns the byte transmission order currently in 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/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 command COMM_RS232 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 is ONLY valid if the oscilloscope is being remotely controlled via the RS-232-C port.

The parameters are:

(a) **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.

(b) **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.

(c) **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.

(d) **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.

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

Some parameters of this command require ASCII strings as actual arguments. In order to facilitate the embedding of non-printable characters into such strings, escape sequences may 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": 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,"<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 instrument 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`☝,`EX5`☝}

<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 instrument will disconnect the input.

☝ **AVAILABILITY**

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

<channel> : = {EX10} available on all models except those in the LC564 and LC584 Series.

<channel> : = {EX5} available on LC564 and LC584 Series oscilloscopes only.

A1M and D1M are not available on model 9362C.

EXAMPLE GPIB)

The following command 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 the 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 1: 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.

Note 2: 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 command switches on the vertical relative cursors:

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

The following command 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 the following four forms:

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

To turn on parameter measurements without statistics, use one of the following three forms:

```
CURSOR_MEASURE CUST
CURSOR_MEASURE HPAR
CURSOR_MEASURE VPAR
```

To turn on parameter statistics, add the keyword `STAT` to the above three forms.

To turn on Pass or Fail tests on parameter or mask tests, use:

```
CURSOR_MEASURE PASS
CURSOR_MEASURE FAIL
```

Use the command:

```
PASS_FAIL_CONDITION
```

to select parameters in the Custom mode, and to modify the test conditions in the Pass/Fail mode.

CURSOR — CURSOR_SET, CRST

Command/Query

DESCRIPTION

The CURSOR_SET command allows the user 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.

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

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 70). To change only the trace without repositioning the cursors, the CURSOR_SET command may be given with no argument (for example,. TB:CRST).

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 1: The suffix DIV is optional.

Note 2: Parameters are grouped in pairs. The first parameter specifies the cursor to be modified and the second one indicates its new value. Parameters may be grouped in any order and may be restricted to those items to be changed.

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 command 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	VABS	vertical absolute
HREL	horizontal relative	VREL	vertical 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>
<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>`

AVAILABILITY

Command/Query available on LC Series oscilloscopes only.

EXAMPLE (GPIB)

The following instruction 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 2089
<hour> : = 0 to 23
<minute> : = 0 to 59
<second> : = 0 to 59

Note: It is not always necessary to specify all the DATE parameters. Only those parameters up to and including the parameter to be changed need be specified in order to change the "year" setting, specify day, month and year together with the required settings. The time settings will remain unchanged. 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 instruction 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 instruments.

EXAMPLE (GPIB)

The following instruction 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 functions in the standard oscilloscopes and WP0X processing packages.

COMMAND SYNTAX

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

Note 1: 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.

Note 2: 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>]

<param_name>	<value>	Description
EQN	'<equation>'	Function equation as defined below
DELAY	<delay>	Delay by time
MAXPTS	<max_points>	Max. number of points to compute
SWEEPS	<max_sweeps>	Maximum number of sweeps
Parameters To Support Additional Functions in WP01		
BITS	<bits>	Number of ERES bits
UNITS	<units>	Physical units
WEIGHT	<weight>	Continuous Average weight
Parameters To Support Additional Functions in WP02		
WINDOW	<window_type>	FFT window function

Parameters To Support Additional Functions in WP03 or DDM		
MAXBINS	<bins>	Number of bins in histogram
MAX_EVENTS	<max_values>	Max. no. of values in histogram
CENTER		Horizontal center position for histogram display.
WIDTH	<width>	Width of histogram display
VERT	<vert_scale>	Vertical scaling type
Parameters To Support Additional Functions in PRML		
LENGTH	<length>	No. points to use from first waveform
START	<start>	Starting point in second waveform

Function Equations And Names Available On All Models	
<source>	Identity
+<source>	Identity
-<source>	Negation
<source1> + <source2>	Addition
<source1> - <source2>	Subtraction
<source1><source2>	Multiplication
<source1>/<source2>	Ratio
AVGS(<source>)	Average Summed
RESAMP(<source>)	Resample (deskew)
SINX(<source>)	Sin(x)/x interpolator
ZOOMONLY (<extended_source>)	Zoom only (No Math)
Extended Functions Available On Instruments With WP01 Processing Firmware	
ABS(<source>)	Absolute Value
AVGC(<source>)	Continuous Average
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)
FLOOR(EXTR(<source>))	Floor (Extrema source only)
INTG(<source>[{+,-} <addend>])	Integral
LN(<source>)	Logarithm base e
LOG10(<source>)	Logarithm base 10
RESC([{+,-}][<multiplier>*]<source>[{+,-}<addend>])	Rescale
ROOF(EXTR(<source>))	Roof (Extrema source only)
1/<source>	Reciprocal
SQR(<source>)	Square
SQRT(<source>)	Square Root
FFT Functions Available on Instruments with WP02 Processing Firmware ***Note: The source waveform must be a time-domain signal, single segment.***	
FFT(<source>)	Fast Fourier Transform (complex result)
REAL(FFT(<source>))	Real part of complex result
IMAG(FFT(<source>))	Imaginary part of complex result
MAG(FFT(<source>))	Magnitude of complex result
PHASE(FFT(<source>))	Phase angle (degrees) of complex result
PS(FFT(<source>))	Power spectrum
PSD(FFT(<source>))	Power density
RESC([{+,-}][<multiplier>*]<source>[{+,-}<addend>])	Rescale

Power Average Functions Available on Instruments with WP02 Processing Firmware ***Note: The source waveform must be another function defined as a Fourier transform.***		
MAG(AVGP(<function>))	PS(AVGP(<function>))	PSD(AVGP(<function>))

Function Equations and Names Available on Instruments with WP03 or DDM Firmware	
HIST(<custom_line>)	Histogram of parameter on custom line
Function Equations and Names Available on Instruments with PRML Firmware	
CORR(<source1>,<source2>)	Cross Correlation

Source values

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

<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 (For standard instruments)

<max_sweeps> : = 1 to 1 000 000 (For WP01 only)

<max_sweeps> : = 1 to 50 000 (WP02 Power Spectrum only)

Values for Resample Function

<delay> : = –2e–6 to +2e–6 seconds

Values for Rescale Function

<addend> : = 0.0 to 1e15

<multiplier> : = 0.0 to 1e15

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 window function

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

FFT Window Function Notation	
LHA	Blackman–Harris window
FLTP	Flat Top window
HABMM	Hamming window
HANN	von Hann window
RECT	Rectangular window

Values for WP03 histogramming

<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

Values for PRML correlation

<length> : = 0 to 10 divisions

<start> : = 0 to 10 divisions

AVAILABILITY

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

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

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

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

EXAMPLE (GPIB)

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

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

WP01 EXAMPLE

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

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

WP02 FFT EXAMPLE (GPIB) The following instruction 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$="TA:DEF EQN,'PS(FFT(C1))',MAXPTS,1000,WINDOW,
RECT": CALL IBWRT(SCOPE%,CMD$)
```

WP02 PS EXAMPLE (GPIB) The following instruction 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$="TB:DEF EQN,'PS(AVGP(TA))',SWEEPS,244":
CALL IBWRT(SCOPE%,CMD$)
```

WP03 EXAMPLE The following command defines Trace C to construct the histogram of the all rise time measurements made on source Channel 1. The rise time measurement is defined on custom line 2. The histogram has a linear vertical scaling and the rise time 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`☝}

<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 MC01 option is fitted.

<device> : HDD available only when HD01 option is fitted.

EXAMPLE (GPIB)

The following command 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`☝}

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

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

Note: the query DIRectory_list? is also accepted for backward compatibility but may not be supported in the future.

RESPONSE FORMAT

`DIRectory DISK,`<device> `"`<directory>`"`

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

☝ **AVAILABILITY**

<device> : CARD available only when MC01 option is fitted.

<device> : HDD available only when HD01 option is fitted.

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 of 04-MAR-1994
  10:46:20 on Memory Card
SC1 000    2859     19-DEC-1994  16:33:06
SC1 001    2859     19-DEC-1994  16:34:32
TEST5       002     20359        12-MAR-1994
    13:34:12
3 File(s) 1948672 bytes free
"
```

DISPLAY

DISPLAY, DISP
Command/Query

DESCRIPTION

The DISPLAY command controls the display screen of the oscilloscope. When the user is remotely controlling the oscilloscope and does not need to use the display, it can be useful to switch off the display via the DISPLAY OFF command. This improves instrument 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 the display has been set to OFF, the real-time clock and the message field are updated. However, the waveforms and associated texts remain unchanged.

COMMAND SYNTAX

`DISPlay` <state>

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

QUERY SYNTAX

`DISPlay?`

RESPONSE FORMAT

`DISPlay` <state>

EXAMPLE (GPIB)

The following instruction turns off the display generation:

```
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 instruction 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 instruction turns dual zoom on:

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

RELATED COMMANDS

HOR_MAGNIFY, HOR_POSITION, MULTI_ZOOM

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 68.*

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 instruction 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 instruction 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		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 specific Error occurred	(5)
2	4	QYE	1	QuerY Error occurred	(6)
1	2	RQC	0	Instrument never requests bus control	(7)
0	1	OPC	0	OPeration Complete bit not used	(8)

Notes

(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 may 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 for further details.*

QUERY SYNTAX

```
EXR?
```

RESPONSE FORMAT

```
EXR <value>
```

<value> : = 21 to 64

EXAMPLE (GPIB)

The following instruction 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 instrument 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, or 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. *

* *For floppy disk and on oscilloscopes fitted with the memory card (MCO1) or hard disk (HD01) options.*

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 8 characters forming a legal DOS filename. Up to 5 characters for SETUP and HCOPY.

Note: No extension can be specified as this is automatically assigned by the oscilloscope.

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 command designates channel 1 waveform files to be "TESTPNT6.xxx" where xxx is a numeric extension assigned by the oscilloscope:

```
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 on instruments fitted with WP03 or DDM option.

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, PACU

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 on instruments fitted with the MC01 option.

EXAMPLE (GPIB)

The following code 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 code 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 supplied, QUICK will be 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 on instruments fitted with the HD01 option.

EXAMPLE (GPIB)

The following code 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

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>`

AVAILABILITY

Available only on color instruments.

EXAMPLE (GPIB)

The following instruction 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 instruction will restart the averaging process:

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

RELATED COMMANDS

DEFINE, INR

ACQUISITION — GLOBAL_BWL, GBWL

Command/Query

DESCRIPTION

This turns on or off the Global Bandwidth Limit. When activated, the Bandwidth Limit applies to all channels; when deactivated, a Bandwidth Limit can be set individually for each channel (*see BWL, page 18*). 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 instruction 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 specifies whether the display is in single (1), dual (2), quad (4), XY or octal (8) grid mode.

The GRID? query returns the grid mode currently in use.

COMMAND SYNTAX

`GRID` <grid>

<grid> : = {`SINGLE`, `DUAL`, `QUAD`, `OCTAL`, `XYONLY`☝}

QUERY SYNTAX

`GRID?`

☝ **AVAILABILITY**

<grid> : = XYONLY available only when XY Display used.

RESPONSE FORMAT

`GRID` <grid>

EXAMPLE (GPIB)

The following instruction sets the screen display to dual grid mode:

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

RELATED COMMANDS

COLOR, INTENSITY, FULL_SCREEN

HARD COPY — HARDCOPY_SETUP, HCSU
Command/Query

DESCRIPTION

The HARDCOPY_SETUP command configures the instrument's hard-copy driver. It enables the user 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☝}
<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 available when MC01 Option is fitted.
<port> : HDD only when HD01 Option is fitted.
<port> : PRT only when GP01 option is fitted.
<cmdiv> only when GP01 option is fitted.
<auto> only when GP01 option is fitted.
<device> *See table page 86* .

The following table lists the printer and graphic formats that can be used for producing hardcopies remotely, either with monochrome or color digital oscilloscopes, or both, using <device>.

Notation	Printer, Plotter or Protocol	Monochrome Instruments	Color Instruments
BMP	BMP	✔	✔
BMPCOMP	BMP compressed	*not available*	✔
CANONCOL	Canon 200/600/800 Series color printers	*not available*	✔
EPSON	Epson b&w	✔	✔
EPSONCOL	Epson color	*not available*	✔
HPTJ	HP ThinkJet	✔	*not available*
HPPJ	HP Paint Jet	✔	*not available*
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	✔	✔
TIFFCOMP	TIFF compressed	✔	*not available*
TIFFCOL	TIFF color	*not available*	✔

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

Hardcopy 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 may be given in any order and may be restricted to those variables to be changed.

The following table gives the Hardcopy command notations and their meanings.

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
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 the user 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 instruction 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 153*) 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 instruction 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 153*) 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 1: The suffix DIV is optional.

Note 2: 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.

QUERY SYNTAX <exp_trace> : `Hor_POSition?`

RESPONSE FORMAT <exp_trace> : `Hor_POSition` <hor_position>[,]

Note 3: The segment number is only given for sequence waveforms.

EXAMPLE (GPIB) The following instruction 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 example issues an identification request to the scope:

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

Response message:

```
*IDN LECROY,9314CM,931401000,7.7.0
```

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 instruction 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...14		0	Reserved for future use
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
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

QUERY SYNTAX

`INR?`

RESPONSE FORMAT

`INR` <state>

<state> : = 0 to 65535

EXAMPLE (GPIB)

The following instruction 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 the user 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 query TEMPLATE? 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*).

The special logical block named WAVEDESC may 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 (m.s.b.†)
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 m.s.b.)

QUERY SYNTAX

<trace> : INSPect? '<string>'[,<data_type>]

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

<string> : = A valid name of a logical block or a valid name of a variable contained in block WAVEDESC (*see the command TEMPLATE*).

<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.

† *most significant bits*

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 instruments.

EXAMPLES (GPIB)

The following instruction 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 command 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/text.

The intensity level is expressed as a percentage (PCT). A level of 100 PCT corresponds to the maximum intensity whilst 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 1: Parameters are grouped in pairs. The first of the pair names the variable to be modified, whilst the second gives the new value to be assigned. Pairs may be given in any order and be restricted to those variables to be changed.

Note 2: The suffix PCT is optional.

QUERY SYNTAX

`INTenSity?`

RESPONSE FORMAT

`INTenSity TRACE,`<value>`,GRID,`<value>

EXAMPLE (GPIB)

The following instruction 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 instrument Operator's Manual, Appendix A, for specifications.*

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>`

AVAILABILITY

Query but not command available on model 9361C.

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 instruction 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

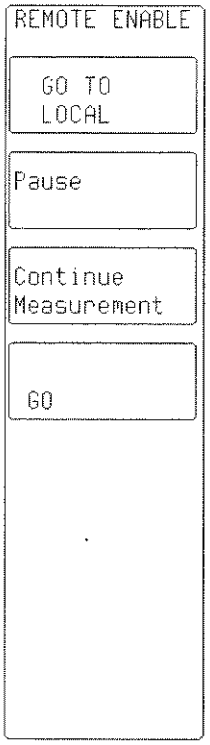
DISPLAY — KEY Command

DESCRIPTION

The KEY command allows control of a program from the front panel (*menus illustrated at right*). It displays strings of up to two lines of 13 characters as menus corresponding to and operated by the lower six menu buttons or soft keys (the top menu and button is automatically "GO TO LOCAL").

String text assigned by the operator to these menus disappears on the next transition to local but reappears when the instrument is switched back into the remote state. Text is cleared at power-up, when the instrument is reset, or if an empty string is assigned to a location (for example: `KEY 2, ' '`).

Pressing any one of the menu buttons while 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>'

<button> : = 1 to 5
<string> : = Up to two 13-character strings (any ASCII code)

EXAMPLE (GPIB)

The menus illustrated this page were created by issuing the following instructions:

```
CMD$="KEY 2, 'Pause'; KEY 3,
'Continue','Measurement';
KEY 4, ' ', 'GO'": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

URR

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>

AVAILABILITY

Available only on color instruments.

EXAMPLE (GPIB)

The following instruction 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 Appendix A of the oscilloscope Operator's Manual for memory size.*

Reducing the number of data points results in 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>

<size> : = {`500, 1000, 2500, 5000, 10K, 25K, 50K, 100K, 250K, 500K, 1M, 2.5M, 5M, 10M`}

Or, alternatively, in standard numeric format

= {`500, 1e+3, ..., 2e+6, 4e+6, 8e+6`}, for example.

Note: The instrument will adapt to the closest valid <size> or numerical <value> according to available channel memory.

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, COMB?

DISPLAY — MESSAGE, MSG
Command/Query

DESCRIPTION

The MESSAGE command displays a string of characters in the Message Field above the grid. The string may be up to 49 characters in length. The string is displayed as long as the instrument 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 instrument is reset, or if an empty string is sent (MSG " ").

The MESSAGE? query allows the user 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 code 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

By setting 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 oscilloscope Operator's Manual, Appendix A, for specifications.*

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 OFFSET? query returns the DC offset value of the specified channel.

COMMAND SYNTAX

<channel> : `OFfSeT` <offset>

<channel> : = {`C1`, `C2`, `C3`☝,`C4`☝}
<offset> : = *See Appendix A of the instrument Operator's Manual for specifications.*

Note: The suffix V is optional.

QUERY SYNTAX

<channel> : `OFfSeT`?

RESPONSE FORMAT

<channel> : `OFfSeT` <offset>

☝ **AVAILABILITY**

<channel> : {C3, C4} only on four-channel instruments.

EXAMPLE (GPIB)

The following command sets the offset of Channel 2 to -3 V:

```
CMD$="C2:OFST -3V": CALL IBWRT(SCOPE%,CMD$)
```

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 instrument 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, i.e. additional firmware or hardware options. 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 WP01, WP02, WP03, DDM, CKIO, PRML, ORM, MC01 and JTA are installed, the response will be returned as:

```
*OPT WP01,WP02,DDM,CKIO,PRML,ORM,MC01,JTA
```

Response message if no options are installed:

```
*OPT 0
```

ADDITIONAL INFORMATION

Notation	
CKTR	CKTRIG Clock-Trigger-Ext. ref. Option
DDFA	Disk Drive Failure Analysis Option
DDM	Disk Drive Measurements Option
FD01	Floppy Disk Option
GP01	Internal Printer/Centronics Option
HD01	Hard Disk Option
JTA	Jitter and Timing Analysis Option
ORM	Optical Recording Measurements Option
PMT	Power Measurement Tools
PRML	PRML Measurements Option
MC01	Memory Card Option
WP01	Waveform Processing Option WP01
WP02	Waveform Processing Option WP02
WP03	Waveform Processing Option WP03

SAVE/RECALL SETUP — PANEL_SETUP, PNSU

Command/Query

DESCRIPTION

The PANEL_SETUP command complements the *SAV/*RST commands. The PANEL_SETUP command allows panel setups to be archived in encoded form on external storage media.

Only setup data read by the PNSU? query may be recalled into the oscilloscope. A panel setup error (*see table on page 70*) 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 instruction saves the instrument'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 instruction 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, (for example, *Dt@lev* or *r@level*) and may also be used to assign any parameter for histogramming.

Note: The measured value of a parameter setup with PACU may be read using PAVA?

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
Parameters available on all models		
DC2DPOS	delta clock to data positive	<source1>,<clockedge>,<level1>,<source2>, <slope2>, <level2>,<hysteresis>
DC2DNEG	delta clock to data negative	<source1>,<clockedge>,<level1>,<source2>, <slope2>, <level2>,<hysteresis>
DDLY	delta delay	<source1>,<source2>
DTLEV	delta time at level	<source1>,<slope1>,<level1>,<source2>,<slope2>,<level2>,<hysteresis>
FLEV	fall at level	<source>,<high>,<low>
PHASE	phase difference	<source1>,<edge1>,<level1>,<source2>,<edge2>,<level2>,<hysteresis>,<angular unit>
RLEV	rise at level	<source>,<low>,<high>
TLEV	time at level	<source>,<slope>,<level>,<hysteresis>
Parameters available on instruments equipped with WP03 or DDM processing firmware		
FWXX	full width at xx% of max	<source>,<threshold>
PCTL	percentile	<source>,<threshold>
XAPK	x position at peak	<source>,<rank>

<param>	definition	<qualifier> list
Parameters available on instruments equipped with DDM processing firmware		
LBASE	local base	<source>,<hysteresis>
LBSEP	local baseline separation	<source>,<hysteresis>
LMAX	local maximum	<source>,<hysteresis>
LMIN	local minimum	<source>,<hysteresis>
LNUM	number of local events	<source>,<hysteresis>
LPP	local peak to peak	<source>,<hysteresis>
LTBE	local time between events	<source>,<hysteresis>
LTBP	local time between peaks	<source>,<hysteresis>
LTBT	local time between troughs	<source>,<hysteresis>
LTMN	local time at minima	<source>,<hysteresis>
LTMX	local time at maxima	<source>,<hysteresis>
LTOT	local time over threshold	<source>,<hysteresis>,<threshold> .
LTPT	local time peak to trough	<source>,<hysteresis>
LTTP	local time trough to peak	<source>,<hysteresis>
LTUT	local time under threshold	<source>,<hysteresis>,<threshold>
NBPH	narrow band phase	<source>,<freq>
NBPW	narrow band power	<source>,<freq>
OWRITE	overwrite	<source 1>,<source 2>,<freq>
PW50	pulse width 50	<source>,<hysteresis>
PW50NEG	pulse width 50 for troughs	<source>,<hysteresis>
PW50POS	pulse width 50 for peaks	<source>,<hysteresis>
RES	resolution	<source 1>,<source 2>,<hysteresis>
TAA	track average amplitude	<source>,<hysteresis>
TAANEG	track average amplitude for troughs	<source>,<hysteresis>
TAAPOS	track average amplitude for peaks	<source>,<hysteresis>

<param>	definition	<qualifier> list
Parameters available on instruments equipped with PRML processing firmware		
ACSN	auto correlation signal to noise	<source>,<length>
NLTS	non-linear transition shift	<source>,<length>,<delay>

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
<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 instruments.

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

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
```

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 instruction 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
NUM_ACQ	number of contributing acquisitions
NUM_VALUES	number of measurements taken for parameter
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, NUM_ACQ, NUM_VALUES, 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(s) of the pulse waveform parameter(s) 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.

Parameters Available on All Models					
ALL	all parameters	**DUTY**	duty cycle	**OVSP**	positive overshoot
AMPL	amplitude	**FALL**	falltime	**PER**	period
AREA	area	**FALL82**	fall 80 to 20%	**PKPK**	peak-to-peak
BASE	base	**FREQ**	frequency	**PNTS**	points
CMEAN	mean for cyclic waveform	**FRST**	first point	**RISE**	risetime
CMEDI	median for cyclic waveform	**LAST**	last point	**RISE28**	rise 20 to 80%
CRMS	root mean square for cyclic part of waveform	**MAX**	maximum	**RMS**	root mean square
CSDEV	standard deviation for cyclic part of waveform	**MEAN**	mean	**SDEV**	standard deviation
CYCL	cycles	**MEDI**	median value	**TOP**	top
DLY	delay	**MIN**	minimum	**WID**	width
DUR	duration of acquisition	**OVSN**	negative overshoot		
Custom Parameters Defined using PARAMETER_CUSTOM Command‡					
CUST1	CUST2	CUST3	CUST4	CUST5	
Parameters Available on Instruments with WP03 or DDM Processing Firmware					
AVG	average of distribution	**HMEDI**	median of a histogram	**PKS**	number of peaks
DATA	data values	**HRMS**	histogram rms value	**RANGE**	range of distribution
FWHM	full width at half max	**HTOP**	histogram top value	**SIGMA**	sigma of distribution
HAMPL	histogram amplitude	**LOW**	low of distribution	**TOTP**	total population
HBASE	histogram base	**MAXP**	maximum population		

‡ *The numbers in the terms CUST1, CUST2, CUST3, CUST4 and CUST5 refer to the line numbers of the selected custom parameters.*

HIGH	high of 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 parameter names on previous page.*

Alternative forms of query for mask tests:

<trace> : `PArameter_VAlue?` <old_mask_test>

<trace> : `PArameter_VAlue?` <mask_test>, <mask>

<mask_test> : = {`ALL_IN, SOME_IN, ALL_OUT, SOME_OUT`}

<old_mask_test> : = {`ALLI, ANYI, ALLO, ANYO`}

<mask> : = {`TA, TB, TC, TD`}

Note: Old mask test keywords ALLI, ANYI, ALLO, ANYO imply testing of <trace> against the mask waveform TD. Old mask test keywords INSIDE and OUTSIDE are equivalent to ALL_IN and SOME_OUT; they are only supported for compatibility with older-model instruments.

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 the standard voltage and standard time parameters followed by their values and states are returned.

AVAILABILITY

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

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 1: 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 parameter names on pages 111 and 117.*

<rel_op> : = {GT, LT}

<ref_value> : = –1e15 to +1e15

Note 2: 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>.

Note 3: Old mask test keywords ALLI and ANYO imply testing of <trace> against the mask waveform TD. Old mask test keywords INSIDE and OUTSIDE are equivalent to ALL_IN and SOME_OUT; they are only supported for compatibility with former versions.

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 instruments.

EXAMPLE (GPIB)

The following instruction 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.

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 1: The BEEP command is accepted only on models equipped with the CLBZ hardware option.

Note 2: The PULS command is accepted only on models equipped with the CKIO software option.

Note 3: The PFDO command with no arguments (i.e. "PFDO") deletes all actions.

Note 4: The STO command performs the store operation as described in the Waveform Store chapter in the Operator's Manual.

Note 5: After every pass or fail detected, the instrument sets the INR bit 12.

QUERY SYNTAX

`Pass_Fail_DO?`

RESPONSE FORMAT

`Pass_Fail_DO` [<pass_fail>[,<act>[,<act>...]]]

EXAMPLE (GPIB)

This following instruction 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 instruments.

EXAMPLE (GPIB)

The following instruction 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

ACQUISITION — PEAK_DETECT, PDET

Command/Query

DESCRIPTION

The PEAK_DETECT command switches ON or OFF the peak detector built into the acquisition system.

The PEAK_DETECT? query returns the current status of the peak detector.

COMMAND SYNTAX

`Peak_DETect` <state>

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

QUERY SYNTAX

`Peak_DETect?`

RESPONSE FORMAT

`PDET` <state>

AVAILABILITY

Available on 9350C, 9354C, 9370C, 9374C, 9384C, LC534A and LC574A models only.

EXAMPLE (GPIB)

The following instruction turns on the peak detector:

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

CURSOR — PER_CURSOR_SET, PECS

Command/Query

DESCRIPTION

The PER_CURSOR_SET command allows the user 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(s).

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 1: The suffix DIV is optional.

Note 2: Parameters are grouped in pairs. The first of the pair names the variable to be modified, whilst the second gives the new value to be assigned. Pairs may be in any order and be restricted to those variables to be changed.

QUERY SYNTAX

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

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

Note 3: 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.

RESPONSE FORMAT

`PEr_Cursor_Set` <cursor>,<position>[,<cursor>,<position>,...,<cursor>,<position>

AVAILABILITY

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

EXAMPLE (GPIB)

The following code 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,7.4DIV"
```

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 instruments.

EXAMPLE (GPIB)

The following code 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 code 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.

COMMAND SYNTAX

`PErsist_CoLor` <state>

<state> : = {`ANALOG`, `COLOR_GRADED`}

QUERY SYNTAX

`PErsist_CoLor?`

RESPONSE FORMAT

`PErsist_CoLor` <state>

AVAILABILITY

Available only on color instruments.

EXAMPLE (GPIB)

The following instruction 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>`

AVAILABILITY

Available on color instruments only.

EXAMPLE (GPIB)

The following instruction 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, ALL`}

<value> : = 0 to 100 PCT

Note: The suffix PCT is optional.

QUERY SYNTAX

`PErsist_SAt?`

RESPONSE FORMAT

`PErsist_SAt` <trace>,<value>

AVAILABILITY

Available on color instruments only.

EXAMPLE (GPIB)

The following instruction sets the saturation level of the persistence data map for channel 3 to be 60%, i.e. 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 instruction 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. The *PRE command allows the user 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 instruction 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. This 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 the instrument. At the end of this calibration, the response indicates how the calibration has terminated, and the instrument 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 instruction 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 the instrument. 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

EXAMPLE (GPIB)

The following instruction 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_NAME?, PRNA?

Query

DESCRIPTION

The PROBE_NAME? query returns the name of a probe connected to the instrument. Passive probes are identified by their attenuation factor.

QUERY SYNTAX

<channel> : `PROBE_NAME?`

RESPONSE FORMAT

<channel> : `PROBE_NAME` <probe name>

EXAMPLE (GPIB)

The following instruction 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 the instrument, using one of the five non-volatile panel setups, by recalling the complete front-panel setup of the instrument. Panel setup 0 corresponds to the default panel setup.

The *RCL command produces the opposite effect 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 the instrument setup previously stored in panel setup 3:

```
CMD$="*RCL 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *SAV, EXR

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 that 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 when MC01 Option is fitted.

<device> : HDD only available when HD01 Option is fitted.

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 when MC01 Option is fitted.

<device> : HDD only available when HD01 Option is fitted.

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

SAVE/RECALL SETUP — *RST Command

DESCRIPTION

The *RST command initiates a device reset. The *RST sets all eight traces to the GND line and recalls the default setup.

COMMAND SYNTAX

```
*RST
```

EXAMPLE (GPIB)

This example 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 the user to control an external timebase. The user sets the number of data points that will be acquired when the instrument 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, RP`☝}

<recordlength> : = {`50, 100, 200, 500, 1K, 2K, 5K, 10K, 20K, 50K, 100K, 200K, 500K, 1M, 2M`}

Or, alternatively, in standard numeric format:

= {`10e+3, 10.0e+3, 11e+3`...}, for example.

<coupling> : = {`D1M` or `D50`☝}

Note: The record length cannot be larger than the maximum available memory of the model being used. The instrument will adapt to the closest valid <recordlength>. See Appendix A of the instrument Operator's Manual for maximums.

QUERY SYNTAX

`Sample_CLocK? [NUM]`

☝ **AVAILABILITY**

Not available on 9361C or 9362C Series.

<state> : {RP} only available on oscilloscopes fitted with the CKTRIG option.

<coupling> : {D50} not available when <state> {RP} is selected.

RESPONSE FORMAT

`Sample_CLocK` <state>,<recordlength>

EXAMPLE

The following sets the instrument 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 the instrument in non-volatile internal memory. The *SAV command stores the complete front-panel setup of the instrument at the time the command is issued.

*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 (*SRE, *PRE, *ESE, INE) are not saved by this command.*

COMMAND SYNTAX

`*SAV` <panel_setup>

<panel_setup> : = 1 to 4

EXAMPLE (GPIB)

The following saves the current instrument setup in Panel Setup 3:

```
CMD$="*SAV 3": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PANEL_SETUP, *RCL

HARD COPY — SCREEN_DUMP, SCDP

Command/Query

DESCRIPTION

The SCREEN_DUMP command causes the oscilloscope to dump the screen contents onto the hard-copy device. This command will halt the instrument'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 save 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>

AVAILABILITY

Available only on color models.

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 70*) 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`, `WRAP`}

<segments> : = the right-hand column in the table below:

Max. memory length per channel	Max. number of segments
10 k	50
25 k	50
50 k	200
100 k	500
200 k	500
250 k	500
500 k	2000
1 M	2000
2 M	2000

<max_size> : = {`50, 100, 250, 500, 1000, 2500, 5K, 10K, 25K, 50K, 100K, 250K, 500K, 1M`}

Or, alternatively, in standard numeric format:

= {...`10e+3`, `10.0e+3`,...`11e+3`,...}, for example.

Note: The instrument 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}

AVAILABILITY

Not available on 9361C or 9362C Series.

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

STATUS — *SRE

Command/Query

DESCRIPTION

The *SRE command sets the Service Request Enable register (SRE). This command allows the user to specify which summary message bit(s) in the STB register will generate a service request. *Refer to the table on page 153 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 instruction 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 register (STB), and the Master Summary Status (MSS). The response represents the values of bits 0 to 5 and 7 of the Status Byte 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 153 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)

Notes

(1) The Master Summary Status (MSS) indicates that the instrument requests service, whilst 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.

ACQUISITION — STOP Command

DESCRIPTION

The STOP command immediately stops the acquisition of a signal. If the trigger mode is AUTO or NORM, it will change to trigger mode STOPPED 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`☝}

Note: If the STORE command is sent without any argument, all traces currently enabled in the Store Setup will be stored. This setup can be modified using the STORE_SETUP command.

☝ **AVAILABILITY**

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

<dest> : CARD only available when MC01 option is fitted.

<dest> : HDD only available when HD01 option is fitted.

EXAMPLE (GPIB)

The following command stores the contents of Trace A (TA) into Memory 1 (M1):

```
CMD$="STO TA,M1": CALL IBWRT(SCOPE%,CMD$)
```

The following command stores all currently displayed waveforms onto the memory card:

```
CMD$="STO ALL_DISPLAYED, CARD":
CALL IBWRT(SCOPE%,CMD$)
```

The following command 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 the instrument, 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 (*SRE, *PRE, *ESE, INE) are not saved by this command.*

COMMAND SYNTAX

`STore_PaNel DISK,` <device>, `FILE,'`<filename>`'`

<device> : = {`CARD`☝, `FLPY`, `HDD`☝}

<filename> : = A string of up to 8 characters, with the extension "`.PNL`".

Note: If no filename (or an empty string) is supplied, the instrument generates a filename according to its internal rules.

AVAILABILITY

<device> : CARD only available when MC01 option is fitted.

<device> : HDD only available when HD01 option is fitted.

EXAMPLE (GPIB)

The following code saves the current instrument 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 may be enabled for storage. This applies to auto-storing or to the STORE, STO command. Traces may 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 that only waveforms stored in BINARY format can be recalled.

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 when MC01 option is fitted.

<dest> : HDD only available when HD01 option is fitted.

EXAMPLE (GPIB)

The following command selects Channel 1 to be stored. 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 the instrument'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.

Note: For revision 2.1, for example, the file name generated will be LECROY21.TPL.

Refer to Chapter 4 for more on the waveform template, and Appendix B for a copy of the template itself.

COMMAND SYNTAX

`STore_TeMplate DISK,`<device>

<device> : = {`CARD`☝, `FLPY`, `HDD`☝}

AVAILABILITY

<device> : CARD only available when MC01 option is fitted.

<device> : HDD only available when HD01 option is fitted.

EXAMPLE (GPIB)

The following code stores the current waveform template on the memory card for future reference:

```
CMD$="STTM DISK, CARD": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TEMPLATE

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. *Refer to Chapter 4 for more on the waveform template, and Appendix B 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 may 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 153*) to be set.

The TIME_DIV? query returns the current timebase setting.

COMMAND SYNTAX

`Time_DIV` <value>

<value> : = *See Appendix A of the instrument Operator's Manual for specifications.*

Note: 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 µsec/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 70*) 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 instruments.

EXAMPLE (GPIB)

The following command displays Trace A (TA):

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

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>

AVAILABILITY

Available on color instruments only.

EXAMPLE (GPIB)

The following allows traces to be distinguished even when they overlap:

```
CMD$="TOPA TRANSPARENT": CALL BWRT(SCOPE%,CMD$)
```

ACQUISITION — *TRG Command

DESCRIPTION

The *TRG command executes an ARM command.

*Note: The *TRG command is the equivalent of the 488.1 GET (Group Execute Trigger) message.*

COMMAND SYNTAX

```
*TRG
```

EXAMPLE (GPIB)

The following command 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.

Note 1: The trigger slope is automatically determined by the instrument when in HFDIV coupling. The TRIG_SLOPE command is not used in HFDIV coupling mode.

Note 2: HFDIV is indicated as HF in the trigger setup menus.

The TRIG_COUPLING? query returns the trigger coupling of the selected source.

COMMAND SYNTAX

<trig_source> : `TRig_CouPling` <trig_coupling>

<trig_source> : = {`C1`, `C2`, `C3`☝, `C4`☝, `EX`, `EX10`☝, `EX5`☝}

<trig_coupling> : = {`AC`☝, `DC`, `HFREJ`☝, `LFREJ`☝, `AUTO`☝}

QUERY SYNTAX

<trig_source> : `TRig_CouPling?`

RESPONSE FORMAT

<trig_source> : `TRig_CouPling` <trig_coupling>

☝ **AVAILABILITY**

<trig_source> : = {C3, C4} only on four-channel instruments.

<trig_source> : = EXT10 not on LC564 or LC584 Series models.

<trig_source> : = EXT5 only on LC564 and LC584 Series models.

<trig_coupling> : AUTO only available on model 9362C.

<trig_coupling> : AC, HFREJ, LFREJ not available on 9362C.

EXAMPLE (GPIB)

The following command 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_LEVEL_2, 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 with respect to the first acquired data point (displayed at the left-hand edge of the screen).

The command expects positive trigger delays 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 command 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_LEVEL_2, 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`☝, `EX5`☝}

<trig_level> : = *See instrument Operator's Manual, Appendix A, for specifications.*

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 instruments.

<trig_source> : = EXT10 not on LC564 or LC584 Series models.

<trig_source> : = EXT5 only on LC564 and LC584 Series models.

EXAMPLE (GPIB)

The following code 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_LEVEL_2, TRIG_MODE, TRIG_SELECT, TRIG_SLOPE, TRIG_WINDOW

ACQUISITION — TRIG_LEVEL_2, TRLV2

Command/Query

DESCRIPTION

The TRIG_LEVEL_2 command adjusts the level of the second trigger threshold reference, used in advanced SMART Trigger modes such as Runt and Slew Rate.

The response to the TRIG_LEVEL_2? query returns the current level of the second trigger.

COMMAND SYNTAX

<trig_source> : `Trigger_LeVel_2` <trig_level>

<trig_source> : = {`C1, C2, C3, C4, 2, EX, EX5`}

<trig_level> : = *See Operator's Manual, Appendix A, for specifications.*

QUERY SYNTAX

`Trigger_LeVel_2?`

RESPONSE FORMAT

<trig_source> : `Trigger_LeVel_2` < trig_level >

AVAILABILITY

Available only on LC564 and LC584A Series models.

EXAMPLE (GPIB)

The following instruction sets the trigger level of channel 2 to 1.5 V:

```
CMD$="C2:TRLV2 1.5V": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_LEVEL, TRIG_COUPLING, TRIG DELAY, TRIG_LEVEL, 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_LEVEL_2, TRIG_PATTERN, 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, and Channel 3 and Channel 4 on four-channel models), 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.

Notation			
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>

AVAILABILITY

Only available on 9350C, 9354C, 9362C, 9370C, 9374C, 9384C and LC Series models.

<source> : {C3, C4} only available on four-channel instruments.

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 may 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	RUNT	Runt
EDGE	Edge	SLEW	Slew Rate
EV	Event	SNG	Single source
GLIT	Glitch	SQ	State-Qualified
HT	Hold type	SR	Source
HV	Hold value	STD	§Standard (Edge Trigger)
HV2	Second hold value	TEQ	Edge-Qualified
IL	Interval larger	TEQ1	Qualified First
INTV	Interval	TI	Time
IS	Interval smaller	TL	Time within
I2	Interval-width window	TV**	TV
OFF	No hold-off on wait	CHAR	Characteristics
PA	Pattern	FLD	Field
PL	Pulse larger	FLDC	Field count
PS	Pulse smaller	ILAC	Interlace
P2	Pulse-width window	LINE	Line
QL	Qualifier	LPIC	Lines per picture

§ Note: HT and HV do not apply to the Standard Trigger.

*** TV Trigger has its own particular command syntax, using the notation listed below it on this table.*

COMMAND SYNTAX
For all but TV Trigger

TRig_SElect <trig_type>,SR,<source>,QL,<source>,
HT,<hold_type>,HV,<hold_value>,HV2,<hold value>
<trig_type> : = {DROP, EDGE, GLIT, INTV, PA, STD, SNG, SQ, TEQ, TEQ1, RUNT, SLEW}
<source> : = {C1, C2, C3☝, C4☝, LINE, EX, EX10☝, EX5☝, PA}
<hold_type> : = {TI, TL, EV, PS, PL, IS, IL, P2, I2, OFF}
<hold_value> : = *See instrument 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>

Note: HV2 only returned if <hold_type> is P2 or I2

☝ **AVAILABILITY**

<source> : {C3, C4} only available on four-channel instruments.

<source> : = EXT10 not on LC564 or LC584 Series models.

<source> : = EXT5 only on LC564 and LC584 Series models.

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 TV,SR,<source>,FLDC,<field_count>,FLD,<field>,
CHAR,<characteristics>,LPIC,<lpic>,ILAC,<ilace>,LINE,<line>

<trig_type> : = {TV☝}

<source> : = {C1, C2, C3☝,C4☝,NE, EX, EX10☝, EX5☝}
<field_count> : = {1, 2, 4, 8}
<field> : = 1 to field_count
<characteristics> : = {NTSC, PALSEC, CUST50, CUST60}
<lpic> : = 1 to 1500
<ilace> : = {1, 2, 4, 8}
<line> : = 1 to 1500 or 0 for any line

Note: 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 TV,SR,<source>,FLDC,<field_count>,FLD,<field>,
CHAR,<characteristic>,LINE,<line>

☝ **AVAILABILITY**

<source> : = {C3, C4} only on four-channel instruments.

<source> : = EXT10 not on LC564 or LC584 Series models.

<source> : = EXT5 only on LC564 and LC584 Series models.

<trig_type> : = TV not available on model 9362C.

EXAMPLE (GPIB)

The following sets up the trigger system to trigger on the 3rd field, line 17, of the 8-field PAL/SECAM TV signal applied to the external input.

```
CMD$="TRSE TV,SR,EX,FLDC,8,FLD,3,CHAR,PALSEC,
LINE,17": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

TRIG_COUPLING, TRIG_DELAY, TRIG_LEVEL, TRIG_LEVEL_2, TRIG_PATTERN, 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 70*) 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`☝, `EX5`☝}

<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.

<trig_source> : = EXT10 not on LC564 or LC584 Series models.

<trig_source> : = EXT5 only on LC564 and LC584 Series models.

<trig_slope> : = WINDOW only available on certain models. And not available when the trigger source is set to LINE.

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_LEVEL_2, TRIG_PATTERN, 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>`

AVAILABILITY

Not available on 9350C, 9354C, 9370C, 9374C, 9384C or 9362C Series oscilloscopes, nor on color models other than those in the LC564 and LC584 Series.

EXAMPLE

The following command 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_LEVEL_2, TRIG_PATTERN, 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 100*). 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 instruction 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 command 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 153*) is set if an out-of-range value is entered.

Note: 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, Appendix A, 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 command 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 instrument, as the oscilloscope 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 the instrument from analyzing new commands until the oscilloscope has completed the current acquisition.

COMMAND SYNTAX

`WAIT`

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 instrument to evaluate the maximum data value in the Channel 1 waveform.

RELATED COMMANDS

*TRG

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.

The WAVEFORM command stores an external waveform back into the oscilloscope's internal memory. A waveform consists of several distinct entities:

1. the descriptor (DESC)
2. the user text (TEXT)
3. the time (TIME) descriptor
4. the data (DAT1) block, and, optionally
5. a second block of data (DAT2).

For further information on the structure of the waveform refer to Chapter 4.

Note: Only complete waveforms queried with "WAVEFORM? ALL" can be restored into the oscilloscope.

The WAVEFORM? query instructs the oscilloscope to transmit a waveform to the controller. The entities may 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 command, the last COMM_ORDER command, and the last COMM_FORMAT command.

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`}

Note: If no parameter is given ALL will be assumed.

RESPONSE FORMAT

<trace> : `WaveForm` <block>,<waveform_data_block>

Note: It may be convenient to disable the response header if the waveform is to be restored. Refer to command COMM_HEADER 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 instrument, 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 1: After power-on, all values are set to 0 (i.e. entire waveforms will be transmitted without sparsing).

Note 2: Parameters are grouped in pairs. The first of the pair names the variable to be modified, whilst the second gives the new value to be assigned. Pairs may be given in any order and may be 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 command specifies that every 3rd data point (SP=3) starting at address 200 should 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 instruments.

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-axis and/or the 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 may 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 command 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 the user 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(s).

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 1: The suffix DIV is optional.

Note 2: Parameters are grouped in pairs. The first of the pair names the cursor to be modified, whilst the second indicates its new value. Pairs may be given in any order and may be restricted to those items to be changed.

QUERY SYNTAX

XY_Cursor_Set? [<cursor,...<cursor>]

<cursor> : = {XABS, XREF, XDIF, YABS, YREF, YDIF, ALL}

Note: If <cursor> is not specified, ALL will be assumed.

RESPONSE FORMAT

```
XY_Cursor_Set
<cursor>,<position>[,<cursor>,<position>...,
<cursor>,<position>]
```

EXAMPLE (GPIB)

The following command 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	DY/DX
<cursor type>_PROD	DY*DX
<cursor type>_ANGLE	arc tan(DY/DX)
<cursor type>_RADIUS	sqrt(DX*DX + DY*DY)

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 enables or disables the XY display mode. When off, the scope is 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 the XY display ON:

```
CMDS$="XYDS ON": CALL IBWRT(SCOPE%,CMDS$)
```

RELATED COMMANDS

GRID

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 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 persistence data maps.

COMMAND SYNTAX

`XY_SAturation` <trace>, <value> [<trace>, <value>]

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

<value> : = 0 to 100 PCT

Note: The suffix PCT is optional.

QUERY SYNTAX

`XY_SAturation?`

RESPONSE FORMAT

`XY_SAturation` <trace>, <value>

☝**AVAILABILITY**

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

EXAMPLE (GPIB)

The following sets the saturation level of the XY persistence data map for channel 3 to be 60%, i.e. 60% of the data points will be displayed with the color spectrum, with the remaining 40% saturated in the brightest color:

```
CMD$="XYSA C3,60": CALL IBWRT(SCOPE%,CMD$)
```

RELATED COMMANDS

PERSIST_SAT

Example 1

Use of 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 user-entered:

```
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

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 the controller is equipped with a National Instruments GPIB interface card. All commands listed in the *System Commands* section 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 disk if 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>
100    CLS
110    PRINT "Control of the 9300 via GPIB and IBM PC"
115    PRINT ""
120    PRINT "Options :    EX to exit    LC local mode"
125    PRINT "       ST store data RC recall data"
130    PRINT ""
140    LINE INPUT "GPIB-address of oscilloscope (1...16)? :",ADDR$
145    DEV$ = "DEV" + ADDR$
150    CALL IBFIND(DEV$,SCOPE%)
155    IF SCOPE% < 0 THEN GOTO 830
160    TMO% = 10  'timeout = 300 msec (rather than default 10 sec)
165    CALL IBTMO(SCOPE%,TMO%)
170    '
200    LOOP% = 1
205    WHILE LOOP%
210           LINE INPUT "Enter command (EX --> Exit) : ",CMD$
220           IF CMD$ = "ex" OR CMD$ = "EX" THEN LOOP% = 0 : GOTO 310
230           IF CMD$ = "st" OR CMD$ = "ST" THEN GOSUB 600 : GOTO 300
240           IF CMD$ = "rc" OR CMD$ = "RC" THEN GOSUB 700 : GOTO 300
250           IF 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$="WFSU NP,0,SP,0,FP,0,SN,0; CHDR SHORT"
640     CALL IBWRT(SCOPE%,CMD$)
645     CMD$=TRACE$+":WF?"
650     CALL IBWRT(SCOPE%,CMD$)
660     CALL IBRD(SCOPE%,RD1$)       'Discard first 3 chars of response
665     CALL IBRDF(SCOPE%,FILE$)
670     IF IBSTA% < 0 THEN GOTO 840
675     PRINT ""
680     RETURN
700     '
705     'SUBROUTINE RECALL_DATA
710     '
715     LINE INPUT "Specify target memory (M1...M4):",MEM$
720     LINE INPUT "Enter filename : ",FILE$
730     CMD$=MEM$+":"
735     CALL IBWRT(SCOPE%,CMD$)
740     CALL IBWRTF(SCOPE%,FILE$)
745     IF IBSTA% < 0 THEN GOTO 840
```

```
750     PRINT ""
755     RETURN
800     '
810     'ERROR HANDLER
820     '
830     PRINT "IBFIND ERROR"
835     END
840     PRINT "GPIB ERROR -- IBERR: ";IBERR%;"IBSTA: ";HEX$(IBSTA%)
845     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

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>
100    CLS
110    PRINT "Control of the 9300 (address 4) via GPIB and IBM PC"
115    PRINT "": PRINT "Options :  EX to exit        LC local mode"
120    PRINT "            ST store data     RC recall data": PRINT""
125    LOOP=1
130    CMD1$ = "?_@$" 'Unlisten, Untalk, Board talker, Device listener
135    CMD2$ = "?_ D" 'Unlisten, Untalk, Board listener, Device talker
140    BDNAME$= "GPIB0": CALL IBFIND(BDNAME$,BRD0%)
145    IF BRD0% < 0 THEN GOTO 420
150    CALL IBSIC(BRD0%): IF IBSTA% < 0 THEN GOTO 425
155    WHILE LOOP
160           LINE INPUT "Enter command (EX --> Exit) : ",CMD$
165           V% = 1: CALL IBSRE(BRD0%,V%)
170           IF CMD$ = "ex" OR CMD$ = "EX" THEN LOOP = FALSE: GOTO 205
175           IF CMD$ = "st" OR CMD$ = "ST" THEN GOSUB 285: GOTO 200
180           IF CMD$ = "rc" OR CMD$ = "RC" THEN GOSUB 365: GOTO 200
185           IF CMD$ = "lc" OR CMD$ = "LC" THEN  GOSUB 240: GOTO 200
190           IF CMD$ = "" THEN GOTO 200
195           CALL IBCMD(BRD0%,CMD1$): CALL IBWRT(BRD0%,CMD$): GOSUB 270
200    WEND
205    CALL IBSIC(BRD0%): V%=0: CALL IBSRE(BRD0%,V%)
210    CALL IBSIC(BRD0%)
215    END
220    '
230    'LOCAL MODE
235    '
240    V% = 0: CALL IBSRE(BRD0%,V%): PRINT ""
245    RETURN
250    '
```

```
260    'SUBROUTINE GET_DATA
265    '
270    CALL IBCMD(BRD0%,CMD2$): CALL IBRD(BRD0%,RD$): I=IBCNT%
275    FOR J=1 TO I: PRINT MID$(RD$,J,1);: NEXT J: PRINT ""
280    RETURN
285    '
290    'SUBROUTINE STORE_DATA
295    '
300    RD1$=SPACE$(3)
305    LINE INPUT "Specify trace (TA...TD,M1...M4,C1...C4):  ",TRACE$
310    LINE INPUT "Enter filename : ",FILE$
315    CALL IBCMD(BRD0%,CMD1$)
320    CMD$="WFSU NP,0,SP,0,FP,0,SN,0;CHDR SHORT"
321    CALL IBWRT(BRD0%,CMD$)
325    CMD$=TRACE$+":WF?": CALL IBWRT(BRD0%,CMD$)
330    CALL IBCMD(BRD0%,CMD2$): CALL IBRD(BRD0%,RD1$)
335    CALL IBRDF(BRD0%,FILE$)
340    IF IBSTA% < 0 THEN GOTO 430
345    PRINT ""
350    RETURN
355    '
360    'SUBROUTINE RECALL_DATA
365    '
370    LINE INPUT "Specify target memory (M1...M4): ",MEM$
375    LINE INPUT "Enter filename : ",FILE$
380    CALL IBCMD(BRD0%,CMD1$)
385    CMD$=MEM$+":": CALL IBWRT(BRD0%,CMD$)
390    CALL IBWRTF(BRD0%,FILE$)
395    IF IBSTA% < 0 THEN GOTO 430
400    PRINT ""
405    RETURN
410    '
415    'ERROR HANDLER
420    '
425    PRINT "IBFIND ERROR": STOP
430    PRINT "GPIB ERROR -- IBERR : ";IBERR%;"IBSTA : ";HEX$(IBSTA%)
435    STOP
440    END
```

***Note:** The Template also describes an array named DUAL. This is simply a way to allow the INSPECT? command to examine the two data arrays together.*

Waveform Template

This template is the instrument's response to a command of the form "TMPL?":

```
/00
000000              LECROY_2_2:  TEMPLATE
                    8 66 111
;
; Explanation of the formats of waveforms and their descriptors on the
; LeCroy Digital Oscilloscopes,
;     Software Release 44.1.1.1, 94/04/18.
;
; 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         8-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 :
;
< 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>           RESERVED3: word
<294>           RESERVED4: word           ; 2 expansion entries
;
<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
;
;=============================================================================
;
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
;
;===============================================================================
;
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
;
;===============================================================================
;
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
```

Index

E

F

G

H

I

K

L

R

S

T