# Programmer Manual

Tektronix

## TDS 340A, TDS 360 & TDS 380
## Digital Real-Time Oscilloscopes

## 070-9442-02

This document applies to firmware version 1.05
and above.

Printed in the U.S.A.

Tektronix, Inc., P.O. Box 1000, Wilsonville, OR 97070–1000

## WARRANTY

Tektronix warrants that the products that it manufactures and sells will be free from defects in materials and workmanship for a period of three (3) years from the date of shipment. If a product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; c) to repair any damage or malfunction caused by the use of non-Tektronix supplies; or d) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

**THIS WARRANTY IS GIVEN BY TEKTRONIX IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES**.

# Table of Contents

## Status and Events



## Examples



## Appendices



## Glossary and Index

# List of Figures

# List of Tables

# General Safety Summary

Review the following safety precautions to avoid injury and prevent damage to this product or any products connected to it. To avoid potential hazards, use the product only as specified.

*Only qualified personnel should perform service procedures.*

## Injury Precautions

**Use Proper Power Cord**

To avoid fire hazard, use only the power cord specified for this product.

**Avoid Electric Overload**

To avoid electric shock or fire hazard, do not apply a voltage to a terminal that is outside the range specified for that terminal.

**Avoid Electric Shock**

To avoid injury or loss of life, do not connect or disconnect probes or test leads while they are connected to a voltage source.

**Ground the Product**

This product is grounded through the grounding conductor of the power cord. To avoid electric shock, the grounding conductor must be connected to earth ground. Before making connections to the input or output terminals of the product, ensure that the product is properly grounded.

**Do Not Operate Without Covers**

To avoid electric shock or fire hazard, do not operate this product with covers or panels removed.

**Use Proper Fuse**

To avoid fire hazard, use only the fuse type and rating specified for this product.

**Do Not Operate in Wet/Damp Conditions**

To avoid electric shock, do not operate this product in wet or damp conditions.

**Do Not Operate in an Explosive Atmosphere**

To avoid injury or fire hazard, do not operate this product in an explosive atmosphere.

## Product Damage Precautions

**Use Proper Power Source**

Do not operate this product from a power source that applies more than the voltage specified.

**Provide Proper Ventilation**

To prevent product overheating, provide proper ventilation.

**Do Not Operate With Suspected Failures**

If you suspect there is damage to this product, have it inspected by qualified service personnel.

## Safety Terms and Symbols

**Terms in This Manual**

These terms may appear in this manual:

***WARNING.*** *Warning statements identify conditions or practices that could result in injury or loss of life.*

***CAUTION.*** *Caution statements identify conditions or practices that could result in damage to this product or other property.*

**Terms on the Product**

These terms may appear on the product:

DANGER indicates an injury hazard immediately accessible as you read the marking.

WARNING indicates an injury hazard not immediately accessible as you read the marking.

CAUTION indicates a hazard to property including the product.

**Symbols on the Product**

The following symbols may appear on the product:

DANGER
High Voltage

Protective Ground
(Earth) Terminal

ATTENTION
Refer to Manual

Double
Insulated

## Certifications and Compliances

**CSA Certified Power Cords**

CSA Certification includes the products and power cords appropriate for use in the North America power network. All other power cords supplied are approved for the country of use.

**Compliances**

Consult the product specifications for Overvoltage Category and IEC Classifications.

# Preface

This is the Programmer Manual for the TDS 340A, TDS 360, and TDS 380 Two Channel Digital Real-Time Oscilloscopes. This manual provides information on operating your oscilloscope using the General Purpose Interface Bus (GPIB) and the RS-232 interface.

## Related Manuals

Following is additional documentation for the oscilloscopes.

- *TDS 340A, TDS 360 & TDS 380 Instruction Manual*
- *TDS 340A, TDS 360 & TDS 380 Reference*
- *The XYZs of Analog and Digital Oscilloscopes*

# Getting Started

This chapter covers the following topics:

- *Connector Locations* shows the RS-232 and GPIB connector locations on the pack panel of the TDS 300 Series oscilloscope.
- *Comparing GPIB and RS-232* compares the characteristics of the GPIB and RS-232 interfaces.
- *Setting Up GPIB Remote Communications* describes setting up for GPIB remote control, including connecting the oscilloscope and setting the appropriate front-panel controls.
- *Setting Up RS-232 Remote Communications* describes setting up for RS-232 remote control, including connecting the oscilloscope and setting the appropriate front-panel controls.
- *Setting Up the Hardcopy Port* describes how to select the port to which you will send screen capture data.
- *Programming Model* describes how the programming interface differs from the front-panel user interface.
- *Tutorial* includes a simple programming example.

For general information about operating the oscilloscope, refer to the *TDS 340A, TDS 360 & TDS 380 User Manual*.

## Connector Locations

All remote-communications connectors are located on the back panel of the TDS 300 Series oscilloscope. Figure 1–1 shows the locations of the connectors.

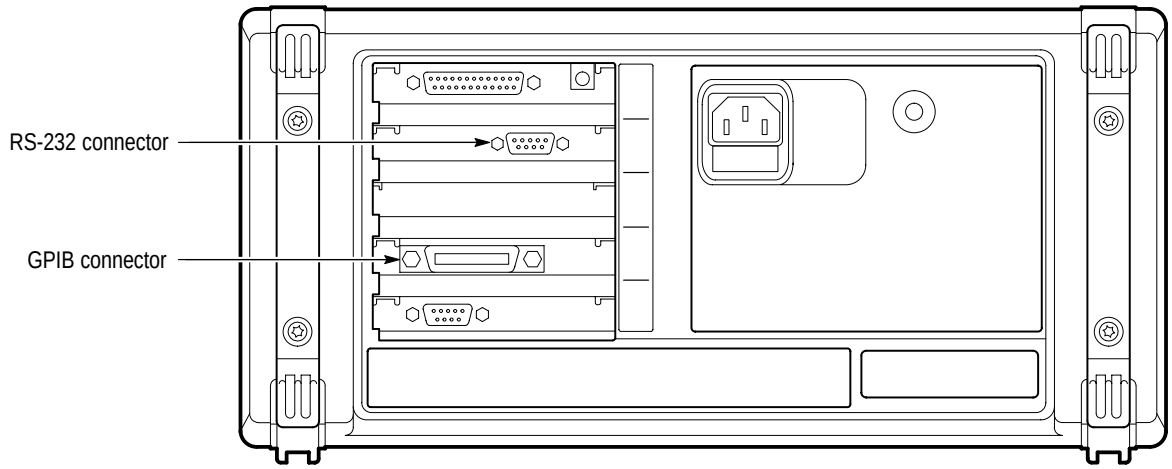



**Figure 1–1: Location of RS-232 and GPIB connectors**

## Comparing GPIB and RS-232

Your system hardware may support both GPIB and RS-232 interfaces. You should select the interface that best meets your requirements. The GPIB interface is an eight-bit parallel bus that provides high-speed data transfer and multiple-instrument control. In contrast, the RS-232 interface is a slower serial data bus for single instrument control, but is easy to connect to with low-cost controllers. Table 1–1 provides a more in-depth comparison of the GPIB and RS-232 interfaces.

**Table 1–1: Comparison of GPIB and RS-232 interfaces**

| Operating Attribute | GPIB | RS-232 |
|---|---|---|
| Cable | IEEE-488 Std. | 9-wire |
| Data flow control | Hardware, 3-wire handshake | Flagging: soft (XON/XOFF), hard (RTS/CTS) |
| Data format | 8-bit parallel | 8-bit serial |
| Interface control | Operator low-level control message | None |
| Interface messages | Most IEEE-488 Std. | Device clear using a break signal |

**Table 1–1: Comparison of GPIB and RS-232 interfaces (Cont.)**

| Operating Attribute | GPIB | RS-232 |
|---|---|---|
| Interrupts reported | Service requests, status and event code | None, must be polled for status |
| Message termination (Receive) | Hardware EOL, software LF, or both | Software CR, LF, CRLF, LFCR |
| Message termination (Transmit) | Hardware EOL, software LF | Software CR, LF, CRLF, LFCR |
| Timing | Asynchronous | Asynchronous |
| Transmission path length (max) | ≤ 2 meters between devices; ≤ meters total cabling for GPIB system | ≤ 15 meters |
| Speed | 200kBytes/sec | 38,400 bits/sec |
| System environment | Multiple devices (≤ 15) | Single terminal (point-to-point connection) |

## Setting Up GPIB Remote Communications

The oscilloscope has a 24-pin GPIB connector on its rear panel, as shown in Figure 1–1. This connector has a D-type shell and conforms to IEEE Std. 488.1-1987.

Attach an IEEE Std 488.1-1987 GPIB cable (available from Tektronix as part number 012-0991-00) to this connector. If needed, you can stack GPIB connectors as shown in Figure 1–2.

### GPIB Requirements

Follow these rules when you connect your oscilloscope to a GPIB network:

- Assign a unique device address to each device on the bus. No two devices can share the same device address.
- Do not connect more than 15 devices to any one bus.
- Connect one device for every 2 meters (6 feet) of cable used.
- Do not use more than 20 meters (65 feet) of cable to connect devices.
- Turn on at least two-thirds of the devices on the network while using the network.
- Connect the devices on the network in a star or linear configuration as shown in Figure 1–3. Do not use loop or parallel configurations.

*Appendix C: Interface Specifications* gives more information on the GPIB configuration of the oscilloscope.

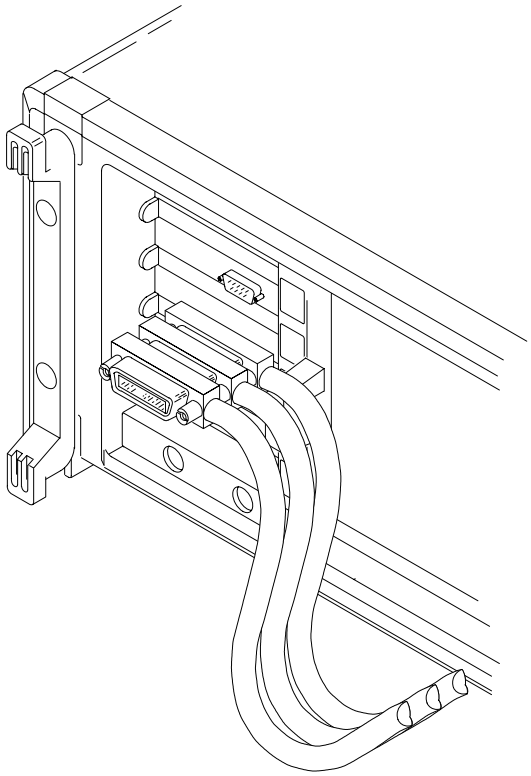


**Figure 1–2: How to stack GPIB connectors**

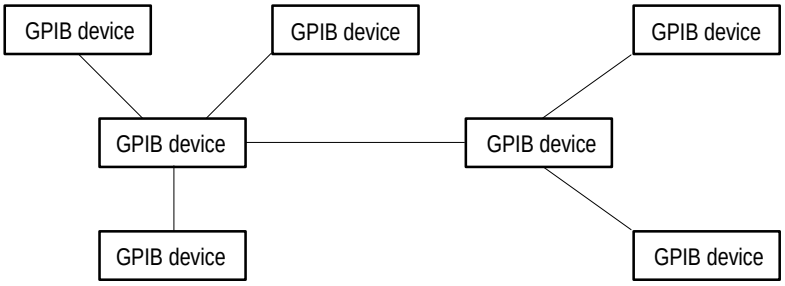



**Figure 1–3: Typical GPIB network configurations**

## Setting the GPIB Parameters

You need to set the GPIB parameters of the oscilloscope to match the configuration of the bus. Once you have set these parameters, you can control the oscilloscope through the GPIB interface.

1. Press the **UTILITY** button to display the Utility menu.
2. Press the **System** button in the main menu until it highlights the **I/O** selection in the pop-up menu.
3. Press the **GPIB Talk/Listen** main-menu button to display the **GPIB Configuration** side menu. See Figure 1–4.
4. Select the **Talk/Listen Address** side menu button.
5. Set the GPIB address using the general purpose knob.

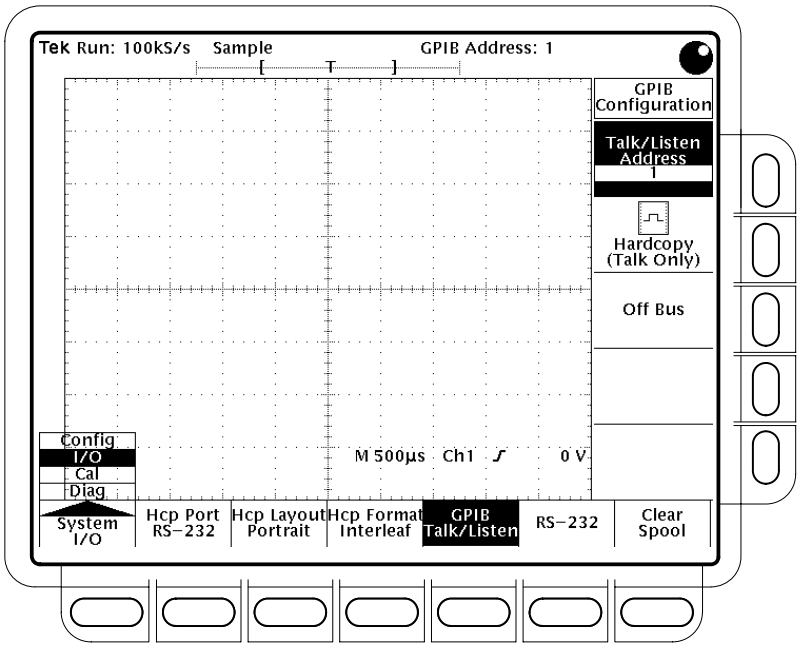



**Figure 1–4: Selecting the GPIB address in the GPIB configuration side menu**

The oscilloscope is now set up for bidirectional communication with your GPIB controller. If you want to isolate the oscilloscope from the bus, press the **Off Bus** side menu button. This disables all communication with the controller.

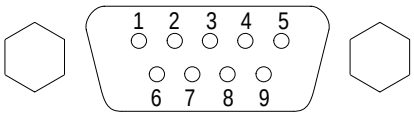
If you want to enter a special mode of operation to communicate directly with non-488.2 hard copy devices, press the **Hardcopy** side menu button to have the oscilloscope send hard copy information only when you press the **HARDCOPY** button (and accept a HARDCOPY ABORT command).

## Setting Up RS-232 Remote Communications

The TDS 300 Series oscilloscope has a 9-pin D-type shell RS-232 connector located on the left side of the rear panel, as shown in Figure 1–1. The RS-232 interface provides a point-to-point connection between the oscilloscope and equipment such as a computer or terminal. This section tells how to connect and set up the oscilloscope for communication over the RS-232 interface.

### Connecting to an RS-232 Device

The RS-232 standard defines two device types: Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). The TDS 300 Series oscilloscope is a DTE device. In standard usage, DTE devices have a male connector, and DCE devices have a female connector. You should use a straight-through female-to-male cable of less than 50 feet for a local DTE-to-DCE connection. Figure 1–5 shows the 9-pin connector with its pin number assignments.




| | | |
|---|---|---|
| 1 | No connection | |
| 2 | Receive data (RxD) | (input) |
| 3 | Transmit data (TxD) | (output) |
| 4 | Data terminal ready (DTR) | (output) |
| 5 | Signal ground (GND) | |
| 6 | Data set ready (DSR) | (input) |
| 7 | Request to send (RTS) | (output) |
| 8 | Clear to send (CTS) | (input) |
| 9 | No connection | |

**Figure 1–5: The RS-232 connector pin assignments**

In terms of the connector and the way the oscilloscope uses the signal lines, the oscilloscope behaves just like a PC/AT COM port. Table 1–2 lists cables you can use to connect the oscilloscope to other devices.

**Table 1–2: RS-232 adapter cables**

| Tektronix Part Number | Cable Type | Use |
|---|---|---|
| 012–1379–00 | 9-pin female to 9-pin female, null modem | PC/AT or laptop |
| 012–1380–00 | 9-pin female to 25-pin female, null modem | Old style PC with 25-pin connector |
| 012–1298–00 | 9-pin female to 25-pin male, null modem | Serial printers, such as an HP Deskjet and Sun workstations |
| 012–1241–00 | 9-pin female to 25-pin male, modem | Telephone modem |

Follow these guidelines when connecting the oscilloscope to another RS-232 device:

- Do not connect the output line of one DTE device to the output line of another DTE device.
- Connect the signal ground of the oscilloscope to the signal ground of the external device.
- Connect the chassis ground of the oscilloscope to the chassis ground of the external device.

## Setting the RS-232 Parameters

To set the RS-232 parameters, do the following steps from the oscilloscope front panel. After these parameters are set, the RS-232 interface is ready to operate.

1. Press the **UTILITY** button to display the Utility menu.
2. Press the **System** main-menu button until **I/O** is selected in the pop-up menu.
3. Press the **RS-232** main-menu button to display the RS-232 side menu (see Figure 1–6).

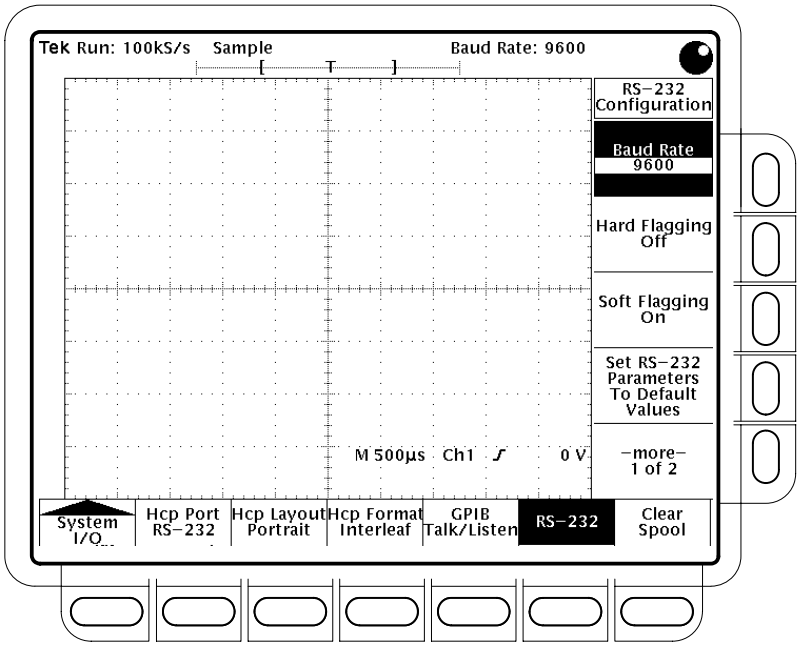



**Figure 1–6: RS-232 parameter settings**

You can set the following parameters:

- Baud Rate — sets the data transmission rate. You can set rates of 300, 600, 1200, 2400, 4800, 9600, 19200, or 38400 baud.
- Hard Flagging — sets hard flagging (RTS/CTS) on or off. Flagging controls the flow of data between devices. When both hard and soft flagging are off, the oscilloscope does not use or recognize any flagging. Use hard flagging for binary data transfers.
- Soft Flagging — sets soft flagging (XON/XOFF) on or off. Hard flagging is the preferred method of controlling the flow of data between devices. When both hard and soft flagging are off, the oscilloscope does not use or recognize any flagging. You should not use soft flagging with binary data transfer since the data may contain XON and XOFF characters.
- Set RS-232 Parameters to Default Values — sets default values for RS-232 parameters (for a list of default settings see Table 1–3).
- EOL — sets the end of line terminator sent by the oscilloscope. You can set CR, LF, CRLF, or LFCR (for more information on line terminators see page 2–6).
- Parity — adds an error check bit (ninth bit) to each character. You can set the error bit for either None, Even, or Odd parity. When the parity setting is odd or even, the oscilloscope generates the selected parity on output and checks incoming data for the selected parity. When the parity setting is none, there is no parity bit.
- Stop Bits — sets the number of stop bits sent after each character. You can set 1 or 2 stop bits.
- Delay — sets the delay time before responding to a query. You can set times from 0 to 60 seconds in 100 ms increments.

**4.** Press, in turn, each side menu button until the desired parameter setting is displayed in the side menu, or press **Set RS-232 Parameters to Default Values**, if the default settings are appropriate. Table 1–3 lists the default RS-232 settings.

**Table 1–3: RS-232 default settings**

| RS-232 Parameter | Default Setting |
|---|---|
| Baud Rate | 9600 |
| Hard Flagging | On |
| Soft Flagging | Off |
| EOL | LF |
| Parity | None |
| Stop Bits | 1 |
| Delay | 0 s |

## RS-232 Conventions

There are processing conventions that are specific to the RS-232 interface. The next sections discuss the following conventions:

- Transferring binary data
- Processing break signals
- Reporting RS-232 I/O errors
- Checking command status

**Transferring Binary Data.** When using the RS-232 port to transfer binary data to the oscilloscope, note the following points:

- Using RTS/CTS (hard) flagging guarantees no data loss.
- All eight bits of binary data contain meaningful information. To make sure that all eight bits are received or transmitted, configure the RS-232 device that is connected to the oscilloscope to receive and transmit eight-bit characters (set the RS-232 word length to eight bits).

**Processing Break Signals.** When the oscilloscope senses a break signal on the RS-232 port, it returns DCL followed by the end of line terminator. Internally, the oscilloscope acts as if it received a GPIB <DCL> command, causing the oscilloscope to flush input and output buffers and then wait for a new command. Break signals do not change oscilloscope settings or stored data and do not interrupt front-panel operation or nonprogrammable functions.

If a break signal is sent in the middle of a character stream, several characters immediately preceding or following the break may be lost. The controller should wait until it receives the DCL and the end of line terminator string before sending more characters.

**Reporting RS-232 I/O Errors.** Errors are reported when there is a problem with parity, framing, or input/output buffer overruns. To report errors, the oscilloscope posts an event code (refer to Section 3, *Status and Events* on page 3–1). When an error occurs, the oscilloscope discards all input and output and waits for a new command. A count of these errors since last power on is included in the error log (in the Diag System of the Utility menu).

You can use the RS232 Line Snapshot entry of the error log to help establish an RS-232 connection. The snapshot reports whether the oscilloscope is waiting to receive a control-Q (yes/no), the state of the hardware CTS line (high/low), and whether characters have been received (yes/no).

Use the following statements to help you interpret the status reported in the error log:

- If soft flagging is on and if Waiting For ^Q is Yes, the oscilloscope must receive an XON character before it will transmit any more data.
- If hard flagging is on and CTS is Low, the oscilloscope will not transmit any data.
- If hard flagging is off, you should ignore the value of CTS since the oscilloscope ignores it.
- If Chars Rcvd is Yes, the oscilloscope has received at least one character since the last power on.

The RS232 Errors line of the error log lists the number of parity, framing, and overrun errors since the last power on.

**Checking Command Status.** If you want to check the status of each command sent, you can append a *STB? query after every command and read the response string.

## RS-232 Troubleshooting

If the oscilloscope and the personal computer or printer have trouble communicating, use the following steps to correct the problem:

1. Verify that you are using the correct RS-232 cable. Determine whether your configuration requires a null-modem connection (where transmit/receive and control lines are switched) or a straight-through RS-232 connection. Refer to Table 1–2 for information about RS-232 cables.
2. Verify that the RS-232 cable is firmly connected to both the oscilloscope and the correct port on your personal computer or printer. Verify that your printer or the program on the personal computer is using the correct port. Try your program or printer again.
3. Verify that the oscilloscope settings match the settings used by your printer or the program on your personal computer. Start by choosing Set RS-232 Parameters to Defaults (in the RS-232 System of the Utility menu). Then, change only those menu items that you know need to be changed, such as the baud rate. Try your printer or computer program again.
4. If you are trying to control the oscilloscope using a personal computer or other computer, look at the diagnostic error log and examine the RS232 Line Snapshot and the RS232 Errors. The RS232 Line Snapshot and the RS232 Errors will not change while you are viewing them. They are reset when the power is turned on. Use Table 1–4 to troubleshoot your setup.

**Table 1–4: RS-232 troubleshooting**

| Symptom | Possible Causes |
|---|---|
| Your personal computer program tried to send characters to the oscilloscope, but the error log displays Chars Rcvd: No. | Your RS-232 cable may be wired as a modem instead of a null modem. If you are attempting to use a telephone modem, the cable may be wired as a null modem instead of a modem. |
| The oscilloscope error log displays Framing errors. | There is a baud rate mismatch between the oscilloscope and the personal computer.<br>There is a data bits mismatch between the oscilloscope and the personal computer (The oscilloscope expects 8-bit data).<br>There is a parity mismatch between the oscilloscope and the personal computer. |
| The oscilloscope error log displays Parity errors. | There is a parity mismatch between the oscilloscope and the personal computer. |
| The oscilloscope error log displays Overrun errors. | Flagging is not being used correctly by the oscilloscope or the personal computer: they are using different types of flagging. |

**Table 1–4: RS-232 troubleshooting (Cont.)**

| Symptom | Possible Causes |
|---|---|
| Transmissions are incomplete, or the oscilloscope does not process all commands from the personal computer. | Flagging is not being used correctly by the oscilloscope or the personal computer: they are using different types of flagging.<br><br>There is an EOL terminator mismatch between the oscilloscope and the personal computer. |
| The oscilloscope error log displays Waiting for ^Q: Yes. | The oscilloscope is using soft flagging, so verify that the personal computer is also using soft flagging, Also, verify that the personal computer is not sending binary data. Binary data may contain ^S characters which cause transmissions to stop. |
| Soft flagging is being used, and transmissions stop. | Verify that both the personal computer and the oscilloscope are not sending binary data. Binary data may contain ^S characters which cause transmissions to stop. |
| The oscilloscope error log displays CTS: Low, and the oscilloscope is using hard flagging. | Verify that the RS-232 cable is the recommended cable. Some cables may be wired without the CTS or RTS lines which are used by hard flagging. Verify that the personal computer program is using CTS/RTS hard flagging. |
| After the personal computer program sends a BREAK, the first message fails. | Verify that the personal computer program is waiting for and reading the DCL and end of line terminator response sent by the oscilloscope. |

## Setting Up the Hardcopy Port

You can set the port to which the oscilloscope will send screen hard-copy data. To set the hard copy port, do the following steps:

1. Press the **UTILITY** button to display the Utility menu.
2. Press the **Hcp Port** main-menu button to display the Hard Copy Port side menu.
3. Select the **Centronics** side-menu button to send hard-copy data to the Centronics port, select **GPIB** to send hard copy data to the GPIB port, select **RS-232** to send hard copy data to the RS-232 port, or select **File** to send hard copy data to the floppy-disk drive. See Figure 1–7.

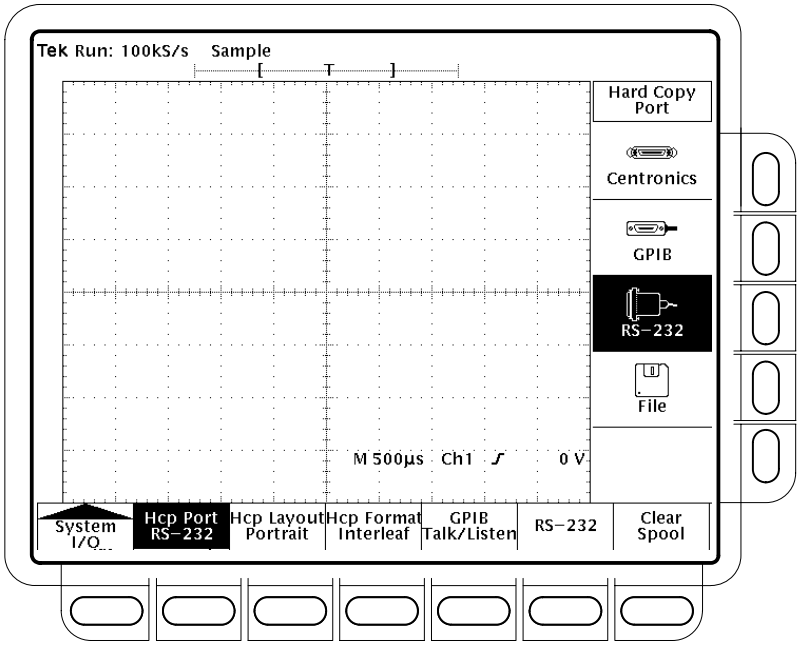



**Figure 1–7: RS-232 Hardcopy menu**

## Programming Model

Table 1–5 describes how the model for programming the oscilloscope differs from controlling the oscilloscope using the front panel.

**Table 1–5: Comparison of programming and front-panel models**

| Programming Model | Front-Panel Model |
|---|---|
| You can access all controls at all times. | You can only change the controls that are accessible in the current menu. |
| You have more control over some features than is available through the front panel. | You have less control over some features than is available through the programming interface. |

# Tutorial

This tutorial contains one simple example of operating the oscilloscope through the programming interface. This example verifies communication with the oscilloscope, acquires a signal, and then takes a frequency measurement. Refer to the chapter *Programming Examples* on page 4–1 for a description of the diskette, included with this manual, that contains additional programming examples.

The tutorial assumes you have already attached a PC to the oscilloscope and that the PC is executing a terminal-emulator program such as a Microsoft Windows Terminal for RS-232 or National Instrument's ibic program for GPIB. Be sure to set the communication settings in the terminal-emulator program to match those of the oscilloscope. Refer to Section 4 for more information on setting up a GPIB card.

1. Connect the oscilloscope probe to the channel 1 input BNC. Attach the probe tip and reference lead to the PROBE COMP connectors just above and to the left of the channel 1 BNC. The PROBE COMP signal is a square wave with a frequency of ≈1 kHz. Figure 1–8 shows how to hook up the probe to the oscilloscope.

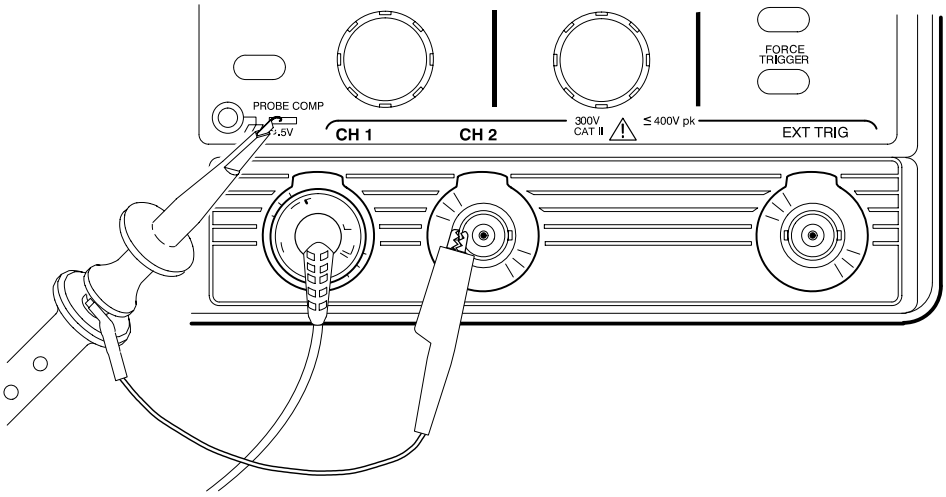



**Figure 1–8: Connecting to the PROBE COMP signal**

2. Press the **ON/OFF** button to turn on the oscilloscope. After a few seconds, you should see a window with the message Power-On self check PASSED.
3. Press the **CLEAR MENU** button to begin operation.
4. From your PC, send the query `ID?`. The oscilloscope responds with its identification string `ID TEK/TDS 340,CF:91.1CT,FV:v1.00` (or similar).

5. Send the command `FACTory` to reset the oscilloscope to the factory default settings. The factory default state is described in *Appendix D: Factory Setup*.
6. Send the command `AUTOSet EXECute` to have the scope automatically acquire the input signal.
7. Send the command `MEASUrement:IMMed:SOURCE CH1` to select measurements on channel 1.
8. Send the command `MEASUrement:IMMed:TYPe FREQuency` to set up the frequency measurement.
9. Send the query `MEASUrement:IMMed:VALue?` to request the measurement result. The oscilloscope will respond with a result similar to `1.0E3`, which is a frequency measurement of the PROBE COMP signal.

This completes the tutorial.

# Command Syntax

You can control the oscilloscope through the GPIB or RS-232 interface using a large group of commands and queries. This section describes the syntax these commands and queries use and the conventions the oscilloscope uses to process them. The commands and queries themselves are listed in the *Commands* section.

You transmit commands to the oscilloscope using the enhanced American Standard Code for Information Interchange (ASCII) character encoding. *Appendix A* contains a chart of the ASCII character set.

This manual uses the Backus-Naur Form (BNF) notation, shown in Table 2–1, to describe commands and queries.

**Table 2–1: BNF symbols and meanings**

| Symbol | Meaning |
|---|---|
| < > | Defined element |
| ::= | Is defined as |
| \| | Exclusive OR |
| { } | Group; one element is required |
| [ ] | Optional; can be omitted |
| . . . | Previous element(s) may be repeated |
| ( ) | Comment |

## Command and Query Structure

Commands consist of set commands and query commands (usually simply called commands and queries). Commands modify oscilloscope settings or tell the oscilloscope to perform a specific action. Queries cause the oscilloscope to return data and information about its status.

Most commands have both a set form and a query form. The query form of the command is the same as the set form but with a question mark on the end. For example, the set command `ACQuire:MODe` has a query form `ACQuire:MODe?`. Not all commands have both a set and a query form; some commands are set only and some are query only.

A command message is a command or query name, followed by any information the oscilloscope needs to execute the command or query. Command messages consist of five different element types, defined in Table 2–2 and shown in the example in Figure 2–1.

**Table 2–2: Command message elements**

| Symbol | Meaning |
|---|---|
| <Header> | The basic command name. If the header ends with a question mark, the command is a query. The header may begin with a colon (:) character; if the command is concatenated with other commands the beginning colon is required. The beginning colon can never be used with command headers beginning with a star (*). |
| <Mnemonic> | A header sub-function. Some command headers have only one mnemonic. If a command header has multiple mnemonics, they are always separated from each other by a colon (:) character. |
| <Argument> | A quantity, quality, restriction, or limit associated with the header. Not all commands have an argument, while other commands have multiple arguments. Arguments are separated from the header by a <Space>. Arguments are separated from each other by a <Comma>. |
| <Comma> | A single comma between arguments of multiple-argument commands. It may optionally have white space characters before and after the comma. |
| <Space> | A white space character between command header and argument. It may optionally consist of multiple white space characters. |

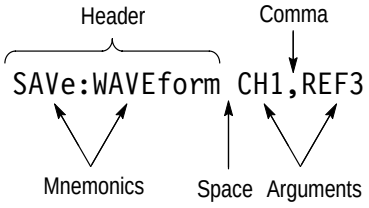



**Figure 2–1: Command message elements**

## Commands

Commands cause the oscilloscope to perform a specific function or change one of its settings. Commands have the structure:

- [:]<Header>[<Space><Argument>[<Comma><Argument>]...]

A command header is made up of one or more mnemonics arranged in a hierarchical or tree structure. The first mnemonic is the base or root of the tree and each subsequent mnemonic is a level or branch off of the previous one. Commands at a higher level in the tree may affect those at a lower level. The leading colon (:) always returns you to the base of the command tree.

## Queries

Queries cause the oscilloscope to return information about its status or settings. Queries have the structure:

- [:]<Header>?
- [:]<Header>?[<Space><Argument>[<Comma><Argument>]...]

You can specify a query command at any level within the command tree unless otherwise noted. These branch queries return information about all the mnemonics below the specified branch or level. For example, DISplay:INTENSITy:CONtrast? returns the intensity of the intensified zone of a waveform, while DISplay:INTENSITy? returns the intensity settings of all parts of the display.

## Headers in Query Responses

You can control whether the oscilloscope returns headers as part of the query response. Use the HEADer command to control this feature. If header is on, the oscilloscope returns command headers as part of the query and formats the query response as a valid set command. When header is off, the oscilloscope sends back only the values in the response. This may make it easier to parse and extract the information from the response. Table 2–3 shows the difference in responses.

**Table 2–3: Comparison of header off and on responses**

| Query | Header Off Response | Header On Responses |
|---|---|---|
| CURSor:VBArs:DELTa? | 1.064E-3 | :CURSor:VBArs:DELTa 1.064E-3 |
| ACQuire:NUMAVg? | 16 | :ACQUIRE:NUMAVG 16 |

## Clearing the Oscilloscope

To clear the Output Queue and reset the oscilloscope to accept a new command, use the Device Clear (DCL) GPIB command or the RS-232 BREAK signal.

## Command Entry

Follow these general rules when entering commands:

- You can enter commands in upper or lower case.
- You can precede any command with white space characters. White space characters include any combination of the ASCII control characters 00, 09 or 20 decimal.
- The oscilloscope ignores commands consisting of any combination of white space characters and line feeds.

### Abbreviating Commands

You can abbreviate many oscilloscope commands. These abbreviations are shown in capitals in the command's listing in the *Commands* section. For example, the command `ACQuire:NUMAvg` can be entered simply as `ACQ:NUMA` or `acq:numa`.

If you use the `HEADer` command to have command headers included as part of query responses, you can further control whether the returned headers are abbreviated or are full length. The `VERBose` command lets you control this.

### Concatenating Commands

You can concatenate any combination of set commands and queries using a semicolon (;). The oscilloscope executes concatenated commands in the order received.

When concatenating commands and queries you must follow these rules.

1. Completely different headers must be separated by both a semicolon and by the beginning colon on all commands but the first. For example, the commands `TRIGger:MODe NORMal` and `ACQuire:NUMAVg 8` would be concatenated into a single command:

   ```
   TRIGger:MODe NORMal;:ACQuire:NUMAVg 8
   ```

2. If concatenated commands have headers that differ by only the last mnemonic, you can abbreviate the second command and eliminate the beginning colon. For example, the commands `ACQuire:MODe ENVelope` and `ACQuire:NUMAVg 4` could be concatenated into a single command:

   ```
   ACQuire:MODe ENVelope; NUMAVg 4
   ```

   The longer version works equally well:

   ```
   ACQuire:MODe ENVelope;:ACQuire:NUMAVg 4
   ```

3. Never precede a star (*) command with a colon:

   ```
   ACQuire:MODe ENVelope;*TRG
   ```

   The oscilloscope processes commands that follow as if the star command was not there. The following example is valid:

   ```
   ACQuire:MODe ENVelope;*TRG;NUMAVg 2
   ```

4. When you concatenate queries, the oscilloscope concatenates responses to all the queries into a single response message. For example, if the display intensity for text is “bright,” and for the waveform it is “dim,” the concatenated query

   ```
   DISplay:INTENsity:TEXt?;WAVEform?
   ```

   will return either `:DISPLAY:INTENSITY:TEXT BRI;:DISPLAY:INTENSITY:WAVEFORM DIM` if header is on, or `BRI;DIM` if header is off.

5. You may concatenate set commands and queries in the same message. For example:

   ```
   ACQuire:MODe NORMal;NUMAVg?;STATE?
   ```

   is a valid message that sets the acquisition mode to normal, then queries the oscilloscope for the number of acquisitions for averaging and the current acquisition state. The oscilloscope executes concatenated commands and queries in the order it receives them.

Here are some invalid concatenations:

- `DISPlay:INTENsity:TEXt BRI;ACQuire:NUMAVg 16`
  (no colon before `ACQuire`)
- `DISPlay:INTENsity:TEXt DIM;:WAVEform BRI`
  (extra colon before `WAVEform` — could also use `DISPlay:INTENsity:WAVEform` instead)
- `DISPlay:INTENsity:TEXt DIM;:*TRG`
  (extra colon before a star (*) command)

### Message Terminators

This manual uses <EOM> (End of message) to represent a message terminator.

**GPIB End of Message Terminators.** GPIB EOM terminators can be the END message (EOI asserted concurrently with the last data byte), the ASCII code for line feed (LF) sent as the last data byte, or both. The oscilloscope always terminates messages with LF and EOI. White space is allowed before the terminator; for example, CR LF is acceptable.

**RS-232 End of Message Terminators.** RS-232 EOM terminators can be a CR (carriage return), LF (line feed), CRLF (carriage return followed by a line feed), or LFCR (line feed followed by a carriage return). When receiving, the oscilloscope accepts all four combinations as valid input message terminators regardless of the currently selected terminator. When a combination of multiple characters is selected (CRLF or LFCR), the oscilloscope interprets the first character as the terminator; the oscilloscope interprets the second character as a null command.

## Constructed Mnemonics

Some header mnemonics specify one of a range of mnemonics. For example, a channel mnemonic can be either CH1 or CH2. You use these mnemonics in the command just as you do any other mnemonic. For example, there is a CH1:VOLts command and there is also a CH2:VOLts command. In the command descriptions, this list of choices is abbreviated CH<x>.

### Cursor Position Mnemonic

When the oscilloscope displays cursors, commands may specify which cursor of the pair to use.

| Symbol | Meaning |
|---|---|
| POSITION<x> | A cursor selector; <x> is either 1 or 2 |

### Measurement Specifier Mnemonics

Commands can specify which measurement to set or query as a mnemonic in the header. The oscilloscope can display up to four automated measurements with each displayed waveform. The displayed measurements are specified in this way:

| Symbol | Meaning |
|---|---|
| MEAS<x> | A measurement specifier; <x> is either 1, 2, 3,or 4 |

### Channel Mnemonics

Commands specify the channel to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| CH<x> | A channel specifier; <x> is either 1 or 2 |

### Math Waveform Mnemonics

Commands can specify the mathematical waveform to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| MATH<x> | A math waveform specifier; <x> is 1 |

### Reference Waveform Mnemonics

Commands can specify the reference waveform to use as a mnemonic in the header.

| Symbol | Meaning |
|---|---|
| REF<x> | A reference waveform specifier; <x> is either 1 or 2 |

### Waveform Mnemonics

In some commands you can specify a waveform, regardless of whether it is a channel waveform, a math waveform, or a reference waveform. Specify these as follows:

| Symbol | Meaning |
|---|---|
| <wfm> | Can be CH<x>, MATH, MATH1, or REF<x> |

## Argument Types

A command argument can be in one of several forms. Each command description describes the argument types to use with that command.

### Numeric Arguments

Many oscilloscope commands require numeric arguments. The syntax shows the format that the oscilloscope returns in response to a query. This is also the preferred format when sending the command to the oscilloscope though it will accept any of the formats. This manual represents these arguments as follows:

| Symbol | Meaning |
|---|---|
| <NR1> | Signed integer value |
| <NR2> | Floating point value without an exponent |
| <NR3> | Floating point value with an exponent |

The oscilloscope will automatically force most numeric arguments to a valid setting, either by rounding or truncating, when you input an invalid number unless otherwise noted in the command description.

### Quoted String Arguments

Some commands accept or return data in the form of a quoted string, which is simply a group of ASCII characters enclosed by a single quote (') or double quote ("). For example: `"this is a quoted string"`

| Symbol | Meaning |
|---|---|
| <QString> | Quoted string of ASCII text |

Follow these rules when you use quoted strings:

- A quoted string can include any character defined in the 7-bit ASCII character set. (See *Appendix A*).
- Use the same type of quote character to open and close the string:

  `"this is a valid string"`

- You can mix quotation marks within a string as long as you follow the previous rule:

  `"this is an 'acceptable' string"`

- You can include a quote character within a string simply by repeating the quote. For example,

  ```
  "here is a "" mark"
  ```

- Strings can have upper or lower case characters.
- You cannot terminate a quoted string with the END message before the closing delimiter.
- A carriage return or line feed embedded in a quoted string does not terminate the string, but is treated as just another character in the string.
- The maximum length of a quoted string returned from a query is 1000 characters.

Here are some invalid strings:

```
"Invalid string argument'
(quotes are not of the same type)

"test<EOI>"
(termination character is embedded in the string)
```

## Block Arguments

Several oscilloscope commands use a block argument form:

| Symbol | Meaning |
|---|---|
| <NZDig> | A non-zero digit character in the range 1-9 |
| <Dig> | A digit character in the range 0-9 |
| <DChar> | A character with the hex equivalent of 00 through FF hexadecimal (0 through 255 decimal) |
| <Block> | A block of data bytes, defined as:<br><Block> ::=<br>{ #<NZDig><Dig>[<Dig>...][<DChar>...]<br>\| #0[<DChar>...]<terminator> } |

<NZDig> specifies the number of <Dig> elements that follow. Taken together, the <Dig> elements form a decimal integer that specifies how many <DChar> elements follow.

#0 means that the <Block> is an indefinite length block. The <terminator> ends the block. You should not use indefinite length blocks with RS-232, because there is no way to include a <terminator> character as a <DChar> character.

The first occurrence of a <terminator> character signals the end of the block and any subsequent <DChar> characters will be interpreted as a syntax error.

With the GPIB, the EOI line signals the last byte. Figure 2–2 shows an example of a block argument.

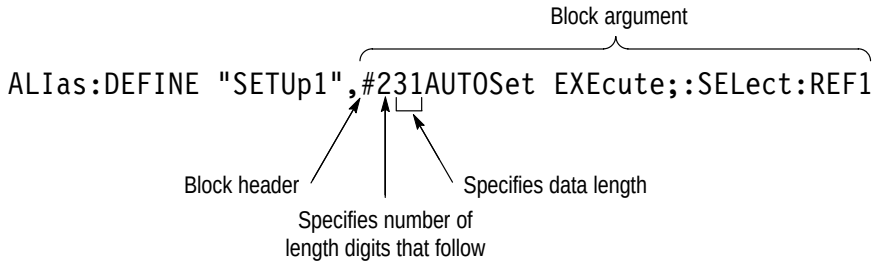



**Figure 2–2: Block argument example**

# Command Groups

This section lists commands by functional groups. The following section, *Command Descriptions*, starting on page 2–27, lists commands alphabetically.

The oscilloscope GPIB and RS-232 interfaces conform to Tektronix standard codes and formats except where noted. The GPIB interface also conforms to IEEE Std 488.2-1987 except where noted.

## Acquisition Commands

Acquisition commands affect the acquisition of waveforms. These commands, shown in Table 2–4, control mode, averaging, enveloping, and single-waveform acquisition.

**Table 2–4: Acquisition commands**

| Header | Description |
|---|---|
| ACQuire? | Return acquisition parameters |
| ACQuire:MODe | Set/query acquisition mode |
| ACQuire:NUMACq? | Return # of acquisitions obtained |
| ACQuire:NUMAVg | Set/query number of acquisitions for average |
| ACQuire:NUMEnv | Set/query number of acquisitions for envelope |
| ACQuire:STATE | Start or stop acquisition system |
| ACQuire:STOPAfter | Set/query acquisition control |

## Alias Commands

Alias commands let you define your own commands as a sequence of standard commands. This is useful when you use the same commands each time you perform a certain task, such as setting up the oscilloscope to take a measurement.

**Table 2–5: Alias commands**

| Header | Description |
|---|---|
| ALIas? | Set/query alias expansion state |
| ALIas:CATALOG? | Return a list of aliases |
| ALIas:DEFINE | Create a new alias |
| ALIas:DELEte | Remove an alias |
| ALIas:DELEte:ALL | Remove all aliases |
| ALIas:DELEte:NAMe | Remove a named alias |
| ALIas:STATE | Turn the alias state on and off |

## Calibration and Diagnostic Commands

Calibration and Diagnostic commands let you run the self-calibration and diagnostic routines that are built into the oscilloscope. The diagnostic test operation includes selecting the test sequence, executing the sequence, and then examining the results.

**Table 2–6: Calibrating and diagnostic commands**

| Header | Description |
|---|---|
| *CAL? | Perform an internal self calibration |
| CALibrate | Perform an internal signal path compensation |
| CALibrate:STATUS | Return status from last adjustment sequence |
| DIAg:RESULT:FLAg? | Return diagnostic tests status |
| DIAg:RESULT:LOG? | Return diagnostic test sequence results |
| DIAg:SELect:ALL | Diagnostic test sequence for<br>Acquisition, Processor, Display, Front panel, and floppy disk drive |
| DIAg:STATE | Control diagnostic tests |

## Cursor Commands

Cursor commands control cursor display and readout.

**Table 2–7: Cursor commands**

| Header | Description |
|---|---|
| CURSor? | Returns cursor settings |
| CURSor:FUNCtion | Set cursors on or off; select cursor type |
| CURSor:HBArs? | Return horizontal bar cursor settings |
| CURSor:HBArs:DELTa? | Return vertical distance between horizontal bar cursors |
| CURSor:HBArs:POSITION<x> | Position a horizontal bar cursor |
| CURSor:HBArs:SELect | Set which cursor the knob controls |
| CURSor:PAIred:HDELTA? | Return horizontal distance between first and second paired cursors |
| CURSor:PAIred:HPOS1? | Return horizontal position of first paired cursor |
| CURSor:PAIred:HPOS2? | Return horizontal position of second paired cursor |
| CURSor:PAIred:POSITION<x> | Set/query vertical bar position of paired cursor |
| CURSor:PAIred:SELect? | Select active paired cursor |
| CURSor:PAIred:VDELTA? | Return vertical distance between first and second paired cursors |
| CURSor:VBArs | Set/query vertical bar cursor settings |
| CURSor:VBArs:DELTa? | Return horizontal distance between vertical bar cursors |
| CURSor:VBArs:POSITION<x> | Position a vertical bar cursor |
| CURSor:VBArs:SELect | Set which cursor the knob controls |
| CURSor:VBArs:UNIts | Set vertical cursors to time or frequency |

## Display Commands

Display commands let you change the graticule style, change the displayed intensities, and clear the menu.

**Table 2–8: Display commands**

| Header | Description |
|---|---|
| CLEARMenu | Clear menus from display |
| DISplay? | Returns display settings |
| DISplay:CLOCk | Set/query display of system date and time |
| DISplay:FORMat | YT or XY display |
| DISplay:GRAticule | Graticule style |
| DISplay:INTENSITy? | Returns intensity settings |
| DISplay:INTENSITy:CONTRast | Waveform intensified zone brightness |
| DISplay:INTENSITy:OVERAll | Main brightness |
| DISplay:INTENSITy:TEXt | Text brightness |
| DISplay:INTENSITy:WAVEform | Waveform brightness |
| DISplay:PERSistence | Set/query the display accumulate time |
| DISplay:STYle | Waveform dots, vectors, dot accumulate or vector accumulate |
| DISplay:TRIGT | Controls the display of the trigger indicator on screen |

## File System Commands

File system commands let you store and recall data using the built-in 3.5 inch floppy disk drive. Table 2–9 lists these commands.

**Table 2–9: File system commands**

| Header | Description |
|---|---|
| FILESystem:COPy | Copy file to new file |
| FILESystem:CWD | Set directory path |
| FILESystem:DELEte | Delete named file |
| FILESystem:DELWarn | Set front-panel delete warning |
| FILESystem:DIR | Make directory |
| FILESystem:FORMat | Format named drive |
| FILESystem:FREESpace | Return free space on current drive |

**Table 2–9: File system commands (Cont.)**

| Header | Description |
|---|---|
| FILESystem:MKDir | Make new directory |
| FILESystem:OVERWrite | Set file-overwrite protection |
| FILESystem:PRInt | Print file to port |
| FILESystem:REName | Assign new name to file |
| FILESystem:RMDir | Delete named directory |

## Hard Copy Commands

The hard copy commands let you control the format of hard copy output and control the initiation and termination of hard copies.

**Table 2–10: Hard copy commands**

| Header | Description |
|---|---|
| HARDCopy | Start or terminate hard copy |
| HARDCopy:FILEName | Set/query filename to which to send hard copy data |
| HARDCopy:FORMat | Hard copy output format |
| HARDCopy:LAYout | Hard copy orientation |
| HARDCopy:PORT | Hard copy port for output |

## Horizontal Commands

Horizontal commands control the oscilloscope time bases. You can set the time per division (or time per point) of both the main and delayed time bases.

**Table 2–11: Horizontal commands**

| Header | Description |
|---|---|
| HORizontal? | Return horizontal settings |
| HORizontal:DELay? | Return delay time base settings |
| HORizontal:DELay:MODe | Set delay time base mode |
| HORizontal:DELay:SCAle | Set delay time base time/division |
| HORizontal:DELay:SECdiv | Same as HORizontal:DELay:SCAle |
| HORizontal:DELay:TIMe | Set delay time |

**Table 2–11: Horizontal commands (Cont.)**

| Header | Description |
|---|---|
| HORizontal:DELay:TIMe? | Return delay time parameters |
| HORizontal:DELay:TIMe:RUNSAfter | Set/query time delay between main trigger and delayed time base |
| HORizontal:FITtoscreen | Set/query horizontal 2X magnify |
| HORizontal:MAIn? | Return time/division of main time base |
| HORizontal:MAIn:SCAle | Set main time base time/division |
| HORizontal:MAIn:SECdiv | Same as HORizontal:MAIn:SCAle |
| HORizontal:MODe | Turn delay time base on or off |
| HORizontal:POSition | Set portion of waveform to display |
| HORizontal:RECOrdlength? | Return number of points in waveform record (always 1000 points) |
| HORizontal:REF<x> | Set position lock for REF waveforms |
| HORizontal:SCAle | Same as HORizontal:MAIn:SCAle |
| HORizontal:SECdiv | Same as HORizontal:MAIn:SCAle |
| HORizontal:TRIGger? | Return main time base trigger position |
| HORizontal:TRIGger:POSition | Set main time base trigger position |

## Measurement Commands

Measurement commands control the automated measurement system. Up to four automated measurements can be displayed on the screen of the oscilloscope. In the commands, these four measurement readouts are named MEAS<x>, where <x> can be 1, 2, 3, or 4.

In addition to the four displayed measurement readouts, the measurement commands let you specify a fifth measurement, IMMed. The immediate measurement has no front-panel equivalent, and the oscilloscope never displays immediate measurements. Immediate measurements slow the waveform update rate less than displayed measurements because immediate measurements are computed only when they are requested.

Whether you are using displayed or immediate measurements, you use the VALue? query to obtain measurement results.

Several measurement commands set and query measurement parameters. You can assign some parameters, such as waveform sources, differently for each measurement readout. Other parameters, such as reference levels, have only one value which applies to all measurements.

**Table 2–12: Measurement commands**

| Header | Description |
|---|---|
| MEASUrement? | Return all measurement parameters |
| MEASUrement:GATing | Set/query measurement gating |
| MEASUrement:IMMed? | Return immediate measurement parameters |
| MEASUrement:IMMed:SOUrce[1] | Set the channel from which to take the immediate measurement |
| MEASUrement:IMMed:TYPe | The immediate measurement to be taken |
| MEASUrement:IMMed:UNIts? | Return immediate measurement units |
| MEASUrement:IMMed:VALue? | Return immediate measurement result |
| MEASUrement:MEAS<x>? | Return parameters on the measurement |
| MEASUrement:MEAS<x>: SOUrce[1] | Set the channel from which to take the measurement |
| MEASUrement:MEAS<x>:STATE | Turn measurement display on or off |
| MEASUrement:MEAS<x>:TYPe | Set/query the measurement to be taken |
| MEASUrement:MEAS<x>:UNIts? | return units to use for measurement |
| MEASUrement:MEAS<x>:VALue? | Measurement result query |
| MEASUrement:METHod | Set/query the method for calculating reference levels |
| MEASUrement:REFLevel? | Return percent and absolute reference levels |
| MEASUrement:REFLevel: ABSolute:HIGH | Set/query the top level for measurement (90% level) |
| MEASUrement:REFLevel: ABSolute:LOW | Set/query the low level for measurement (10% level) |
| MEASUrement:REFLevel: ABSolute:MID | Set/query the mid level for measurements |
| MEASUrement:REFLevel: METHod | Set/query the method to assign HIGH, MID, and LOW levels: either % or absolute volts |
| MEASUrement:REFLevel: PERCent:HIGH | Set/query the top level for measurement (90% level) |
| MEASUrement:REFLevel: PERCent:LOW | Set/query the low level for measurement (10% level) |
| MEASUrement:REFLevel: PERCent:MID | Set/query the mid level for measurements |

## Miscellaneous Commands

Miscellaneous commands are a group of commands that do not fit into any other category.

Several commands and queries used with the oscilloscope are common to all devices on the GPIB bus and the RS-232 interface. These commands and queries are defined by IEEE Std. 488.2-1987 and Tek Standard Codes and Formats 1989 and begin with an asterisk (*) character.

**Table 2–13: Miscellaneous commands**

| Header | Description |
|---|---|
| AUTOSet | Automatic oscilloscope setup |
| DATE | Set/query oscilloscope date |
| *DDT | Define group execute trigger (GET) |
| FACtory | Same as *RST |
| HDR | Same as HEADer |
| HEADer | Return command header with query |
| ID? | Return Tektronix Codes and Format instrument identification |
| *IDN? | Return IEEE-488 instrument identification |
| LOCk | Lock front panel (local lockout) |
| *LRN? | Learn device setting |
| NEWpass | Change password for User Protected Data |
| PASSWord | Access to change User Protected Data |
| *PUD | Set/query user-protected data |
| REM | No action; remark only |
| *RST | Return most settings to factory default |
| SET? | Same as *LRN? |
| TEKSecure | Initialize waveforms and setups |
| TIMe | Set/query oscilloscope time |
| *TRG | Perform Group Execute Trigger (GET) |
| *TST? | Return self-test results |
| UNLock | Unlock front panel (local lockout) |
| VERBos | Set/query full command name or minimum spellings with query |

## RS-232 Commands

RS-232 commands let you set or query the RS-232 port parameters.

**Table 2–14: RS-232 commands**

| Header | Description |
|---|---|
| RS232? | Return RS232 parameters |
| RS232:BAUd | Set/query baud rate |
| RS232:CONTrol:DCD | Return DCD monitoring setting |
| RS232:CONTrol:RTS | Set/query hard flagging |
| RS232:HARDFlagging | Set/query hard flagging |
| RS232:MODE | Always returns RAW |
| RS232:PACE | Set/query soft flagging |
| RS232:PARity | Set/query parity type |
| RS232:PRESet | Set default RS-232 parameters |
| RS232:SBITS | Set/query number of stop bits |
| RS232:SOFTFlagging | Set/query soft flagging |
| RS232:STOPBits | Set/query number of stop bits |
| RS232:TRANsmit:DELay | Set/query delay before query response |
| RS232:TRANsmit:TERMinator | Set/query end-of-line terminator |

## Save and Recall Commands

Save and Recall commands let you save and retrieve internal waveforms and settings. Saving settings saves most of the current oscilloscope settings. Recalling settings returns the oscilloscope settings to those of the saved settings.

**Table 2–15: Save and recall commands**

| Header | Description |
|---|---|
| *RCL | Recall saved oscilloscope setting |
| RECAll:SETUp | Recall saved oscilloscope setting |
| RECAll:WAVEform | Recall saved waveform from file |
| *SAV | Save oscilloscope setting |
| SAVe:SETUp | Save oscilloscope setting |
| SAVe:WAVEform | Save waveform |
| SAVe:WAVEform:FILEFormat | Save waveform to specified file format |

## Status and Error Commands

Table 2–16 lists the status and error commands the oscilloscope supports. These commands let you determine the status of the oscilloscope.

Several commands and queries used with the oscilloscope are common to all devices on the GPIB bus. These commands and queries are defined by IEEE Std. 488.2-1987 and Tek Standard Codes and Formats 1989, and begin with an asterisk (*) character.

**Table 2–16: Status and error commands**

| Header | Description |
|---|---|
| ALLEv? | Return all events |
| BUSY? | Return scope busy events |
| *CLS | Clear device |
| DESE | Device event status enable |
| *ESE | Event status enable |
| *ESR? | Return standard event status register |
| EVENT? | Return event code |
| EVMsg? | Return event code and message |
| EVQty? | Return number of events in queue |
| *OPC | Operation complete |
| *PSC | Power-on status clear |
| *SRE | Service request enable |
| *STB? | Read status byte |
| *WAI | Wait to continue |

## Trigger Commands

Trigger commands control all aspects of oscilloscope triggering.

You can set the main trigger to one of two modes: edge and video. The default mode is Edge triggering.

Edge triggering lets you acquire a waveform when the signal passes through a voltage level of your choosing. Video triggering adds the capability of triggering on NTSC or PAL standard video fields and lines.

**Table 2–17: Trigger commands**

| Header | Description |
|---|---|
| TRIGger | Set/query trigger event |
| TRIGger:MAIn | Set main trigger level to 50% |
| TRIGger:MAIn:EDGE? | Return edge trigger parameters |
| TRIGger:MAIn:EDGE:COUPling | Set edge trigger coupling |
| TRIGger:MAIn:EDGE:SLOpe | Set edge trigger slope |
| TRIGger:MAIn:EDGE:SOUrce | Set edge trigger source |
| TRIGger:MAIn:HOLDoff? | Return trigger holdoff value |
| TRIGger:MAIn:HOLdoff:VALue | Set/query trigger holdoff value |
| TRIGger:MAIn:LEVel | Set/query trigger level |
| TRIGger:MAIn:MODe | Set/query trigger mode |
| TRIGger:MAIn:TYPe | Set main trigger edge or video |
| TRIGger:MAIn:VIDeo:FIELD | Set video trigger field |
| TRIGger:MAIn:VIDeo:HOLdoff? | Return video trigger holdoff value |
| TRIGger:MAIn:VIDeo:HOLdoff:VALue | Set video trigger holdoff value |
| TRIGger:MAIn:VIDeo:SCAN | Set video trigger scan rate |
| TRIGger:MAIn:VIDeo:SOUrce | Set video trigger source |
| TRIGger:STATE? | Return trigger system status |

## Vertical Commands

Vertical commands control the display of channels and of math and reference waveforms. The SELect:<wfm> command also selects the waveform to be used by many commands in other command groups.

**Table 2–18: Vertical commands**

| Header | Description |
|---|---|
| CH<x>? | Return vertical parameters |
| CH<x>:BANdwidth | Channel bandwidth |
| CH<x>:COUPling | Channel coupling |
| CH<x>:INVert | Invert channel |
| CH<x>:OFFSet | Channel offset |
| CH<x>:POSition | Channel position |

**Table 2–18: Vertical commands (Cont.)**

| Header | Description |
|---|---|
| CH<x>:PRObe? | Return channel probe attentuation |
| CH<x>:SCAle | Channel volts or dB per division |
| CH<x>:VOLts | Same as CH<x>:SCAle |
| EXT:PRObe? | Return external trigger attenuation factor |
| MATH1? | Return math waveform definition |
| MATH1:DEFINE | Math waveform definition |
| SELect? | Return selected waveform |
| SELect:<wfm> | Set selected waveform |
| SELect:CONTROl | Set/query the channel affected by the front-panel controls |
| ZOOM:VERTical:POSition | Set/query the vertical position of math and reference waveforms |
| ZOOM:VERTical:SCALEe | Set/query the vertical expansion or compression factor |

## Waveform Commands

Waveform commands let you transfer waveform data points to and from the oscilloscope. Waveform data points are a collection of values that define a waveform. One data value usually represents one data point in the waveform record. When working with enveloped waveforms, each data value is either the min or max of a max/min pair. Before you can transfer waveform data, you must specify the data format, record length, and waveform locations.

### Waveform Data Commands

Acquired waveform data uses either one or two 8-bit data bytes to represent each data point. The number of bytes used depends on the acquisition mode specified when you acquired the data. Data acquired in SAMple, ENVelope, or PEAKdetect modes use one 8-bit byte per waveform data point; data acquired in AVErage mode uses two 8-bit bytes per point. For more information on the acquisition modes see the ACQuire:MODe command on page 2–28.

The DATa:WIDth command lets you specify the number of bytes per data point when transferring data to and from the oscilloscope. If you specify two bytes for data that uses only one, the least significant byte will be filled with zeros; if you specify one byte for data that uses two, the least significant byte will be ignored.

The oscilloscope can transfer waveform data in either ASCII or binary format. You specify the format with the DATa:ENCdg command.

**ASCII Data.** ASCII data is represented by single-byte, signed-integer values in the range –128 to 127. Each data point value consists of up to three ASCII characters for the value and one for the minus sign if the value is negative. Commas separate data points. The DATa:WIDth command is ignored when using ASCII format since the byte width is always one.

An example ASCII waveform data string may look like this:

```
CURVE<space>-110,-109,-110,-110,-109,-107,-109,-107,
-106,-105,-103,-100,-97,-90,-84,-80
```

**Binary Data.** Binary data can be represented by signed integer or positive integer values. The range of the values depends on the byte width specified. When the byte width is one, signed integer data ranges from –128 to 127 and positive integer values range from 0 to 255. When the byte width is two, the values range from –32768 to 32767.

The defined binary formats also specify the order in which the bytes are transferred giving a total of four binary formats: RIBinary, RPBinary, SRIbinary, and SRPbinary.

RIBinary is a signed integer when the most significant byte is transferred first, and RPBinary is a positive integer when the most significant byte is transferred first. SRIbinary and SRPbinary correspond to RIBinary and RPBinary respectively but use a swapped byte order when the least significant byte is transferred first. The byte order is ignored when DATa:WIDth is set to 1.

### Waveform Data/record Lengths

You can transfer multiple points for each waveform record. You can transfer a portion of the waveform or you can transfer the entire record. The DATa:STARt and DATa:STOP commands let you specify the first and last data points of the waveform record.

When transferring data into the oscilloscope you must specify the location of the first data point within the waveform record. For example, when DATa:STARt is set to 1, data points will be stored starting with the first point in the record, and when DATa:STARt is set to 500, data will be stored starting at the 500$^{th}$ point in the record. DATa:STOP will be ignored when transferring data into the oscilloscope as the oscilloscope will stop reading data when there is no more data to read or when the record length has been reached.

When transferring data from the oscilloscope you must specify the first and last data points in the waveform record. Setting DATa:STARt to 1 and DATa:STOP to 1000 always returns the entire waveform.

### Waveform Data Locations and Memory Allocation

The DATa:SOUrce command specifies the location of the data when transferring waveforms from the oscilloscope. You can transfer multiple waveforms at one time by specifying more than one source.

You can transfer only one waveform to the oscilloscope at a time. Waveforms sent to the oscilloscope are always stored in one of the two reference memory locations. You specify the reference memory location with the DATa:DESTination command. The waveform should be 1000 data points in length.

***NOTE**. The oscilloscope accepts waveforms that are ≤1000 data points long. The oscilloscope will truncate waveforms larger than 1000 data points.*

### Waveform Preamble

Each waveform that is transferred has an associated waveform preamble that contains information such as the horizontal scale, vertical scale, and other settings in place when the waveform was created. Refer to the WFMPre commands starting on page 2–167 for more information about the waveform preamble.

### Scaling Waveform Data

Once you transfer the waveform data to the controller, you can convert the data points into voltage values for analysis using information from the waveform preamble. Use the GETWFM program on the diskette supplied with this manual to learn how to scale data.

### Transferring Waveform Data from the Oscilloscope

Transfer waveforms from the oscilloscope to an external controller using the following sequence.

1. Select the waveform source(s) using the DATa:SOUrce command. If you want to transfer multiple waveforms, select more than one source.
2. Specify the waveform data format using DATa:ENCdg.
3. Specify the number of bytes per data point using DATa:WIDth.
4. Specify the portion of the waveform that you want to transfer using DATa:STARt and DATa:STOP.
5. Transfer waveform preamble information using WFMPRe? query.
6. Transfer waveform data from the oscilloscope using the CURVe? query.

### Transferring Waveform Data to the Oscilloscope

Transfer waveform data to one of the two reference memory locations in the oscilloscope using the following sequence.

1. Specify the waveform reference memory using DATa:DESTination.
2. Specify the waveform data format using DATa:ENCdg.
3. Specify the number of bytes per data point using DATa:WIDth.
4. Specify the first data point in the waveform record using DATa:STARt.
5. Transfer waveform preamble information using WFMPRe:<wfm>.
6. Transfer waveform data to the oscilloscope using CURVe.

**Table 2–19: Waveform commands**

| Header | Description |
|---|---|
| CURVe | Transfer waveform data |
| DATa | Waveform data format and location |
| DATa:DESTination | Destination for waveforms sent to oscilloscope |
| DATa:ENCdg | Waveform data encoding method |
| DATa:SOUrce | Source of CURVe? data |
| DATa:STARt | Starting point in waveform transfer |
| DATa:STOP | Ending point in waveform transfer |
| DATa:TARget | Same as DATa:DESTination |
| DATa:WIDth | Byte width of waveform points |
| WAVFrm? | Returns waveform preamble and curve data |
| WFMPre? | Returns waveform preamble |
| WFMPre:BIT_Nr | Preamble bit width of waveform points |
| WFMPre:BN_Fmt | Preamble binary encoding type |
| WFMPre:BYT_Nr | Preamble byte width of waveform points |
| WFMPre:BYT_Or | Preamble byte order of waveform points |
| WFMPre:ENCdg | Preamble encoding method |
| WFMPre:PT_Fmt | Format of curve points |
| WFMPre:PT_Off | Trigger Position |
| WFMPre:XINcr | Horizontal sampling interval |
| WFMPre:YMUlt | Vertical scale factor |
| WFMPre:YOFf | Vertical offset |
| WFMPre:YZEro | Offset voltage |

**Table 2–19: Waveform commands (Cont.)**

| Header | Description |
|---|---|
| WFMPre:<wfm>? | Return waveform format data |
| WFMPre:<wfm>:NR_Pt | Number of points in the curve |
| WFMPre:<wfm>:PT_Fmt | Format of curve points |
| WFMPre:<wfm>:PT_Off | Trigger position |
| WFMPre:<wfm>:WFId | Curve identifier |
| WFMPre:<wfm>:XINcr | Horizontal sampling interval |
| WFMPre:<wfm>:XUNit | Horizontal units |
| WFMPre:<wfm>:YMUlt | Vertical scale factor |
| WFMPre:<wfm>:YOFf | Vertical offset |
| WFMPre:<wfm>:YUNit | Vertical units |
| WFMPre:<wfm>:YZEro | Offset voltage |

# Command Descriptions

Commands either set oscilloscope features or query oscilloscope values. You can use some commands to do both, some to only set, and some to only query. This manual marks set only commands with the words “No Query Form” included with the command name. It marks query only commands with a question mark appended to the header, and includes the words “Query Only” in the command name.

This manual fully spells out headers, mnemonics, and arguments with the minimal spelling shown in upper case. For example, to use the abbreviated form of the ACQuire:MODe command just type ACQ:MOD.

## ACQuire? (Query Only)

Returns all the current acquisition parameters.

**Group** Acquisition

**Syntax** `ACQuire?`

**Examples** `ACQUIRE?`
might return the string `:ACQUIRE:STOPAFTER RUNSTOP;STATE 1;MODE SAMPLE;NUMENV 10;NUMAVG 16` for the current acquisition parameters.

## ACQuire:MODe

Sets or queries the acquisition mode of the oscilloscope. This command affects all live waveforms. This command is equivalent to setting Mode in the Acquire menu.

Waveforms are the displayed data point values taken from acquisition intervals. Each acquisition interval represents a time duration that is determined by the horizontal scale (time per division). The oscilloscope sampling system always samples at the maximum rate, and so an acquisition interval may include more than one sample.

The acquisition mode, which you set using this ACQuire:MODe command, determines how the final value of the acquisition interval is generated from the many data samples.

**Group** Acquisition

**Related Commands** ACQuire:NUMAVg, ACQuire:NUMENv, CURVe?, DATa:WIDth

**Syntax**

```
ACQuire:MODe { SAMple | PEAKdetect | AVErage | ENVelope }
ACQuire:MODe?
```

**Arguments**

`SAMple` specifies that the displayed data point value is simply the first sampled value that was taken during the acquisition interval. In sample mode, all waveform data has 8 bits of precision. You can request 16 bit data with a CURVe? query, but the lower-order 8 bits of data will be zero. `SAMple` is the default mode.

`PEAKdetect` specifies the display of the high-low range of the samples taken from a single waveform acquisition. The oscilloscope displays the high-low range as a vertical column that extends from the highest to the lowest value sampled during the acquisition interval. `PEAKdetect` mode can reveal the presence of aliasing.

`AVErage` specifies averaging mode, where the resulting waveform shows an average of `SAMple` data points from several separate waveform acquisitions. The number of waveform acquisitions that go into making up the average waveform is set or queried using the ACQuire:NUMAVg command.

`ENVelope` specifies envelope mode, where the resulting waveform shows the `PEAKdetect` range of data points from several separate waveform acquisitions. The number of waveform acquisitions that go into making up the envelope waveform is set or queried using the ACQuire:NUMENv command.

**Examples**

```
ACQUIRE:MODE ENVELOPE
```

sets the acquisition mode to display a waveform that is an envelope of many individual waveform acquisitions.

```
ACQUIRE:MODE?
```

might return `ENVELOPE`.

## ACQuire:NUMACq? (Query Only)

Indicates the number of acquisitions that have taken place since starting acquisition. This value is reset to zero when any Acquisition, Horizontal, or Vertical arguments that affect the waveform are modified. The maximum number of acquisitions that can be counted is $2^{30}$–1. This is the same value that the oscilloscope displays in the top line of the screen.

**Group** Acquisition

**Related Commands** ACQuire:STATE

**Syntax**

```
ACQuire:NUMACq?
```

**Returns**

```
<NR1>
```

**Examples**

```
ACQUIRE:NUMACQ?
```

might return 350, indicating that 350 acquisitions took place since an ACQUIRE:STATE RUN command was executed.

## ACQuire:NUMAVg

Sets the number of waveform acquisitions that make up an averaged waveform. This is equivalent to setting the **Average** count in the Acquisition Mode side menu.

**Group** Acquisition

**Related Commands** ACQuire:MODe

**Syntax** `ACQuire:NUMAVg <NR1>`

`ACQuire:NUMAVg?`

**Arguments** `<NR1>` is the number of waveform acquisitions, from 2 to 256.

**Examples** `ACQUIRE:NUMAVG 10`
specifies that an averaged waveform will show the result of combining 10 separately acquired waveforms.

`ACQUIRE:NUMAVG?`
might return 75, indicating that there are 75 acquisitions specified for averaging.

## ACQuire:NUMENv

Sets the number of waveform acquisitions that make up an envelope waveform. This is equivalent to setting the **Envelope** count in the Acquisition Mode side menu.

**Group** Acquisition

**Related Commands** ACQuire:MODe

**Syntax**

```
ACQuire:NUMENv { <NR1> | INFInite }
```

```
ACQuire:NUMENv?
```

**Arguments**

<NR1> ≠ 0 is the number of waveform acquisitions, from 2 to 256. The envelope will restart after the specified number of envelopes have been acquired or when the ACQuire:STATE RUN command is sent.

INFInite or <NR1> = 0 specifies continuous enveloping.

***NOTE**. If you set the acquisition system to single sequence envelope mode and set the number of envelopes to infinity, the oscilloscope will envelope a maximum of 257 acquisitions.*

**Examples**

```
ACQUIRE:NUMENV 10
```
specifies that an enveloped waveform will show the result of combining 10 separately acquired waveforms.

```
ACQUIRE:NUMENV?
```
might return 0, indicating that acquisitions are acquired infinitely for enveloped waveforms.

## ACQuire:STATE

Starts or stops acquisitions. This is the equivalent of pressing the front-panel **RUN/STOP** button. If ACQuire:STOPAfter is set to SEQuence, other signal events may also stop acquisition.

**Group** Acquisition

**Related Commands** ACQuire:NUMACq?, ACQuire:STOPAfter

**Syntax**

```
ACQuire:STATE { OFF | ON | RUN | STOP | <NR1> }
```

```
ACQuire:STATE?
```

**Arguments**

OFF or STOP or <NR1> = 0 stops acquisitions.

ON or RUN or <NR1> $\neq$ 0 starts acquisition and display of waveforms. If the command was issued in the middle of an acquisition sequence (for instance averaging or enveloping), RUN restarts the sequence, discarding any data accumulated before the STOP. It also resets the number of acquisitions.

**Examples**

```
ACQUIRE:STATE:RUN
```

starts acquisition of waveform data and resets the number of acquisitions count (NUMACQ) to zero.

```
ACQUIRE:STATE?
```

returns either 0 or 1, depending on whether the acquisition system is running.

## ACQuire:STOPAfter

Tells the oscilloscope when to stop taking acquisitions. This is equivalent to setting **Stop After** in the Acquire menu.

**Group** Acquisition

**Related Commands** ACQuire:MODe, ACQuire:STATE

**Syntax**

```
ACQuire:STOPAfter { RUNSTop | SEQuence}
ACQuire:STOPAfter?
```

**Arguments**

RUNSTop specifies that the run and stop state should be determined by the user pressing the front-panel **RUN/STOP** button.

SEQuence specifies "single sequence" operation, where the oscilloscope stops after it has acquired enough waveforms to satisfy the conditions of the acquisition mode. For example, if the acquisition mode is set to sample, and the horizontal scale is set to a speed that allows real-time operation, then the oscilloscope stops after digitizing a waveform from a single trigger event. However, if the acquisition mode is set to average 16 waveforms, then the oscilloscope stops only after acquiring all 16 waveforms. The ACQuire: STATE command and the front-panel RUN/STOP button also stop acquisitions when the oscilloscope is in single sequence mode.

***NOTE**. If you set the acquisition system to single sequence, envelope mode, and set the number of envelopes to infinity, the oscilloscope will envelope a maximum of 257 acquisitions.*

**Examples**

```
ACQUIRE:STOPAFTER RUNSTop
```
sets the oscilloscope to stop acquisition when the user presses the front-panel **RUN/STOP** button.

```
ACQUIRE:STOPAFTER?
```
might return SEQUENCE.

## ALIas

Turns command aliases on or off. This command is identical to the ALIas:STATE command.

**Group** Alias

**Syntax**

```
ALIas { OFF | ON | <NR1> }
ALIas?
```

**Arguments**

OFF or <NR1> = 0 turns alias expansion off. If a defined alias label is sent when ALIas is OFF, an execution error will be generated.

ON or <NR1> $\neq$ 0 turns alias expansion on. When the oscilloscope receives a defined alias, it substitutes the specified command sequence for the alias and executes it.

**Examples**

```
ALIAS ON
```
turns the alias feature on.

```
ALIAS?
```
returns 1 when aliases are on.

## ALIas:CATALOG? (Query Only)

Returns a list of the currently defined alias labels separated by commas. If no aliases are defined, the query returns the string "".

**Group** Alias

**Syntax**

```
ALIas:CATALOG?
```

**Returns**

```
<QString>[,<QString>...]
```

**Examples**

```
ALIAS:CATALOG?
```
might return the string "SETUP1","TESTMENU1","DEFAULT", showing there are 3 aliases named SETUP1, TESTMENU1, and DEFAULT.

## ALIas:DEFINE

Assigns a sequence of program messages to an alias label. If ALIas:STATE has been turned ON, these messages are substituted for the alias whenever it is received as a command or query. The ALIas:DEFINE? query returns the definition of a selected alias.

You can define up to 10 aliases at one time. Aliases can be recursive. That is, aliases can include other aliases with up to 10 levels of recursion.

**Group** Alias

**Syntax**

```
ALIas:DEFINE <QString><Comma>{ <QString> | <Block> }
```

```
ALIas:DEFINE? <QString>
```

**Arguments** The first <QString> is the alias label. This label cannot be a command name. Labels must start with a letter and can contain only letters, numbers, and underscores; other characters are not allowed. The label must be ≤12 characters.

The second <QString> or <Block> is a complete sequence of program messages. The messages can contain only valid commands separated by semicolons and following all rules for concatenating commands (see page 2–4). The sequence must be ≤80 characters.

---

***NOTE**. Attempting to give two aliases the same name causes an execution error. To give a new alias the name of an existing alias, you must first delete the existing alias.*

---

**Examples**

```
ALIAS:DEFINE "ST1",":RECALL:SETUP 5;:AUTOSET EXECUTE;:SELECT:CH1
ON"
```

defines an alias named “ST1” that sets up the oscilloscope.

```
ALIAS:DEFINE? "ST1"
```

might return `:ALIAS:DEFINE "ST1",#239:RECALL:SETUP 5;:AUTOSET EXECUTE;:SELECT:CH1 ON`

## ALIas:DELETe (No Query Form)

Removes a specified alias. This command is identical to ALIas:DELETe:NAMe.

**Group** Alias

**Syntax** `ALIas:DELETe <QString>`

**Arguments** `<QString>` is the name of the alias you want to remove. Using ALIas:DELETe without specifying an alias causes an execution error. `<QString>` must be a previously defined alias.

**Examples** `ALIAS:DELETE "SETUP1"`
deletes the alias named SETUP1.

## ALIas:DELETe:ALL (No Query Form)

Deletes all existing aliases.

**Group** Alias

**Syntax** `ALIas:DELETe:ALL`

**Examples** `ALIAS:DELETE:ALL`
deletes all aliases.

## ALIas:DELETe:NAMe (No Query Form)

Removes a specified alias. This command is identical to ALIas:DELEte.

**Group** Alias

**Syntax** `ALIas:DELEte:NAMe <QString>`

**Arguments** `<QString>` is the name of the alias to remove. Using ALIas:DELEte:NAMe without specifying an alias causes an execution error. `<QString>` must be a previously defined alias.

**Examples** `ALIAS:DELETE:NAME "STARTUP"`
deletes the alias named STARTUP.

## ALIas:STATE

Turns aliases on or off. This command is identical to the ALIas command.

**Group** Alias

**Syntax** `ALIas:STATE { OFF | ON | <NR1> }`

`ALIas:STATE?`

**Arguments** `OFF` or `<NR1>` = 0 turns alias expansion off. If a defined alias is sent when ALIas:STATE is OFF, a command error (102) will be generated.

`ON` or `<NR1>` $\neq$ 0 turns alias expansion on. When the oscilloscope receives a defined alias, it substitutes the specified command sequence for the alias and executes it.

**Examples** `ALIAS:STATE OFF`
turns the command alias feature off.

`ALIAS:STATE?`
returns 0 when alias mode is off.

## ALLEv? (Query Only)

Causes the oscilloscope to return all events and their messages, and removes the returned events from the Event Queue. The messages are separated by commas. Use the *ESR? query to enable the events to be returned. For a complete discussion of the use of these registers, see page 3–1. This command is similar to repeatedly sending *EVMsg? queries to the oscilloscope.

**Group** Status and Error

**Related Commands** *CLS, DESE, *ESE, *ESR?, EVENT?, EVMsg?, EVQty?, *SRE, *STB?

**Syntax** `ALLEv?`

**Returns** The event code and message in the following format:

```
<Event Code><Comma><QString>[<Comma><Event
Code><Comma><QString>...]

<QString>::= <Message>;[<Command>]
```

<Command> is the command that caused the error and may be returned when a command error is detected by the oscilloscope. As much of the command is returned as possible without exceeding the 60 character limit of the <Message> and <Command> strings combined. The command string is right-justified.

**Examples**

```
ALLEV?
```

might return the string `:ALLEV 2225,"Measurement error, No waveform to measure; ",420,"Query UNTERMINATED; "`.

## AUTOSet (No Query Form)

Causes the oscilloscope to adjust its vertical, horizontal, and trigger controls to provide a stable display of the selected waveform. This is equivalent to pressing the front-panel **AUTOSET** button. For a detailed description of the autoset function, consult the *TDS 340A, TDS 360, and TDS 380 Instruction Manual*.

***NOTE****. The AUTOSet command does not return control to the instrument controller until the autoset operation is complete.*

**Group** Miscellaneous

**Syntax** `AUTOSet { EXECute }`

**Arguments** `EXECute` autosets the selected waveform.

## BUSY? (Query Only)

Returns the status of the oscilloscope. This command allows you to synchronize the operation of the oscilloscope with your application program. Synchronization methods are described on page 3–7.

**Group** Status and error

**Related Commands** *OPC, *WAI

**Syntax** BUSY?

**Returns** <NR1> = 0 means that the oscilloscope is not busy processing a command whose execution time is extensive. These commands are listed in Table 2–20.

<NR1> = 1 means that the oscilloscope is busy processing one of the commands listed in Table 2–20.

**Table 2–20: Commands that affect BUSY? response**

| Operation | Command |
|---|---|
| Single sequence acquisition | ACQuire:STATE ON or<br>ACQuire:STATE RUN<br>(when ACQuire:STOPAfter is set to SEQuence) |
| Hard copy output | HARDCopy STARt |
| Signal path compensation | *CAL? or CALibrate |

**Examples** BUSY?
might return 1, indicating that the oscilloscope is busy.

## *CAL? (Query Only)

Runs an internal self calibration and returns oscilloscope calibration status. The self calibration can take three and a half minutes or more to respond. No other commands will be executed until calibration is complete.

**Group** Calibration and Diagnostic

**Related Commands** CALibrate INTERNal

**Syntax** *CAL?

**Returns** <NR1> = 0 indicates that the calibration completed without any errors detected.

<NR1> ≠ 0 indicates that the calibration did not complete successfully or completed with errors.

**Examples** *CAL?
performs an internal self calibration and might return 0 to indicate that the calibration was successful.

## CALibrate (No Query Form)

Runs an internal self-calibration.

***NOTE.*** *The self-calibration can take three and a half minutes or more to run. You cannot execute any other commands until calibration is complete.*

**Group** Calibration and Diagnostic

**Related Commands** *CAL?

**Syntax** `CALibrate INTERNal`

**Arguments** `INTERNal` specifies an internal self-calibration.

**Examples** `CALIBRATE INTERNAL`
performs an internal self-calibration.

## CALibrate:STATUS? (Query Only)

The CALibrate:STATUS? query returns the status from the last adjustment sequence.

**Group** Calibration and Diagnostic

**Related Commands** CALibrate INTERNal

**Syntax** `CALibrate:STATUS?`

**Returns** `PASS` indicates that the last adjustment sequence completed without any errors detected.

`FAIL` indicates that the last adjustment sequence did not complete successfully or completed with errors.

**Examples** `CALibrate:STATUS?`
might return `:CALIBRATE:STATUS FAIL` if oscilloscope calibration did not pass.

## CH<x>? (Query Only)

Returns the vertical parameters. Because CH<x>:SCAle and CH<x>:VOLts are identical, only CH<x>:SCAle is returned.

**Group** Vertical

**Syntax** `CH<x>?`

**Examples** `CH1?`
might return the string `:CH1:SCALE 1.0E-2;POSITION 0.0E0; OFFSET 0.0E0;COUPLING DC;BANDWIDTH FULL` for channel 1.

## CH<x>:BANdwidth

Sets or queries the bandwidth setting of the specified channel. This is equivalent to setting **Bandwidth** in the Vertical menu.

**Group** Vertical

**Syntax** `CH<x>:BANdwidth { TWEnty | FULl }`

`CH<x>:BANdwidth?`

**Arguments** `TWEnty` sets the channel bandwidth to 20 MHz.

`FULl` sets the channel bandwidth to the full bandwidth of the oscilloscope.

**Examples** `CH2:BANDWIDTH TWENTY`
sets the bandwidth of channel 2 to 20 MHz.

`CH1:BANDWIDTH?`
might return `FULL`, which indicates that there is no bandwidth limiting on channel 1.

## CH<x>:COUPling

Sets or queries the input attenuator coupling setting of the specified channel. This is equivalent to setting **Coupling** in the Vertical menu.

**Group** Vertical

**Syntax**

```
CH<x>:COUPling { AC | DC | GND }
```

```
CH<x>:COUPling?
```

**Arguments**

`AC` sets the specified channel to AC coupling.

`DC` sets the specified channel to DC coupling.

`GND` sets the specified channel to ground. Only a flat ground-level waveform is displayed.

**Examples**

```
CH1:COUPLING AC
```
establishes AC coupling on channel 1.

```
CH2:COUPLING?
```
might return `DC`, indicating that channel 2 is set to DC coupling.

## CH<x>:INVert

Sets or queries the input polarity setting of the specified channel. This command is equivalent to setting **Invert** in the Vertical menu.

**Group** Vertical

**Related Commands** None

**Syntax**

```
CH<x>:INVert { ON | OFF | <NR1> }
```

```
CH<x>:INVert?
```

**Arguments**

<NR1> = 0 sets the specified channel to invert off (normal display).

<NR1> = 1 sets the specified channel to invert on.

**Examples**

```
CH1:INVert OFF
```
establishes a non-inverted display on channel 1.

```
CH2:INVert?
```
might return 1, indicating that channel 2 is inverted.

## CH<x>:OFFSet

Sets or queries the offset, in volts, that is subtracted from the specified input channel before it is acquired. The greater the offset, the lower on the display the waveform appears. This is equivalent to setting **Offset** in the Vertical menu.

**Group** Vertical

**Related Commands** CH<x>:POSition

**Syntax**

```
CH<x>:OFFSet <NR3>
```

```
CH<x>:OFFSet?
```

**Arguments** <NR3> is the desired offset in volts. The range is dependent on the scale and the probe attenuation factor. The offset ranges are shown below.

**Table 2–21: Offset ranges (all channels)**

| CH<x>:SCAle | OFFSet Range |
|---|---|
| 2 mV/div — 99.5 mV/div | ±1 V |
| 100 mV/div — 995 mV/div | ±10 V |
| 1 V/div — 10 V/div | ±100 V |

**Examples**

```
CH1:OFFSET 0.5E0
```
lowers the channel 1 displayed waveform by 0.5 V.

```
CH1:OFFSET?
```
might return `5.0E-1`, indicating that the current channel 1 offset is 0.5 V.

## CH<x>:POSition

Sets or queries the vertical position of the specified channel. The position voltage value is applied to the signal before digitization. This is equivalent to setting **Position** in the Vertical menu or adjusting the front-panel **VERTICAL POSITION** knob.

**Group** Vertical

**Related Commands** CH<x>:OFFSet

**Syntax** `CH<x>:POSition <NR3>`

`CH<x>:POSition?`

**Arguments** `<NR3>` is the desired position, in divisions from the center graticule. The range is ±5 divisions.

**Examples** `CH2:POSITION 1.3E0`
positions the channel 2 input signal 1.3 divisions above the center of the display.

`CH1:POSITION?`
might return `-1.3E0`, indicating that the current position of channel 1 is at –1.3 divisions.

## CH<x>:PRObe? (Query Only)

Returns the attenuation factor of the probe that is attached to the specified channel.

**Group** Vertical

**Syntax** `CH<x>:PRObe?`

**Returns** `<NR3>`

**Examples** `CH2:PROBE?`
might return `1.0E1` for a 10x probe.

## CH<x>:SCAle

Sets or queries the vertical gain of the specified channel. This is equivalent to setting **Fine Scale** in the Vertical menu or adjusting the front-panel **Vertical SCALE** knob.

**Group** Vertical

**Related Commands** CH1:VOLts

**Syntax**

```
CH<x>:SCAle <NR3>
```

```
CH<x>:SCAle?
```

**Arguments** <NR3> is the gain, in volts per division. The range is 10 V/div to 2 mV/div when using a 1x probe.

**Examples**

```
CH1:SCALE 1.0E-1
```

sets the channel 1 gain to 100 mV/div.

```
CH2:SCALE?
```

might return 1.0E0, indicating that the current V/div setting of channel 2 is 1 V/div.

## CH<x>:VOLts

Sets or queries the vertical gain of the specified channel. This command is identical to the CH<x>:SCAle command and is included for compatibility purposes. Only CH<x>:SCAle is returned in response to a CH<x>? query.

**Group** Vertical

**Related Commands** CH1:SCAle

**Syntax** `CH<x>:VOLts <NR3>`

`CH<x>:VOLts?`

**Examples** `CH1:VOLTS 1.0E-1`
sets the channel 1 gain to 100 mV/div.

`CH2:VOLTS?`
might return `1.0E0`, indicating that the current V/div setting of channel 2 is 1 V/div.

## CLEARMenu (No Query Form)

Clears the current menu from the display. For some floppy-disk-drive menus, this command returns you to the previous menu. This command is equivalent to pressing the **CLEAR MENU** button on the front panel.

**Group** Display

**Syntax** `CLEARMenu`

**Examples** `CLEARMENU`
clears the menu from the display.

## *CLS (No Query Form)

Clears the oscilloscope status data structures.

**Group** Status and Error

**Related Commands** DESE, *ESE, *ESR?, EVENT?, EVMsg?, *SRE, *STB?

**Syntax** `*CLS`

The *CLS command clears the following:

- the Event Queue
- the Standard Event Status Register (SESR)
- the Status Byte Register (except the MAV bit; see below)

If the *CLS command immediately follows an `<EOI>`, the Output Queue and MAV bit (Status Byte Register bit 4) are also cleared. MAV indicates information is in the output queue. The device clear (DCL) GPIB control message will clear the output queue and thus MAV. *CLS does not clear the output queue or MAV. (A complete discussion of these registers and bits, and of event handling in general, begins on page 3–1.)

*CLS can suppress a Service Request that is to be generated by an *OPC. This happens if a hardcopy output or single sequence acquisition operation is still being processed when the *CLS command is executed.

## CURSor? (Query Only)

Returns all current cursor settings.

**Group** Cursor

**Syntax** `CURSor?`

**Examples** `CURSOR?`
might return `:CURSOR:FUNCTION OFF;VBARS:UNITS SECONDS; POSITION1 5.0E-4;POSITION2 4.5E-3;SELECT CURSOR1; :CURSOR:HBARS:POSITION1 3.2E0;POSITION2 -3.2E0; SELECT CURSOR1` as the current cursor settings.

## CURSor:FUNCtion

Selects and displays the cursor type. Cursors are attached to the selected channel. This command is equivalent to setting **Function** in the Cursor menu.

**Group** Cursor

**Related Commands** SELect:CONTROl

**Syntax** `CURSor:FUNCtion { HBArs | OFF | PAIred | VBArs }`

`CURSor:FUNCtion?`

**Arguments** `HBArs` specifies horizontal bar cursors that measure the vertical units (volts, dB).

`OFF` removes the cursors from the display.

`VBArs` specifies vertical bar cursors that measure time or frequency.

`PAIred` specifies paired cursors that show both horizontal and vertical units.

**Examples** `CURSOR:FUNCtion VBARS`
selects vertical bar type cursors.

## CURSor:HBArs? (Query Only)

Returns the current settings for the horizontal bar cursors.

**Group** Cursor

**Syntax** `CURSor:HBArs?`

**Examples** `CURSOR:HBARS?`
might return `:CURSOR:HBARS:POSITION1 0;POSITION2 0;SELECT CURSOR1`.

## CURSor:HBArs:DELTa? (Query Only)

Returns the difference (in vertical units) between the two horizontal bar cursors.

**Group** Cursor

**Syntax** `CURSor:HBArs:DELTa?`

**Returns** `<NR3>`

**Examples** `CURSOR:HBARS:DELTA?`
might return `5.08E0` for the difference between the two cursors.

## CURSor:HBArs:POSITION<x>

Positions a horizontal bar cursor.

**Group** Cursor

**Syntax** `CURSor:HBArs:POSITION<x> <NR3>`

`CURSor:HBArs:POSITION<x>?`

**Arguments** `<NR3>` specifies the cursor position relative to ground, in vertical units.

**Examples** `CURSOR:HBARS:POSITION1 2.5E-2`
positions one of the horizontal cursors at 25.0 mV, assuming the vertical units are volts.

`CURSOR:HBARS:POSITION2?`
might return `-6.4E-2`, indicating that one of the horizontal bar cursors is at –64.0 mV.

## CURSor:HBArs:SELect

Selects which horizontal bar cursor is active for front-panel control. The active cursor is displayed as a solid horizontal line and can be moved using the front-panel **General Purpose Knob**. The unselected cursor is displayed as a dashed horizontal line. This command is equivalent to pressing the **SELECT** button on the front panel.

**Group** Cursor

**Syntax** `CURSor:HBArs:SELect { CURSOR1 | CURSOR2 }`

`CURSor:HBArs:SELect?`

**Arguments** `CURSOR1` selects the first horizontal bar cursor.

`CURSOR2` selects the second horizontal bar cursor.

**Examples** `CURSOR:HBARS:SELECT CURSOR1`
selects the first horizontal bar cursor as the active cursor.

`CURSOR:HBARS:SELECT?`
returns `CURSOR1` when the first cursor is the active cursor.

## CURSor:PAIred:HDELta? (Query Only)

Queries the vertical distance between the first and second paired cursor. This is the absolute value of the first cursor's vertical position minus the second cursor's vertical position.

**Group** Cursor

**Related Commands** CURSor:FUNCtion

**Syntax** `CURSor:PAIred:HDELta?`

**Examples** `CURSOR:PAIRED:HDELTA?`
might return `5.08E0` for the voltage difference between the two cursors.

## CURSor:PAIred:HPOS1? (Query Only)

Queries the vertical position of the first paired cursor.

**Group** Cursor

**Related Commands** CURSor:FUNCtion

**Syntax** `CURSor:PAIred:HPOS1?`

**Examples** `CURSOR:PAIRED:HPOS1?`
might return `-6.4E-2`, indicating that the first cursor is at –64.0 mV.

## CURSor:PAIred:HPOS2? (Query Only)

Queries the vertical position of the second paired cursor.

**Group** Cursor

**Related Commands** CURSor:FUNCtion

**Syntax** `CURSor:PAIred:HPOS2?`

**Examples** `CURSOR:PAIRED:HPOS2?`
might return `-6.4E-2`, indicating the second cursor is at –64.0 mV.

## CURSor:PAIred:POSITION1

Sets or queries the horizontal (time or frequency) position of the first paired cursor.

**Group** Cursor

**Related Commands** CURSor:FUNCtion

**Syntax** `CURSor:PAIred:POSITION1 <NR3>`

`CURSor:PAIred:POSITION1?`

**Arguments** `<NR3>` specifies the position of the first paired cursor.

**Examples** `CURSOR:PAIRED:POSITION1 9.0E-6`
specifies the first paired cursor is at 9 μs.

`CURSOR:POSITION1?`
might return `1.0E-6`, indicating that the first paired cursor is at 1 μs.

## CURSor:PAIred:POSITION2

Sets or queries the horizontal (time or frequency) position of the second paired cursor.

**Group** Cursor

**Related Commands** CURSor:FUNCtion

**Syntax** `CURSor:PAIred:POSITION2 <NR3>`

`CURSor:PAIred:POSITION2?`

**Arguments** `<NR3>` specifies the position of the second paired cursor.

**Examples** `CURSOR:POSITION2?`
might return `1.0E-6`, indicating that the second paired cursor is at 1 μs.

## CURSor:PAIred:SELect

Selects the active paired cursor. The active cursor appears as a solid vertical line. The unselected cursor appears as a dashed vertical line. This command is equivalent to pressing the **SELECT** button on the front panel.

**Group** Cursor

**Syntax** `CURSor:PAIred:SELect { CURSOR1 | CURSOR2 }`

`CURSor:PAIred:SELect?`

**Arguments** `CURSOR1` specifies the first paired cursor.

`CURSOR2` specifies the second paired cursor.

**Examples** `CURSOR:PAIRED:SELECT CURSOR2`
selects the second paired cursor as the active cursor.

`CURSOR:PAIRED:SELECT?`
returns `CURSOR1` when the first paired cursor is the active cursor.

## CURSor:PAIred:VDELta? (Query Only)

Queries the horizontal distance (time or frequency) between paired cursors.

**Group** Cursor

**Related Commands** CURSor:FUNCtion

**Syntax** `CURSor:PAIred:VDELta?`

**Examples** `CURSOR:PAIRED:VDELTA?`
might return `1.064E0`, indicating that the time between the paired cursors is 1.064 seconds.

## CURSor:VBArs

Sets or queries the vertical bar cursor settings for horizontal position, delta, cursor selection, and units.

**Group** Cursor

**Related Commands** DATa:STARt, DATa:STOP

**Syntax** `CURSor:VBArs SNAp`

`CURSor:VBArs?`

**Arguments** `SNAp` positions the vertical bar cursors at DATa:STARt and DATa:STOP.

**Examples** `CURSOR:VBARS SNAP`
specifies that the cursors' positions are the same as the current DATA:START and DATA:STOP values.

`CURSOR:VBARS?`
might return `:CURSOR:VBARS:UNITS SECONDS;POSITION1 1.0E-6;POSITION2 9.0E-6;SELECT CURSOR2`.

## CURSor:VBArs:DELTa? (Query Only)

Returns the time or frequency between the two vertical bar cursors. The units (seconds or Hertz) are specified by the CURSor:VBArs:UNIts command.

**Group** Cursor

**Related Commands** CURSor:VBArs:UNIts

**Syntax** `CURSor:VBArs:DELTa?`

**Returns** `<NR3>`

**Examples** `CURSOR:VBARS:DELTa?`
might return `1.064E0`, indicating that the time between the vertical bar cursors is 1.064 seconds.

## CURSor:VBArs:POSITION<x>

Positions a vertical bar cursor for both vertical bar and paired cursors. The units are specified by the CURSor:VBArs:UNIts command.

**Group** Cursor

**Related Commands** CURSor:VBArs:UNIts

**Syntax**

```
CURSor:VBArs:POSITION<x> <NR3>
```

```
CURSor:VBArs:POSITION<x>?
```

**Arguments** <NR3> specifies the cursor position in the units specified by the CURSor:VBArs:UNIts command. The position is relative to the trigger position.

**Examples**

```
CURSOR:VBARS:POSITION2 9.0E-6
```

positions one of the vertical bar cursors at 9 μs.

```
CURSOR:VBARS:POSITION1?
```

might return 1.0E-6, indicating a vertical bar cursors is at 1 μs.

## CURSor:VBArs:SELect

Selects which vertical bar cursor is active. The active cursor is displayed as a solid vertical line and is moved using the front-panel **General Purpose Knob**. The unselected cursor is displayed as a dashed vertical line. This command is equivalent to pressing the **SELECT** button on the front panel.

**Group** Cursor

**Syntax**

```
CURSor:VBArs:SELect { CURSOR1 | CURSOR2 }
CURSor:VBArs:SELect?
```

**Arguments**

CURSOR1 specifies the first vertical bar cursor.

CURSOR2 specifies the second vertical bar cursor.

**Examples**

```
CURSOR:VBARS:SELECT CURSOR2
```
selects the second vertical bar cursor as the active cursor.

```
CURSOR:VBARS:SELECT?
```
returns CURSOR1 when the first vertical bar cursor is the active cursor.

## CURSor:VBArs:UNIts

Sets or queries the units for the vertical bar cursors. This is equivalent to setting **Time Units** in the Cursor menu. Note that if the current waveform is an FFT, the oscilloscope will display units as Hertz even if you specified seconds.

**Group** Cursor

**Related Commands** CURSor:VBArs:DELTa?, CURSor:VBArs:POSITION<x>

**Syntax**

```
CURSor:VBArs:UNIts { SECOnds | HERtz }
CURSor:VBArs:UNIts?
```

**Examples**

```
CURSOR:VBARS:UNITS SECONDS
```
sets the units for the vertical bar cursors to seconds.

```
CURSOR:VBARS:UNITS?
```
returns HERTZ when the vertical bar cursor units are Hertz.

## CURVe

Transfers waveform data to and from the oscilloscope in binary or ASCII format. Each waveform that is transferred has an associated waveform preamble that contains information such as data format and scale. Refer to the WFMPre command starting on page 2–167 for information about the waveform preamble. The data format is specified by the DATa:ENCdg and DATa:WIDTH commands.

The CURVe? query transfers data from the oscilloscope. The data source is specified by the DATa:SOUrce command. If more than one source is specified, a comma-separated list of data blocks is returned. The first and last data points that are transferred are specified by the DATa:STARt and DATa:STOP commands.

The CURVe command transfers waveform data to the oscilloscope. The data is stored in the reference memory location specified by DATa:DESTination starting with the data point specified by DATa:STARt. Only one waveform can be transferred at a time. The waveform is only displayed if the reference is displayed.

**Group** Waveform

**Related Commands** DATa, WFMPre

**Syntax**

```
CURVe { <Block> | <asc curve> }
CURVe?
```

**Arguments**

<Block> is the waveform data in binary format. The waveform is formatted as: #<x><yyy><data><newline> where <x> is the number of y bytes. For example, if <yyy> = 500, then <x> = 3. <yyy> is the number of bytes to transfer. If width is 1, then all bytes on the bus are single data points. If width is 2, then all bytes on the bus are 2-byte pairs. Use the DATa:WIDth command to set the width. <data> is the curve data. <newline> is a single byte newline character at the end of the data. See the GETWFM.C or GETWFM.BAS examples in the accompanying disk for more information.

<asc curve> is the waveform data in ASCII format. The format for ASCII data is <NR1>[,<NR1>...] where each <NR1> represents a data point.

**Examples**

```
CURVE?
```

might return, for ASCII data: CURVE

```
0,0,0,0,-1,1,0,-1,0,0,-1,0,0,-1,0,-1,
-1,1,0,0,0,-1,0,0,-1,0,1,1,0,-1,0,0,-1,0,0,-1,0,0
```

# DATa

Sets or queries the format and location of the waveform data that is transferred with the CURVe command. Since DATa:DESTination and DATa:TARget are equivalent, only DATa:DESTination is returned by the DATa? query.

**Group** Waveform

**Related Commands** CURVE, WAVFrm

**Syntax**

```
DATa { INIT | SNAp }
DATa?
```

**Arguments**

`INIT` initializes the waveform data parameters to their factory defaults.

`SNAp` sets DATa:STARt and DATa:STOP to match the current vertical bar cursor positions.

**Examples**

```
DATA SNAP
```
assigns DATA:START and DATA:STOP to the current position of the vertical bar cursors.

```
DATA?
```
might return the string `:DATA:ENCDG RPBINARY;DESTINATION REF4; SOURCE REF2;START 1;STOP 500;WIDTH 2`

## DATa:DESTination

Sets or queries the reference memory location for storing waveform data that is transferred into the oscilloscope by the CURVe command. This command is identical to the DATa:TARget command.

**Group** Waveform

**Syntax** `DATa:DESTination REF<x>`

`DATa:DESTination?`

**Arguments** `REF<x>` is the reference memory location where the waveform will be stored.

**Examples** `DATA:DESTINATION REF1`
stores incoming waveform data in reference memory 1.

`DATA:DESTINATION?`
might return `REF2` as the reference memory location that is currently selected.

## DATa:ENCdg

Sets or queries the format of the waveform data. This command is equivalent to setting WFMPre:ENCdg, WFMPre:BN_Fmt, and WFMPre:BYT_Or as shown in Table 2–22. Setting the DATa:ENCdg value causes the corresponding WFMPre values to be updated and vice versa.

**Group** Waveform

**Related Commands** WFMPre:ENCdg, WFMPre:BN.FMT, WFMPre:BYT_Or

**Syntax**

```
DATa:ENCdg { ASCIi | RIBinary | RPBinary | SRIbinary | SRPbinary}
DATa:ENCdg?
```

**Arguments**

`ASCIi` specifies the ASCII representation of signed integer (`RIBinary`) data. If this is the value at power-on, the WFMPre values for BN_Fmt, BYT_Or, and ENCdg are set as RP, MSB, and ASC respectively.

`RIBinary` specifies signed integer data-point representation with the most significant byte transferred first. This format results in the fastest data transfer rate when DATa:WIDth is set to 2.

The range is –128 to 127 when DATa:WIDth is 1. Zero is center screen. The range is –32768 to 32767 when DATa:WIDth is 2. The upper limit is one division above the top of the screen and the lower limit is one division below the bottom of the screen.

`RPBinary` specifies positive integer data-point representation with the most significant byte transferred first.

The range is 0 to 255 when DATa:WIDth is 1. 127 is center screen. The range is 0 to 65,535 when DATa:WIDth is 2. The upper limit is one division above the top of the screen and the lower limit is one division below the bottom of the screen.

`SRIbinary` is the same as `RIBinary` except that the byte order is swapped, meaning that the least significant byte is transferred first. This format is useful when transferring data to IBM compatible PCs.

`SRPbinary` is the same as `RPBinary` except that the byte order is swapped, meaning that the least significant byte is transferred first. This format is useful when transferring data to IBM compatible PCs.

**Table 2–22: DATa and WFMPre parameter settings**

| | WFMPre Settings | | |
|---|---|---|---|
| **DATa:ENCdg Setting** | **:ENCdg** | **:BN_Fmt** | **:BYT_Or** |
| ASCIi | ASC | N/A | N/A |
| RIBinary | BIN | RI | MSB |
| RPBinary | BIN | RP | MSB |
| SRIbinary | BIN | RI | LSB |
| SRPbinary | BIN | RP | LSB |

**Examples**

DATA:ENCDG RPBINARY
sets the data encoding format to be positive integer where the most significant byte is transferred first.

DATA:ENCDG?
might return SRPBINARY for the format of the waveform data.

## DATa:SOUrce

Sets or queries the location of the waveform data that is transferred from the oscilloscope by CURVe?. The source data is always transferred in a predefined order regardless of the order they are specified using this command. The predefined order is CH1, CH2, MATH, MATH1, REF1, REF2.

**Group**

Waveform

**Syntax**

DATa:SOUrce <wfm>[<Comma><wfm>]...

DATa:SOUrce?

**Arguments**

<wfm> is the source of the waveform data that is transferred from the oscilloscope to the controller.

**Examples**

DATA:SOURCE REF2, CH2, MATH1, CH1
specifies that four waveforms will be transferred in the next CURVE? query. The order that the data will be transferred is CH1, CH2, MATH1, and REF2.

DATA:SOURCE?
might return REF1, indicating the source for the waveform data that is transferred using CURVE?.

## DATa:STARt

Sets or queries the starting data point for waveform transfer. This command allows for the transfer of partial waveforms to and from the oscilloscope.

**Group** Waveform

**Related Commands** CURVe?, DATa SNAp, DATa:STOP

**Syntax**

```
DATa:STARt <NR1>
DATa:STARt?
```

**Arguments** <NR1> ranges from 1 to the record length and is the first data point that will be transferred. Data is transferred from <NR1> to DATa:STOP or the record length, whichever is less. If <NR1> is greater than the record length, then the oscilloscope transfers data until it reaches the record length. When DATa:STOP is less than DATa:STARt, the values are swapped internally for CURVe?.

**Examples**

```
DATA:START 10
```
specifies that the waveform transfer will begin with data point 10.

```
DATA:START?
```
might return 214 as the first waveform data point that will be transferred.

## DATa:STOP

Sets or queries the last data point that will be transferred when using CURVe?. This lets you transfer partial waveforms to the controller.

When using the CURVe command, the oscilloscope stops reading data when there is no more data to read or when the specified record length is reached; this command is ignored.

**Group** Waveform

**Related Commands** CURVe?, DATa SNAp

**Syntax**

```
DATa:STOP <NR1>
```

```
DATa:STOP?
```

**Arguments** <NR1> ranges from 1 to 1000 and is the last data point that will be transferred. If <NR1> is greater than the record length, then data will be transferred up to the record length. If both DATa:STARt and DATa:STOP are greater than the record length, an execution error will occur. When DATa:STOP is less than DATa:STARt, the values are swapped internally for CURVe?.

If you always want to transfer complete waveforms, set DATa:STARt to 1 and DATa:STOP to the record length (1000).

**Examples**

```
DATA:STOP 150
```
specifies that the waveform transfer will stop at data point 150.

```
DATA:STOP?
```
might return 285 as the last data point that will be transferred.

## DATa:TARget

Sets or queries the location for storing waveform data transferred to the oscilloscope using the CURVe command. This command is equivalent to the DATa:DESTINATION command and is included here for compatibility with older Tektronix instruments.

**Group** Waveform

**Related Commands** CURVe

**Syntax** `DATa:TARget REF<x>`

`DATa:TARget?`

## DATa:WIDth

Sets the number of bytes per data point in the waveform transferred using the CURVe command.

**Group** Waveform

**Related Commands** CURVe, WFMPre:BIT_Nr, WFMPre:BYT_Nr

**Syntax**

```
DATa:WIDth <NR1>
```

```
DATa:WIDth?
```

**Arguments**

<NR1> = 1 specifies that there is 1 byte (8 bits) per point. This format is useful when the acquisition mode is set to SAMple, ENVelope, or PEAKdetect. If used for AVErage, the low order byte is not transmitted.

<NR1> = 2 specifies that there are 2 bytes (16 bits) per point. This format is useful for AVErage waveforms. If used for ENVelope, PEAKdetect, or SAMple, the least significant byte is always zero.

If DATA:WIDTH is set to 2, the block is twice as long as when it is 1. The length or number of bytes in the block can be calculated by ((DATA:STOP – DATA:START)+1)*DATA:WIDTH. If DATA:START and/or DATA:STOP extend beyond the limits of the waveform the number of bytes will be less.

**Examples**

```
DATA:WIDth 1
```

sets the data width to 1 byte per data point for CURVe data.

# DATE

Sets or queries the date that the oscilloscope can display.

**Group** Miscellaneous

**Related Commands** DISplay:CLOCk, TIMe

**Syntax**

```
DATE <QString>
DATE?
```

**Arguments**

<QString> is a date in the form "yyyy-mm-dd".
mm refers to a two-digit month number from 01 to 12.
dd refers to a two-digit day number in the month.
yyyy refers to a four-digit year number.
There must a dash (–) after the yyyy and after the mm.

**Examples**

```
DATE "1996-01-24"
```

specifies that the date is set to January 24th, 1996.

# *DDT

Allows the user to specify a command or a list of commands that are executed when the oscilloscope receives a *TRG command or the GET GPIB interface message. This is a special alias that *TRG uses.

**Group** Miscellaneous

**Related Commands** ALIAS:DEFINE, *TRG, Get GPIB interface message

**Syntax**

```
*DDT { <Block> | <QString> }
```

```
*DDT?
```

**Arguments** <Block> or <QString> is a complete sequence of program messages. The messages must contain only valid commands that must be separated by semicolons and must follow all rules for concatenating commands (see page 2–4). The sequence must be ≤80 characters. <Block> format is always returned as a query response.

**Examples**

```
*DDT #217ACQUIRE:STATE RUN<EOI>
```

specifies that the acquisition system will be started each time a *TRG command is sent.

# DESE

Sets or queries the bits in the Device Event Status Enable Register (DESER). The DESER is the mask that determines whether or not events are reported to the Standard Event Status Register (SESR), and entered into the Event Queue. For a more detailed discussion of the use of these registers, see page 3–1.

**Group** Status and Error

**Related Commands** *CLS, *ESE, *ESR?, EVENT?, EVMsg?, *SRE, *STB?

**Syntax**

```
DESE <NR1>
DESE?
```

**Arguments** <NR1> is a value in the range from 0 to 255. The binary bits of DESER are set according to this value. For example, DESE 209 sets the DESER to the binary value 11010001 (that is, the most significant bit in the register is set to 1, the next most significant bit to 1, the next bit to 0, and so on).

The power-on default for DESER is all bits set if *PSC is 1. If *PSC is 0, the DESER maintains its value through a power cycle.

***NOTE.*** *Setting DESER and ESER to the same value allows only those codes to be entered into the Event Queue and summarized on the ESB bit (bit 5) of the Status Byte Register. Use the* *ESE *command to set ESER. A discussion of event handling begins on page 3–1.*

**Examples**

```
DESE 209
```

sets the DESER to binary 11010001, which enables the PON, URQ, EXE, and OPC bits.

```
DESE?
```

might return the string :DESE 186, showing that DESER contains the binary value 10111010.

## DIAg:RESUlt:FLAg? (Query Only)

Returns the pass/fail status from the last diagnostic test sequence execution. Used the DIAg:RESUlt:LOG? query to determine which test(s) has failed.

**Group** Calibration and Diagnostic

**Related Commands** DIAg:RESUlt:LOG?

**Syntax** `DIAg:RESUlt:FLAg?`

**Returns** `PASS` indicating that all of the selected diagnostic tests have passed.`FAIl` indicating that at least one of the selected diagnostic tests have failed.

**Examples** `DIAG:RESULT:FLAG?`
returns either `PASS` or `FAIl`.

## DIAg:RESUlt:LOG? (Query Only)

Returns the internal results log from the last diagnostic test sequence execution. The list contains all modules and module interfaces that were tested along with the pass/fail status of each.

**Group** Calibration and Diagnostic

**Related Commands** DIAg:RESUlt:FLAg?

**Syntax** `DIAg:RESUlt:LOG?`

**Returns** `<QString>` in the following format:

`<Status>,<Module name>[,<Status>,<Module name>...]`

**Examples** `DIAG:RESULT:LOG?`
might return `:DIAG:RESULT:LOG "pass--Processor,pass--Display, pass--FP/Proc Interface,fail--Front Panel"`

## DIAg:SELect:ALL (No Query Form)

Specifies that all system test sequences will be run when the DIAg:STATE EXECUte command is sent.

**Group** Calibration and Diagnostic

**Syntax** `DIAg:SELect:ALL ALL`

**Arguments** `ALL` selects functional, memory, and register tests for the acquisition, processor and display systems, and self diagnostics for the front panel.

## DIAg:STATE (No Query Form)

Executes the diagnostic tests specified by the DIAg:SELect command.

When the test sequence has completed, any of the modules or module interfaces that failed diagnostics are displayed on the screen and stored in an internal log file. The pass/fail status is returned by DIAg:RESUlt:FLAg? and the internal log is returned by DIAg:RESUlt:LOG?. This command is equivalent to running Extended Diagnostics by selecting **Execute** in the Utility menu when **System** is set to Diag.

***NOTE.*** *The DIAg:STATE EXECute command can take 30 seconds or more to respond.*

**Group** Calibration and Diagnostic

**Syntax** `DIAg:STATE EXECute`

**Arguments** `EXECute` runs the diagnostic test sequences specified by the DIAg:SELect command. When complete, the oscilloscope returns to the state it was in just prior to the test. The pass/fail status of the tests can be returned by executing DIAg:RESUlt:FLAg?.

The DIAg:STATE EXECute command clears the Event Queue, the Input Queue, and the Status Registers (SESR and SBR).

**Examples** `DIAg:STATE EXECUTE`
executes all the diagnostic tests that have been selected.

## DISplay? (Query Only)

Returns the current display settings.

**Group** Display

**Syntax** `DISplay?`

**Examples** `DISPLAY?`
might return `:DISPLAY:FORMAT YT;STYLE VECTORS;PERSISTENCE 5.0E-1;GRATICULE FULL;TRIGT 1;INTENSITY:OVERALL 85;WAVEFORM 70;TEXT 60;CONTRAST 150;CLOCK 1`

## DISplay:CLOCk

Controls the display of the date and time. This is equivalent to setting the **Display Date/Time** in the Readout Options side menu. The query form returns an ON (1) or an OFF (0).

**Group** Display

**Syntax** `DISplay:CLOCk { OFF | ON | <NR1> }`

`DISplay:CLOCk?`

**Arguments** `<OFF>` or `<NR1>` = 0 removes the clock from the display.

`<ON>` or `<NR1>` $\neq$ 0 displays the clock on the display.

**Examples** `DISPLAY:CLOCK ON`
sets the display to show time and date.

`DISPLAY:CLOCK?`
might return 1 indicating that the display shows time and date.

## DISplay:FORMat

Sets or queries the display format. This command is equivalent to setting **Format** in the Display menu.

**Group** Display

**Syntax**

```
DISplay:FORMat { XY | YT }
```

```
DISplay:FORMat?
```

**Arguments**

XY displays the voltage of one waveform against the voltage of another. The sources that make up an XY waveform are predefined and are listed in Table 2–23. Displaying one source causes its corresponding source to be displayed.

**Table 2–23: XY format pairs**

| X-Axis Source | Y-Axis Source |
|---|---|
| Ch1 | Ch2 |
| Ref1 | Ref2 |

YT sets the display to a voltage versus time format and is the normal mode.

**Examples**

```
DISPLAY:FORMAT YT
```
selects a voltage versus time format for the display.

```
DISPLAY:FORMAT?
```
might return XY for the display format.

## DISplay:GRAticule

Selects the type of graticule that is displayed. This command is equivalent to setting **Graticule** in the Display menu.

**Group** Display

**Syntax**

```
DISplay:GRAticule { CROSSHair | FRAme | FULl | GRId }
DISplay:GRAticule?
```

**Arguments**

`CROSSHair` specifies a frame and cross hairs only.

`FRAme` specifies just a frame.

`FULl` specifies a frame, a grid, and cross hairs.

`GRId` specifies a frame and grid only.

**Examples**

```
DISPLAY:GRATICULE FRAME
```
sets the graticule type to display the frame only.

```
DISPLAY:GRATICULE?
```
returns `FULL` when all graticule elements (grid, frame, and cross hairs) are selected.

## DISplay:INTENSITy? (Query Only)

Returns the current intensity settings for different parts of the display.

**Group** Display

**Syntax**

```
DISplay:INTENSITy?
```

**Examples**

```
DISPLAY:INTENSITY?
```
might return `:DISPLAY:INTENSITY:OVERALL 85;WAVEFORM BRI;TEXT DIM;CONTRAST 150`

## DISplay:INTENSITy:CONTRast

Sets the intensity of the intensified zone on a waveform. It also sets the intensity of BRIGHT versus DIM. This command is equivalent to setting **Contrast** in the Display Intensity side menu.

**Group** Display

**Related Commands** HORizontal:MODe

**Syntax** DISplay:INTENSITy:CONTRast <NR1>

DISplay:INTENSITy:CONTRast?

**Arguments** <NR1> ranges from 100% to 250%.

**Examples** DISPLAY:INTENSITY:CONTRAST 140
sets the intensity of the intensified portion of a waveform and other bright parts of the display to 140% of normal.

## DISplay:INTENSITy:OVERAll

Sets the intensity of the entire display. This command is equivalent to setting **Overall** in the Display Intensity side menu.

**Group** Display

**Syntax** DISplay:INTENSITy:OVERAll <NR1>

DISplay:INTENSITy:OVERAll?

**Arguments** <NR1> ranges from 20% to 100%.

**Examples** DISplay:INTENSITY:OVERALL 50
sets the intensity of the display to the middle of the range.

DISplay:INTENSITY:OVERALL?
might return 75 as the overall display intensity.

## DISplay:INTENSITy:TEXt

Sets the intensity of the text and the graticule. This command is equivalent to setting **Text/Grat** in the Display Intensity side menu.

**Group** Display

**Syntax**

```
DISplay:INTENSITy:TEXt { DIM | BRIght }
```

```
DISplay:INTENSITy:TEXt?
```

**Arguments**

`DIM` sets the intensity equal to the overall intensity.

`BRIght` sets the intensity equal to the contrast setting (100% to 250% of the overall intensity).

**Examples**

```
DISPLAY:INTENSITY:TEXT BRIght
```

sets the intensity of the text to the brightest level.

## DISplay:INTENSITy:WAVEform

Sets the intensity of the waveforms. This command is equivalent to setting **Waveform** in the Display Intensity side menu.

**Group** Display

**Syntax**

```
DISplay:INTENSITy:WAVEform { DIM | BRIght }
```

```
DISplay:INTENSITy:WAVEform?
```

**Arguments**

`DIM` sets the intensity equal to the overall intensity.

`BRIght` sets the intensity equal to the contrast setting (100% to 250% of the overall intensity).

**Examples**

```
DISPLAY:INTENSITY:WAVEFORM?
```

might return `DIM`, indicating that the waveform intensity is equal to the overall intensity.

## DISplay:PERSistence

Sets the length of time that dots (or vectors) are displayed when DISplay:STYle is set to ACCUMDOTS or ACCUMVECTORS.

**Group** Display

**Related Commands** DISplay:STYle

**Syntax**

```
DISplay:PERSistence { <NR3> }
```

```
DISplay:PERSistence?
```

**Arguments** <NR3> specifies the length, in seconds, that the waveform points are displayed on the screen. The range is 100 ms to 10 s.

0 specifies infinite persistence.

**Examples**

```
DISPLAY:PERSISTENCE 3
```

specifies that the waveform points are displayed on the screen for 3 seconds before they fade.

## DISplay:STYle

Selects how the data is displayed. This command is equivalent to setting **Style** in the Display menu.

**Group** Display

**Related Commands** DISplay:PERSistence

**Syntax**

```
DISplay:STYle { DOTs | ACCUMDots | ACCUMVectors |
VECtors }
```

```
DISplay:STYle?
```

**Arguments**

`DOTs` displays individual data points.

`ACCUMDots` accumulates data points on the display until the PERSistence time is met.

`VECtors` connects adjacent data points. Old points are immediately replaced by new ones.

`ACCUMVectors` accumulates data points with a line vector waveform until the PERSistence time is met.

**Examples**

```
DISPLAY:STYLE VEC
```
sets the display to connect adjacent data points.

```
DISPLAY:STYLE?
```
might return `DOTS` indicating that the display shows individual waveform data points.

## DISplay:TRIGT

Controls the display of the trigger indicator. This is equivalent to setting the **Display 'T' @ Trigger Point** in the Readout Options side menu. The query form returns an ON (1) or an OFF (0).

**Group** Display

**Syntax**

```
DISplay:TRIGT { OFF | ON | <NR1> }
```

```
DISplay:TRIGT?
```

**Arguments**

<OFF> or <NR1> = 0 removes the trigger indicator from the display.

<ON> or <NR1> $\neq$ 0 displays a trigger indicator on each of the displayed waveforms. The trigger indicator is in reverse video for the selected waveform.

**Examples**

```
DISPLAY:TRIGT ON
```
sets the display to show trigger indicators.

```
DISPLAY:TRIGT?
```
might return 1 indicating that the display shows trigger indicators.

# *ESE

Sets or queries the bits in the Event Status Enable Register (ESER). The ESER prevents events from being reported to the Status Byte Register (STB). For a more detailed discussion of the use of these registers, see page 3–1.

**Group** Status and Error

**Related Commands** *CLS, DESE, *ESR?, EVENT?, EVMsg? *SRE, *STB?

**Syntax** `*ESE <NR1>`

`*ESE?`

**Arguments** `<NR1>` is a value in the range from 0 through 255. The binary bits of the ESER are set according to this value.

The power-on default for ESER is 0 if `*PSC` is 1. If `*PSC` is 0, the ESER maintains its value through a power cycle.

***NOTE**. Setting the DESER and the ESER to the same value allows only those codes to be entered into the Event Queue and summarized on the ESB bit (bit 5) of the Status Byte Register. Use the* `DESE` *command to set the DESER. A discussion of event handling begins on page 3–1.*

**Examples** `*ESE 209`
sets the ESER to binary 11010001, which enables the PON, URQ, EXE, and OPC bits.

`*ESE?`
might return the string `*ESE 186`, showing that the ESER contains the binary value 10111010.

## *ESR? (Query Only)

Returns the contents of the Standard Event Status Register (SESR). *ESR? also clears the SESR (since reading the SESR clears it). For a more detailed discussion of the use of these registers see page 3–1.

**Group** Status and Error

**Related Commands** ALLEv?, *CLS, DESE, *ESE, EVENT?, EVMsg?, *SRE, *STB?

**Syntax** `*ESR?`

**Examples** `*ESR?`
might return the value 213, showing that the SESR contains binary 11010101.

## EVENT? (Query Only)

Returns from the Event Queue an event code that provides information about the results of the last *ESR? read. EVENT? also removes the returned value from the Event Queue. A discussion of event handling begins on page 3–1.

**Group** Status and Error

**Related Commands** ALLEv?, *CLS, DESE, *ESE, *ESR?, EVMsg?, *SRE, *STB?

**Syntax** `EVENT?`

**Examples** `EVENT?`
might return the response `:EVENT 110`, showing that there was an error in a command header.

## EVMsg? (Query Only)

Removes from the Event Queue a single event code associated with the results of the last *ESR? read, and returns the event code along with an explanatory message. A more detailed discussion of event handling begins on page 3–1.

**Group** Status and Error

**Related Commands** ALLEv?, *CLS, DESE, *ESE, *ESR?, EVENT?, *SRE, *STB?

**Syntax** `EVMsg?`

**Returns** The event code and message in the following format:

`<Event Code><Comma><QString>[<Event Code><Comma><QString>...]`

`<QString>::= <Message>;[<Command>]`

where `<Command>` is the command that caused the error and may be returned when a command error is detected by the oscilloscope. As much of the command as possible is returned without exceeding the 60 character limit of the `<Message>` and `<Command>` strings combined. The command string is right-justified.

**Examples** `EVMSG?`
might return the message `:EVMSG 110,"Command header error"`.

## EVQty? (Query Only)

Returns the number of event codes that are in the Event Queue. This is useful when using ALLEv? since it lets you know exactly how many events will be returned.

**Group** Status and Error

**Related Commands** ALLEv?, EVENT?, EVMsg?

**Syntax** `EVQty?`

**Returns** `<NR1>`

**Examples** `EVQTY?`
might return 3 as the number of event codes in the Event Queue.

## EXT:PRObe? (Query Only)

Returns the attenuation factor of the probe that is attached to the external trigger connector.

**Group** Vertical

**Syntax** `EXT:PRObe?`

**Returns** `<NR3>`

**Examples** `EXT:PROBE?`
might return `1.0E1` for a 10x probe.

## FACtory (No Query Form)

Resets the oscilloscope to its factory default settings (see *Appendix D*). The FACtory command does everything that the *RST command does.

**Group** Miscellaneous

**Related Commands** *PSC, *RCL, RECAll:SETUp, *RST, *SAV, SAVe:SETUp, TEKSecure

**Syntax** `FACtory`

## FILESystem:COPy (No query form)

Copies a named file or files to a new file. The new file can be in a totally separate directory than the old file. Also, you can use wild card characters (*.*) to copy multiple files with one command.

**Group** File system

**Syntax** `FILESystem:COPy { <source file path>,<destination file path> | <source directory path>,<destination file path> | <source directory path>,<destination directory path> }`

**Arguments** <file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>/<dir>/<filename>. <drive> and one or more <dir>s are optional. If you do not specify them, the oscilloscope will copy the file in the current directory. <filename> stands for a filename of up to 8 characters and can be followed by a period (".") and a 3-character extension. You can also use the inclusive filename *.* in the source file path to copy all files.

<directory path> is a quoted string that defines the directory. Input the directory using the form <drive>/<dir>/<directory name>. <drive> and one or more <dir>s are optional. If you do not specify them, the oscilloscope will copy the directory in the current directory. <directory name> stands for a directory name of up to 8 characters and can be followed by a period (".") and a 3-character extension.

**Examples** FILESYSTEM:COPY "TEK00001.SET","fd0:/TEK00001.SET"
copies the file named TEK00001.SET on the current drive to a file named TEK00001.SET on the drive fd0: in the root directory.

FILESYSTEM:COPY "fd0:/YOURDIR/TEK00001.SET","fd0:/MYDIR"
copies the file named TEK00001.SET on the fd0: drive and the YOURDIR directory to the MYDIR directory on the same drive.

FILESYSTEM:COPY "YOURDIR","fd0:/MYDIR"
copies the files in the YOURDIR directory in the current directory to the MYDIR directory on the fd0: drive.

## FILESystem:CWD

Sets or queries the path of the current working directory (CWD).

**Group** File system

**Syntax** FILESystem:CWD <directory path>

FILESystem:CWD?

**Arguments** <directory path> is a quoted string that defines the directory name and path.

**Examples** FILESYSTEM:CWD "fd0:/MYDIR"
will set fd0:/MYDIR as the current directory.

FILESYSTEM:CWD?
might return fd0:/MYDIR if that is the current directory.

## FILESystem:DELEte (No query form)

Deletes a named file.

**Group** File system

**Syntax** `FILESystem:DELEte <file path>`

**Arguments** `<file path>` is a quoted string that defines the file name and path. Input the file path using the form `<drive>/<dir>/<filename>`. `<drive>` and one or more `<dir>`s are optional. If you do not specify them, the oscilloscope will delete the file in the current directory. `<filename>` stands for a filename of up to 8 characters and can be followed by a period (".") and a 3-character extension. You can also use the inclusive filename `*.*` to delete all files.

**Examples** `FILESYSTEM:DELETE "NOT-MINE.SET"`
deletes the file named `NOT-MINE.SET` on the default drive and directory.

`FILESYSTEM:DELETE "*.*"`
deletes all the files in the default directory on the default drive.

## FILESystem:DELWarn

Turns on or off the front panel file delete warning. No warning is returned from the GPIB or RS-232 interface.

**Group** File system

**Syntax** `FILESystem:DELWarn { ON | OFF | <NR1> }`

`FILESystem:DELWarn?`

**Arguments** `ON` or `<NR1>` $\neq$ `0` turns on the front panel delete warning.

`OFF` or `<NR1> = 0` turns off the front panel delete warning.

**Examples** `FILESYSTEM:DELWARN OFF`
disables the front panel delete warning.

`FILESYSTEM:DELWARN?`
might return `0` indicating the front panel warning is disabled.

## FILESystem:DIR (Query only)

Returns a list of quoted strings. Each string contains the name of a file or directory in the current directory.

**Group** File system

**Syntax** `FILESystem:DIR?`

**Examples** `FILESYSTEM:DIR?`
returns a list of files and directories in the default directory.

## FILESystem:FORMat (No query form)

Formats a named drive.

**Group** File system

**Syntax** `FILESystem:FORMat <drive name>`

**Arguments** `<drive name>` is a quoted string that defines the disk drive to format. `fd0:` refers to the floppy-disk drive built into the oscilloscope.

**Examples** `FILESYSTEM:FORMAT "fd0:"`
formats the media on drive fd0:.

## FILESystem:FREESpace? (Query only)

Returns the amount of freespace (in bytes) on the current drive.

**Group** File system

**Syntax** `FILESystem:FREESpace?`

**Examples** `FILESYSTEM:FREESpace?`
might return 0 as the amount of freespace available if the disk was full.

## FILESystem:MKDir (No query form)

Make a new directory.

**Group** File system

**Syntax** FILESystem:MKDir <directory path>

**Arguments** <directory path> is a quoted string that defines the directory. Input the directory using the form <drive>/<dir>/<directory name>. <drive> and one or more <dir>s are optional. If you do not specify them, the oscilloscope will create the directory in the current directory. <directory name> stands for a directory name of up to 8 characters and can be followed by a period (".") and a 3-char extension.

**Examples** FILESYSTEM:MKDIR "NEATPICS"
creates the directory named NEATPICS on the current drive.

## FILESystem:OVERWrite

Turns on or off the file overwrite protection. Turning on file overwrite protection prevents writing over existing files.

**Group** File system

**Syntax** FILESystem:OVERWrite { ON | OFF | <NR1> }

FILESystem:OVERWrite?

**Arguments** ON or <NR1> $\neq$ 0 turns on the file overwrite protection.

OFF or <NR1> = 0 turns off the file overwrite protection.

**Examples** FILESYSTEM:OVERWRITE OFF
lets you overwrite existing files.

FILESYSTEM:OVERWRITE?
might return 0 indicating you cannot overwrite existing files.

## FILESystem:PRInt (No query form)

***Option 14 Communications Interface Only***

Prints a named file to the named port.

**Group** File system

**Syntax** `FILESystem:PRInt <filepath>,{ GPIb | RS232 | CENtronics }`

**Arguments** `<file path>` is a quoted string that defines the file name and path. Input the file path using the form `<drive>/<dir>/<filename>`. `<drive>` and one or more `<dir>`s are optional. If you do not specify them, the oscilloscope will print the file in the current directory. `<filename>` stands for a filename of up to 8 characters and can be followed by a period (".") and a 3-character extension.

`GPIb` specifies that the hardcopy is sent out the GPIB port (Option 14 Communications Interface only).

`CENtronics` specifies that the hardcopy is sent out the Centronics port (Option 14 Communications Interface only).

`RS232` specifies that the hardcopy is sent out the RS232 port (Option 14 Communications Interface only).

**Examples** `FILESYSTEM:PRINT "TEK00000.IBM",CENTRONICS`
sends the file named `TEK00000.IBM` out the Centronics port.

## FILESystem:REName (No query form)

Assigns a new name to a file.

**Group** File system

**Syntax** `FILESystem:REName <old file path>,<new file path>`

**Arguments** <old file path> is a quoted string that defines the file to rename. Input the file path using the form <drive>/<dir>/<filename>. <drive> and one or more <dir>s are optional. If you do not specify them, the oscilloscope will look for the filename in the current directory.

<new file path> is a quoted string that defines the new name of the file. Input the file path using the form <drive>/<dir>/<filename>. <drive> and one or more <dir>s are optional. If you do not specify them, the oscilloscope will place the newly named file in the current directory.

**Examples** FILESYSTEM:RENAME "TEK00000.SET","MYSETTING.SET"
gives the file named TEK00000.SET the new name of MYSETTING.SET. The file remains on the current directory.

## FILESystem:RMDir (No query form)

Deletes a named directory.

**Group** File system

**Syntax** FILESystem:RMDir <directory path>

**Arguments** <directory path> is a quoted string that defines the directory. Input the directory using the form <drive>/<dir>/<directory name>. <drive> and one or more <dir>s are optional. If you do not specify them, the oscilloscope will delete the directory in the current directory. <directory name> stands for a directory name of up to 8 characters and can be followed by a period (".") and a 3-character extension.

**Examples** FILESYSTEM:RMDIR "NEATPICS"
deletes the directory named NEATPICS in the current directory.

## HARDCopy

Sends a copy of the screen display followed by an EOI to the port specified by HARDCopy:PORT. The format and layout of the output is specified with the HARDCopy:FORMat and HARDCopy:LAYout commands. This command is equivalent to pressing the front-panel **HARDCOPY** button.

HARDCopy? returns format, layout, and port information.

***NOTE.*** *This command is NOT IEEE Std 488.2-1987 compatible.*

**Group** Hardcopy

**Syntax**

```
HARDCopy { ABOrt | CLEARSpool | STARt }
HARDCopy?
```

**Arguments**

`ABOrt` terminates the hardcopy output in process.

***NOTE.*** *DCL does NOT clear the output queue once a hardcopy is in process. The only way to abort the hardcopy process is to send the HARDCopy ABOrt command. The output queue can then be cleared using DCL.*

`CLEARSpool` clears the printer output spooler.

`STARt` initiates a screen copy that is sent to the controller where it can be stored in a file or redirected to a printing device.

***NOTE.*** *Use the *WAI command between HARDCopy STARt commands to ensure that the first hardcopy is complete before starting another.*

**Examples**

```
HARDCOPY ABORT
```

stops any hardcopy output that is in process.

## HARDCopy:FILEName

Selects the file to send the hardcopy data to on the next hardcopy command (HARDCOPY START). This is equivalent to setting the target file name in the Hardcopy menu.

**Group** Hardcopy

**Related Commands** HARDCopy

**Syntax**

```
HARDCopy:FILEName <file path>
```

```
HARDCopy:FILEName?
```

**Arguments** <file path> specifies that the hardcopy is sent to the named file. <file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>/<dir>/<filename>.<drive> and one or more <dir>s are optional. If you do not specify them, the oscilloscope will write the file to the current directory. <filename> stands for a filename of up to 8 characters followed by a period (".") and any 3-character extension.

You can automatically create different names for files. You do this by using the question mark (?) as a special wildcard character. These stand for numbers the oscilloscope will insert sequentially in the filename. For example, if you placed two question marks at the end of the filename then the oscilloscope would append 00 to the first file created, 01 to the next, and 02 to the next. This helps you automatically create different names for files. It is particularly useful in automated testing situations.

**Examples**

```
HARDCOPY:FILENAME "TEK.IBM"
```
selects TEK.IBM as the selected file name.

```
HARDCOPY:FILENAME?
```
might return TEK.IBM as the selected file name.

```
HARDCOPY:FILENAME "TEK??.IBM"
```
selects TEK as the selected file name with a numeric, two-digit suffix. The oscilloscope might return TEK00.IBM as the first file, TEK01.IBM as the second.

## HARDCopy:FORMat

Selects the output data format for hard copies. This is equivalent to setting **Format** in the Hardcopy menu.

**Group** Hardcopy

**Syntax**

```
HARDCopy:FORMat { BMP | DESKJet | DPU411 | DPU412 | EPSImage |
EPSOn | INTERLeaf | LASERJet | PCX | THInkjet | TIFf }
```

```
HARDCopy:FORMat?
```

**Arguments**

`BMP` sets the format to Microsoft Windows file format.

`DESKjet` sets the format to high-resolution printer format.

`DPU411` sets the format to thermal printer format for DPU 411/II or HC 411.

`DPU412` sets the format to thermal printer format for DPU 412.

`EPSImage` sets the format to encapsulated postscript image file format.

`EPSOn` sets the format to 9-pin or 24-pin dot matrix printer format.

`INTERLeaf` sets the format to Interleaf image object file format.

`LASERJet` sets the format to laser printer format.

`PCX` sets the format to PC Paintbrush monochrome image file format.

`THInkjet` sets the format to inkjet printer format.

`TIFf` sets the format to tag image file format.

**Examples**

```
HARDCOPY:FORMAT TIFf
```
sets the hardcopy output format to TIFF.

```
HARDCOPY:FORMAT?
```
might return `INTERLEAF` as the hardcopy output format.

## HARDCopy:LAYout

Selects the printing orientation. This is equivalent to setting **Layout** in the Hardcopy menu.

**Group** Hardcopy

**Syntax** `HARDCopy:LAYout { LANDscape | PORTRait }`

`HARDCopy:LAYout?`

**Arguments** `LANDscape` specifies that the bottom of the hardcopy is along the long side of the page.

`PORTRait` specifies that the bottom of the hardcopy is along the short side of the page. This is the standard format.

**Examples** `HARDCOPY:LAYOUT?`
might return `PORTRAIT` as the page layout format of the hardcopy output.

## HARDCopy:PORT

Selects the output port for the printer. This is equivalent to setting **Port** in the Hardcopy menu.

**Group** Hardcopy

**Related Commands** HARDCopy

**Syntax** `HARDCopy:PORT { GPIb | CENtronics | RS232 | FILE }`

`HARDCopy:PORT?`

`GPIb` specifies that the hard copy is sent out the GPIB port.

`CENtronics` specifies that the hard copy is sent out the Centronics port.

`RS232` specifies that the hard copy is sent out the RS232 port.

`FILE` specifies that the hard copy is sent to a file.

**Examples** `HARDCOPY:PORT?`
might return `GPIB` as the selected hardcopy output port.

## HDR

This command is identical to the HEADer query and is included for compatibility with older Tektronix instruments.

**Group** Miscellaneous

**Syntax** HDR { <NR1> | OFF | ON }

HDR?

## HEADer

Sets or queries the Response Header Enable State that causes the oscilloscope to either include or omit headers on query responses. This command does not affect IEEE Std. 488.2-1987 Common Commands (those starting with an asterisk); they never return headers.

**Group** Miscellaneous

**Related Commands** VERBose

**Syntax** HEADer { <NR1> | OFF | ON }

HEADer?

**Arguments** ON or <NR1> $\neq$ 0 sets the Response Header Enable State to true. This causes the oscilloscope to include headers on applicable query responses. You can then use the query response as a command.

OFF or <NR1> = 0 sets the Response Header Enable State to false. This causes the oscilloscope to omit headers on query responses so that only the argument is returned.

**Examples** HEADER OFF
causes the oscilloscope to omit headers from query responses.

HEADER?
might return the value 1, showing that the Response Header Enable State is true.

## HORizontal? (Query Only)

Returns all settings for the horizontal commands. The commands HORizontal:MAIn:SCAle, HORizontal:MAIn:SECdiv, HORizontal:SCAle, and HORizontal:SECdiv are equivalent, so HORizontal:MAIn:SCAle is the only value that is returned.

**Group** Horizontal

**Syntax** `HORizontal?`

**Examples** `HORIZONTAL?`
might return the string `:HORIZONTAL:MODE MAIN;RECORDLENGTH 1000;POSITION 5.0E0;TRIGGER:POSITION 50;:HORIZONTAL:MAIN:SCALE 1.0E-6;:HORIZONTAL:DELAY:MODE RUNSAFTER;SCALE 1.0E-6;TIME:RUNSAFTER 1.6E-8;:HORIZONTAL:REF1 LOCK;REF2 LOCK;FITTOSCREEN 0`

## HORizontal:DELay? (Query Only)

Returns all horizontal delayed time base parameters. The commands HORizontal:DELay:SECdiv and HORizontal:DELay:SCAle are identical, so only HORizontal:DELay:SCAle is returned.

**Group** Horizontal

**Related Commands** HORizontal?, HORizontal:DELay:MODe?, HORizontal:DELay:SCAle?, HORizontal:DELay:SECdiv?, HORizontal:DELay:TIMe?

**Syntax** `HORizontal:DELay?`

**Examples** `HORIZONTAL:DELAY?`
might return the delay parameters `:HORIZONTAL:DELAY:MODE RUNSAFTER;SCALE 1.0E-6;TIME:RUNSAFTER 1.6E-8`

## HORizontal:DELay:MODe

Included for compatibility purposes only.

**Group** Horizontal

**Syntax**

```
HORizontal:DELay:MODe RUNSAfter
HORizontal:DELay:MODe?
```

**Arguments** `RUNSAfter` specifies that the delayed time base runs a user-specified amount of delay time after the main trigger event.

## HORizontal:DELay:SCAle

Sets the time per division for the delayed time base. This is equivalent to setting **SCALE** for the delayed time base.

**Group** Horizontal

**Related Commands** HORizontal:DELay:SECdiv

**Syntax**

```
HORizontal:DELay:SCAle <NR3>
HORizontal:DELay:SCAle?
```

**Arguments** `<NR3>` is the time per division. The range is 5 ns (TDS 340A), 2.5 ns (TDS 360), or 2.5 ns (TDS 380) to 5 s in a 1-2-5 sequence. Values that are not in a 1-2-5 sequence are set to the closest valid value. If the delayed time base scale is set slower than the main time base scale, both the main and delayed time base scales are set to the delay scale value.

**Examples**

```
HORIZONTAL:DELAY:SCALE 2.0E-6
```
sets the delay scale to 2 µs per division.

```
HORIZONTAL:DELAY:SCALE 9.0E-6
```
sets the delay scale to 10 µs per division. Since 9 µs is not a valid value within the 1-2-5 sequence, it is automatically set to the closest valid value.

```
HORIZONTAL:DELAY:SCALE?
```
might return `1.0E-3`, indicating that the delay time is 1 ms per division.

## HORizontal:DELay:SECdiv

This command is identical to the HORizontal:DELay:SCAle command. It is provided to maintain program compatibility with some older models of Tektronix oscilloscopes.

**Group** Horizontal

**Syntax**

```
HORizontal:DELay:SECdiv <NR3>
HORizontal:DELay:SECdiv?
```

## HORizontal:DELay:TIMe

Sets or queries the delay time to wait after the main trigger before the delayed time base begins. This is equivalent to setting **Delayed Runs After Main** in the Horizontal **Time Base** side menu.

**Group** Horizontal

**Related Commands** HORizontal:DELay:MODe, HORizontal:DELay:TIMe:RUNSAfter

**Syntax**

```
HORizontal:DELay:TIMe <NR3>
HORizontal:DELay:TIMe?
```

**Arguments** <NR3> is the time, in seconds, between the main trigger and the delayed trigger. The range is from one acquired sample interval to 50 s. Resolution depends on the time base setting (see Table 2–24).

**Table 2–24: Horizontal delay time resolution**

| Time Base Setting[1] | Delay Time Resolution |
|---|---|
| 1 µs or faster | 16.5 ns |
| 2.5 µs | 49.5 ns |
| 5 µs | 99 ns |

**Table 2–24: Horizontal delay time resolution (Cont.)**

| Time Base Setting[1] | Delay Time Resolution |
|---|---|
| 10 µs | 198 ns |
| slower than 10 µs | one sample interval (0.02 × the delayed time base setting) |

1 **When the horizontal delay mode is main only or intensified, use the main timebase setting. When the horizontal delay mode is delayed only, use the delay timebase setting.**

**Examples**

`HORIZONTAL:DELAY:TIME 2.0E-3`
sets the delay time between the main and delayed time base to 2 ms.

## HORizontal:DELay:TIMe:RUNSAfter

Sets or queries the delay time to wait after the main trigger before the delayed time base begins. This is equivalent to setting **Delayed Runs After Main** in the Horizontal **Time Base** side menu. This command is equivalent to the HORizontal:DELay:TIMe command.

**Group**

Horizontal

**Related Commands**

HORizontal:DELay:MODe, HORizontal:DELay:TIMe

**Syntax**

`HORizontal:DELay:TIMe:RUNSAfter <NR3>`

`HORizontal:DELay:TIMe:RUNSAfter?`

**Arguments**

`<NR3>`, see HORizontal DELay:TIMe.

**Examples**

`HORIZONTAL:DELAY:TIME:RUNSAFTER 2.0E-3`
sets the delay time between the main and delayed time base to 2 ms.

## HORizontal:FITtoscreen

Setup horizontal waveform compress operation. This command is equivalent to setting **Fit To Screen** in the Horizontal menu. Waveform compress lets you fit a captured waveform to the visible screen.

**Group** Horizontal

**Syntax**

```
HORizontal:FITtoscreen { <NR1> | OFF | ON }
HORizontal:FITtoscreen?
```

**Arguments**

`<NR1>` indicates OFF if it's a 0. It indicates ON if it's a non-zero value.

`ON` means turn on waveform compress.

`OFF` means turn off waveform compress.

**Examples**

```
HORIZONTAL:FITTOSCREEN ON
```

turns on waveform compress.

## HORizontal:MAIn? (Query Only)

Returns the time per division of the main time base. The commands HORizontal:MAIn:SECdiv and HORizontal:MAIn:SCAle are identical, so only HORizontal:MAIn:SCAle is returned.

**Group** Horizontal

**Related Commands** HORizontal:SCAle, HORizontal:SECdiv, HORizontal:MAIn:SECdiv

**Syntax**

```
HORizontal:MAIn?
```

**Examples**

```
HORIZONTAL:MAIN?
```

might return `:HORIZONTAL:MAIN:SCALE 1.0E-6`.

## HORizontal:MAIn:SCAle

Sets the time per division for the main time base. This command is equivalent to setting **SCALE** for the main time base.

**Group** Horizontal

**Related Commands** HORizontal:DELay:SCAle, HORizontal:DELay:SECdiv, HORizontal:MAIn:SECdiv

**Syntax** `HORizontal:MAIn:SCAle <NR3>`

`HORizontal:MAIn:SCAle?`

**Arguments** `<NR3>` is the time per division. The range is 5 ns (TDS 340A), 2.5 ns (TDS 360), or 2.5 ns (TDS 380) to 5 s in a 1-2-5 sequence. Values that are not in a 1-2-5 sequence are set to the closest valid value.

**Examples** `HORIZONTAL:MAIN:SCALE 2E-6`
sets the main scale to 2 μs per division.

## HORizontal:MAIn:SECdiv

Sets the time per division for the main time base. This command is identical to the HORizontal:MAIn:SCAle command. It is provided to maintain program compatibility with some older models of Tektronix oscilloscopes.

**Group** Horizontal

**Related Commands** HORizontal:DELay:SCAle, HORizontal:DELay:SECdiv, HORizontal:MAIn:SCAle

**Syntax** `HORizontal:MAIn:SECdiv <NR3>`

`HORizontal:MAIn:SECdiv?`

## HORizontal:MODe

Selects whether the horizontal display uses the main or delayed time base or both. This command is equivalent to setting **Time Base** in the Horizontal menu.

**Group** Horizontal

**Related Commands** DISplay:INTENSITy:CONTRast

**Syntax**

```
HORizontal:MODe { DELAYEd | INTENSIFied | MAIn }
```

```
HORizontal:MODe?
```

**Arguments**

`DELAYEd` means that the selected waveform is horizontally scaled relative to the delayed time base.

`INTENSIFied` uses both the main and delay scales to display the waveform. The portion of the waveform that would be displayed in DELAYEd mode is intensified. Set the level intensity with the DISplay:INTENSITy:CONTRast command.

`MAIn` means that the waveform is horizontally scaled relative to the main time base.

**Examples**

```
HORIZONTAL:MODE DELAYED
```
uses the delayed horizontal scale to display the waveform.

```
HORIZONTAL:MODE?
```
might return `INTENSIFIED`, indicating that the waveform is displayed using both the main and delayed time base scale.

## HORizontal:POSition

Positions the waveform horizontally on the display. This is used for both main and delayed time bases. This command is equivalent to adjusting the front-panel **HORIZONTAL POSITION** knob.

**Group** Horizontal

**Syntax** `HORizontal:POSition <NR3>`

`HORizontal:POSition?`

**Arguments** `<NR3>` is from 0 to 99.9 and is the percent of the waveform that is displayed left of the center graticule.

**Examples** `HORIZONTAL:POSITION 10`
sets the horizontal position of the waveform such that 10% of the waveform is to the left of screen center.

## HORizontal:RECOrdlength

Sets the number of data points that are acquired for each record. You can only set the record length to 1000. This command is here for compatibility with other TDS-series scopes.

**Group** Horizontal

**Syntax** `HORizontal:RECOrdlength <NR1>`

`HORizontal:RECOrdlength?`

**Arguments** `<NR1>` is 1000.

**Examples** `HORIZONTAL:RECORDLENGTH 1000`
specifies that 1000 data points will be acquired for each record.

`HORIZONTAL:RECORDLENGTH?`
returns 1000 as the number of data points per record.

## HORizontal:REF<x>

Sets or queries the reference waveform position lock.

**Group** Horizontal

**Syntax** `HORizontal:REF<x> { LOCk | INDEpendent }`

`HORizontal:REF<x>?`

**Arguments** `LOCk` locks the horizontal position of the reference waveform to the active waveforms.

`INDEpendent` unlocks the horizontal position of the reference waveform and allows it to be positioned independently.

**Examples** `HORIZONTAL:REF1 LOCK`
locks the horizontal position of REF 1 to the active waveforms.

## HORizontal:SCAle

Sets the time per division for the main time base and is identical to the HORizontal:MAIn:SCAle command. It is included here for compatibility purposes.

**Group** Horizontal

**Syntax** `HORizontal:SCAle <NR3>`

`HORizontal:SCAle?`

## HORizontal:SECdiv

Sets the time per division for the main time base and is identical to the HORizontal:MAIn:SCAle command. It is included here for compatibility purposes.

**Group** Horizontal

**Syntax** `HORizontal:SECdiv <NR3>`

`HORizontal:SECdiv?`

## HORizontal:TRIGger? (Query Only)

Returns the horizontal trigger parameter.

**Group** Horizontal

**Syntax** `HORizontal:TRIGger?`

**Examples** `HORIZONTAL:TRIGGER?`
might return `:HORIZONTAL:TRIGGER:POSITION 50`.

## HORizontal:TRIGger:POSition

Sets or queries the position of the trigger. This is equivalent to setting **Trigger Position** in the Horizontal menu.

**Group** Horizontal

**Syntax** `HORizontal:TRIGger:POSition <NR1>`

`HORizontal:TRIGger:POSition?`

**Arguments** `<NR1>` is from 0 to 100% (maximum of 75% in the TDS 380 ), and is the amount of pretrigger information in the waveform.

**Examples** `HORIZONTAL:TRIGGER:POSITION?`
might return 50.

## ID? (Query Only)

Returns identifying information about the oscilloscope and its firmware.

**Group** Status and Error

**Related Commands** *IDN?

**Syntax** ID?

**Returns** TEK/<model number>,CF:91.1CT,FV:<firmware version number>

**Examples** ID?
might return TEK/TDS360,CF:91.1CT,FV:v1.00.

## *IDN? (Query Only)

Returns the oscilloscope identification code.

**Group** Miscellaneous

**Related Commands** ID

**Syntax** *IDN?

**Returns** TEKTRONIX,<model number>,0,CF:91.1CT FV:<firmware version number>

**Examples** *IDN?
might return the response
TEKTRONIX,TDS360,0,CF:91.1CT FV:v1.00

# LOCk

Enables and disables all front-panel buttons and knobs. There is no front-panel equivalent.

**Group** Miscellaneous

**Related Commands** UNLock, Remote Enable Group, Local Lockout Group

**Syntax** `LOCk { ALL | NONe }`

**Arguments** `ALL` disables all front-panel controls.

`NONe` enables all front-panel controls. This is equivalent to the UNLock ALL command.

***NOTE**. If the oscilloscope is in the Remote With Lockout State (RWLS), the LOCk NONe command has no effect. For more information, see the ANSI-IEEE Std. 488.1-1987 Standard Digital Interface for Programmable Instrumentation, section 2.8.3 on RL State Descriptions.*

**Examples** `LOCK?`
returns `NONe` when the front-panel controls are enabled by this command.

`LOCk ALL`
locks the front-panel controls.

## *LRN? (Query Only)

Returns a string listing the oscilloscope settings, except for configuration information for the calibration values. You can use this string to return the oscilloscope to the state it was in when you sent *LRN?.

**Group** Miscellaneous

**Related Commands** HEADer, SET?, VERBose

**Syntax** `*LRN?`

***NOTE***. **LRN? always returns a string including command headers, regardless of the setting of the HEADer command. This is because the returned string is intended to be sent back to the oscilloscope as a command string. The VERBose command can still be used normally to specify whether the returned headers should be abbreviated.*

**Examples** `*LRN?`
a partial response might look like this:

```
:ACQUIRE:STATE 1;MODE SAMPLE;NUMENV 10;NUMAVG 16;
STOPAFTER RUNSTOP;COUNT 1;:HEADER 1;:VERBOSE 1;
:CURSOR:FUNCTION OFF;VBARS:UNITS SECONDS;POSITION1 1.00E-6;
POSITION2 9.00E-6;SELECT CURSOR1
```

## MATH1? (Query Only)

Returns the definition for the math waveform.

**Group** Vertical

**Syntax** `MATH1?`

# MATH1:DEFINE

Allows the user to define a new waveform using a mathematical expression.

**Group** Vertical

**Syntax** `MATH1:DEFINE { <QString> }`

**Arguments** `<QString>` contains the mathematical expression. The expression can include any amount of white space.

The format for a dual waveform expression is:

`<source><operator><source>`

where:

`<operator> ::= { + | - | * }`

`<source> ::= CH<x>`

The format for a single waveform expression is:

`FFT (CH<x>)` displays the FFT frequency domain information for the specified channel.

**Examples** `MATH1:DEFINE "CH1 + CH2"`
sets the math waveform so that it displays the sum of channel 1 and channel 2.

`MATH:DEFINE "FFT(CH1)"`
displays the FFT frequency domain information for channel 1.

## MEASUrement? (Query Only)

Returns all measurement parameters.

**Group** Measurement

**Syntax** MEASUrement?

**Examples** MEASUREMENT?
might return :MEASUREMENT:MEAS1:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;:MEASUREMENT:MEAS2:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;:MEASUREMENT:MEAS3:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;:MEASUREMENT:MEAS4:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s";SOURCE1 CH1;:MEASUREMENT:METHOD HISTOGRAM;REFLEVEL:METHOD PERCENT;ABSOLUTE:HIGH 0.0E0;LOW 0.0E0;MID 0.0E0;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 9.0E1;LOW 1.0E1;MID 5.0E1

## MEASUrement:GATing

Sets or queries measurement gating. Use the vertical bar cursors to define the measurement start and stop locations.

**Group** Measurement

**Syntax** MEASUrement:GATing { OFF | ON | <NR1> }

MEASUrement:GATing?

**Arguments** ON (or 1) turns on measurement gating.

OFF (or 0) turns off measurement gating.

**Examples** MEASUREMENT:GATING ON

MEASUREMENT:GATING?
might return MEASUREMENT:GATING 1 if measurement gating is turned on.

## MEASUrement:IMMed? (Query Only)

Returns all immediate measurement setup parameters.

**Group** Measurement

**Syntax** `MEASUrement:IMMed?`

**Examples** `MEASUREMENT:IMMED?`
might return `:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s"; SOURCE1 CH1`

## MEASUrement:IMMed:SOURCE[1]

Sets or queries the source for all immediate measurements.

**Group** Measurement

**Syntax** `MEASUrement:IMMed:SOURCE[1] { CH<x> | MATH | MATH1 | REF<x> }`

`MEASUrement:IMMed:SOURCE[1]?`

**Arguments** `CH<x>` is an input channel.

`MATH1` is the math waveform.

`REF<x>` is a reference waveform.

**Examples** `MEASUREMENT:IMMED:SOURCE MATH1`
specifies MATH1 as the immediate measurement source.

## MEASUrement:IMMed:TYPe

Specifies the immediate measurement.

**Group** Measurement

**Syntax**

```
MEASUrement:IMMed:TYPe { AMPLitude | BURst | CMEan | CRMs | FALL
| FREQuency | HIGH | LOW | MAXimum | MEAN | MINImum | NDUTy |
NOVershoot | NWIdth | PDUTy | PERIod | PK2pk | POVershoot |
PWIdth | RISe | RMS }

MEASUrement:IMMed:TYPe?
```

**Arguments**

`AMPLitude` is the high value minus the low value.

`BURst` is the time from the first MidRef crossing to the last MidRef crossing.

`CMEan` is the arithmetic mean over one cycle.

`CRMs` is the true Root Mean Square value over one cycle.

`FALL` is the time that it takes for the falling edge of a pulse to fall from a HighRef value to a LowRef value.

`FREQuency` is the reciprocal of the period measured in Hertz.

`HIGH` is the 100% reference level.

`LOW` is the 0% reference level.

`MAXimum` is the highest amplitude.

`MEAN` is the arithmetic mean over the entire waveform.

`MINImum` is the lowest amplitude.

`NDUTy` is the ratio of the negative pulse width to the signal period expressed as a percentage.

`NOVershoot` is the negative overshoot, expressed as:

$$NOVershoot = 100 \times \frac{(Low - Minimum)}{Amplitude}$$

`NWIdth` is the distance (time) between MidRef (usually 50%) amplitude points of a negative pulse.

`PDUTy` is the ratio of the positive pulse width to the signal period expressed as a percentage.

PERIod is the time, in seconds, it takes for one complete signal cycle to happen.

PK2pk is the absolute difference between the maximum and minimum amplitude.

POVershoot is the positive overshoot, expressed as:

$$POVershoot = 100 \times \frac{(Maximum - High)}{Amplitude}$$

PWIdth is the distance (time) between MidRef (usually 50%) amplitude points of a positive pulse.

RISe is the time that it takes for the leading edge of a pulse to rise from a low reference value to a high reference value.

RMS is the true Root Mean Square value.

**Examples**

```
MEASUREMENT:IMMED:TYPE FREQUENCY
```

defines the immediate measurement to be a frequency measurement.

## MEASUrement:IMMed:UNIts? (Query Only)

Returns the units for the immediate measurement.

**Group** Measurement

**Related Commands** MEASUrement:IMMed:TYPe

**Syntax**

```
MEASUrement:IMMed:UNIts?
```

**Returns** <QString> returns "V" for volts, "VV" for volts squared, "dB" for decibels, "s" for seconds, "Hz" for Hertz, or "%" for percent.

**Examples**

```
MEASUREMENT:IMMED:UNITS?
```

might return "s", indicating that the units for the immediate measurement are seconds.

## MEASUrement:IMMed:VALue? (Query Only)

Executes the immediate measurement specified by the MEASUrement:IMMed:TYPe command. The measurement is taken on the source specified by the MEASUrement:IMMed:SOUrce command.

**Group** Measurement

**Syntax** `MEASUrement:IMMed:VALue?`

**Returns** `<NR3>`

## MEASUrement:MEAS<x>? (Query Only)

Returns all measurement parameters for the displayed measurement specified by <x>, where <x> is 1 through 4.

**Group** Measurement

**Syntax** `MEASUrement:MEAS<x>?`

**Examples** `MEASUREMENT:MEAS3?`
might return `:MEASUREMENT:MEAS3:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1`.

## MEASUrement:MEAS<x>:SOURCE[1]

Sets or queries the source for all single channel measurements.

**Group** Measurement

**Syntax** MEASUrement:MEAS<x>:SOURCE[1] { CH<x> | MATH | MATH1 | REF<x> }

MEASUrement:MEAS<x>:SOURCE[1]?

**Arguments** CH<x> is an input channel.

MATH1 is the math waveform.

REF<x> is a reference waveform.

**Examples** MEASUREMENT:MEAS2:SOURCE1 MATH1
specifies MATH1 as the measurement 2 source.

## MEASUrement:MEAS<x>:STATE

Controls the measurement system. The source specified by MEASUrement:MEAS<x>:SOUrce1 must be selected for the measurement to be displayed. The source is selected using the SELect:CONTROl command.

**Group** Measurement

**Syntax** MEASUrement:MEAS<x>:STATE { OFF | ON | <NR1> }

MEASUrement:MEAS<x>:STATE?

**Arguments** OFF or <NR1> = 0 turns measurements off. You can also turn the state off by deselecting the source.

ON or <NR1> $\neq$ 0 turns measurements on.

**Examples** MEASUREMENT:MEAS1:STATE ON
turns measurement defined as MEAS1 on.

MEASUREMENT:MEAS4:STATE?
returns either 0 or 1, indicating the state of MEAS4.

## MEASUrement:MEAS<x>:TYPe

Sets or queries the measurement type for the measurement specified by <x>. This is equivalent to selecting the measurement in the Select Measurement side menu.

**Group** Measurement

**Syntax**

```
MEASUrement:MEAS<x>:TYPe { AMPLitude | BURst | CMEan | CRMs |
FALL | FREQuency | HIGH | LOW | MAXimum | MEAN | MINImum | NDUTy
| NOVershoot | NWIdth | PDUTy | PERIod | PK2pk | POVershoot |
PWIdth | RISe | RMS }
```

```
MEASUrement:MEAS<x>:TYPe?
```

**Arguments**

AMPLitude is the high value minus the low value or HIGH – LOW.

BURst is the time from the first MidRef crossing to the last MidRef crossing.

CMEan is the arithmetic mean over one cycle.

CRMs is the true Root Mean Square value over one cycle.

FALL is the time that it takes for the falling edge of a pulse to fall from a HighRef value to a LowRef value.

FREQuency is the reciprocal of the period measured in Hertz.

HIGH is the 100% reference level.

LOW is the 0% reference level.

MAXimum is the highest amplitude.

MEAN is the arithmetic mean over the entire waveform.

MINImum is the lowest amplitude.

NDUTy is the ratio of the negative pulse width to the signal period expressed as a percentage.

NOVershoot is the negative overshoot, expressed as:

$$NOVershoot = 100 \times \frac{(Low - Minimum)}{Amplitude}$$

NWIdth is the distance (time) between MidRef (usually 50%) amplitude points of a negative pulse.

PDUTy is the ratio of the positive pulse width to the signal period expressed as a percentage.

PERIod is the time, in seconds, it takes for one complete signal cycle to happen.

PK2pk is the absolute difference between the maximum and minimum amplitude.

POVershoot is the positive overshoot, expressed as:

$$POVershoot = 100 \times \frac{(Maximum - High)}{Amplitude}$$

PWIdth is the distance (time) between MidRef (usually 50%) amplitude points of a positive pulse.

RISe is the time that it takes for the leading edge of a pulse to rise from a low reference value to a high reference value.

RMS is the true Root Mean Square value.

**Examples** MEASUREMENT:MEAS3:TYPE RMS
specifies MEAS3 to calculate the Root Mean Square value.

## MEASUrement:MEAS<x>:UNIts? (Query Only)

Returns the units for the measurement specified by MEASUrement:MEAS<x>:TYPe.

**Group** Measurement

**Syntax** MEASUrement:MEAS<x>:UNIts?

**Returns** <QString> returns "V" for volts, "VV" for volts squared, "dB" for decibels, "s" for seconds, "Hz" for Hertz, or "%" for percent.

**Examples** MEASUREMENT:MEAS3:UNITS?
might return "%", indicating the units for Measurement 3 are percent.

## MEASUrement:MEAS<x>:VALue? (Query Only)

Returns the value that has been calculated for the measurement specified by <x>.

***NOTE.*** *This value is a display value and will be updated every 1/3 second.*

**Group** Measurement

**Syntax** `MEASUrement:MEAS<x>:VALue?`

**Returns** `<NR3>`

## MEASUrement:METHod

Sets or queries the method used to calculate the 0% and 100% reference level. This is equivalent to setting the **High-Low Setup** in the Measure menu.

**Group** Measurement

**Syntax** `MEASUrement:METHod { HIStogram | MINMax }`

`MEASUrement:METHod?`

**Arguments** `HIStogram` sets the high and low waveform levels statistically using a histogram algorithm.

`MINMax` sets the high and low waveform levels to MAX and MIN, respectively.

**Examples** `MEASUREMENT:METHOD HISTOGRAM`
specifies that the high and low reference levels are set statistically.

`MEASUREMENT:METHOD?`
returns `MINMAX` when the reference levels are set to MIN and MAX.

## MEASUrement:REFLevel? (Query Only)

Returns the measurement reference levels in percent and absolute terms.

**Group** Measurement

**Syntax** `MEASUrement:REFLevel?`

## MEASUrement:REFLevel:ABSolute:HIGH

Sets or queries the high reference level, and is the 100% reference level when MEASUrement:REFLevel:METHod is set to ABSolute. This command is equivalent to setting the **Reference Levels** in the Measure menu.

**Group** Measurement

**Syntax** `MEASUrement:REFLevel:ABSolute:HIGH <NR3>`

`MEASUrement:REFLevel:ABSolute:HIGH?`

**Arguments** `<NR3>` is the high reference level, in vertical units. The default is 0.0.

**Examples** `MEASUREMENT:REFLEVEL:ABSOLUTE:HIGH 1.71`
sets the high reference level to 1.71 V.

## MEASUrement:REFLevel:ABSolute:LOW

Sets or queries the low reference level, and is the 0% reference level when MEASUrement:REFLevel:METHod is set to ABSolute. This command is equivalent to setting the **Reference Levels** in the Measure menu.

**Group** Measurement

**Syntax** `MEASUrement:REFLevel:ABSolute:LOW <NR3>`

`MEASUrement:REFLevel:ABSolute:LOW?`

**Arguments** `<NR3>` is the low reference level, in vertical units. The default is 0.0.

**Examples** `MEASUrement:REFLevel?`
might return the string `:MEASUREMENT:REFLEVEL:METHOD PERCENT;ABSOLUTE:HIGH 0.0E0;LOW 0.0E0;MID 0.0E0;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 9.0E1;LOW 1.0E1;MID 5.0E1`

## MEASUrement:REFLevel:ABSolute:MID

Sets or queries the mid reference level, and is the 50% reference level when MEASUrement:REFLevel:METHod is set to ABSolute. This command is equivalent to setting the **Reference Levels** in the Measure menu.

**Group** Measurement

**Syntax** `MEASUrement:REFLevel:ABSolute:MID <NR3>`

`MEASUrement:REFLevel:ABSolute:MID?`

**Arguments** `<NR3>` is the mid reference level, in vertical units. The default is 0.0.

**Examples** `MEASUREMENT:REFLEVEL:ABSOLUTE:MID 0.71`
sets the mid reference level to 0.71 volts.

## MEASUrement:REFLevel:METHod

Specifies which reference levels are used for measurement calculations. This command is equivalent to setting the levels in the Reference Levels side menu.

**Group** Measurement

**Syntax**

```
MEASUrement:REFLevel:METHod { ABSolute | PERCent }
```

```
MEASUrement:REFLevel:METHod?
```

**Arguments**

`ABSolute` specifies that the reference levels are set explicitly using the MEASUrement:REFLevel:ABSolute commands. This method is useful when precise values are required.

`PERCent` specifies that the reference levels are calculated as a percent relative to HIGH and LOW. The percentages are defined using the MEASUrement:REFLevel:PERCent commands.

**Examples**

```
MEASUREMENT:REFLEVEL:METHOD ABSolute
```
specifies that explicit user-defined values are used for the reference levels.

```
MEASUREMENT:REFLEVEL:METHOD?
```
returns either `ABSolute` or `PERCENT`, indicating the reference levels used.

## MEASUrement:REFLevel:PERCent:HIGH

Sets or queries the percent, relative to HIGH, that is used to calculate the high reference level when MEASUrement:REFLevel:METHod is set to PERCent. This command is equivalent to setting the **Reference Levels** in the Measure menu.

**Group** Measurement

**Syntax** `MEASUrement:REFLevel:PERCent:HIGH <NR3>`

`MEASUrement:REFLevel:PERCent:HIGH?`

**Arguments** `<NR3>` ranges from 0 to 100%, and is the high reference level. The default is 90%.

**Examples** `MEASUREMENT:REFLEVEL:PERCENT:HIGH 95`
specifies that the high reference level is set to 95% of HIGH.

## MEASUrement:REFLevel:PERCent:LOW

Sets or queries the percent, relative to LOW, that is used to calculate the low reference level when MEASUrement:REFLevel:METHod is set to PERCent. This command is equivalent to setting the **Reference Levels** in the Measure menu.

**Group** Measurement

**Syntax** `MEASUrement:REFLevel:PERCent:LOW <NR3>`

`MEASUrement:REFLevel:PERCent:LOW?`

**Arguments** `<NR3>` ranges from 0 to 100%, and is the low reference level. The default is 10%.

**Examples** `MEASUREMENT:REFLEVEL:PERCENT:LOW?`
might return 15, meaning that the low reference level is 15% of LOW.

## MEASUrement:REFLevel:PERCent:MID

Sets or queries the percent, relative to MID, that is used to calculate the mid reference level when MEASUrement:REFLevel:METHod is set to PERCent. This command is equivalent to setting the **Reference Levels** in the Measure menu.

**Group** Measurement

**Syntax**

```
MEASUrement:REFLevel:PERCent:MID <NR3>
MEASUrement:REFLevel:PERCent:MID?
```

**Arguments** <NR3> ranges from 0 to 100%, and is the mid reference level. The default is 50%.

**Examples**

```
MEASUREMENT:REFLEVEL:PERCENT:MID 60
```

specifies that the mid reference level is set to 60% of MID.

## NEWpass (No Query Form)

Changes the password that enables access to password protected data. The PASSWord command must be successfully executed before using this command or an execution error will be generated.

**Group** Miscellaneous

**Related Commands** PASSWord, *PUD

**Syntax**

```
NEWpass <QString>
```

**Arguments** <QString> is the new password. The password can include up to 10 characters.

**Examples**

```
NEWPASS "mypassword"
```

creates a new password for accessing the user protected data.

# *OPC

Generates the operation complete message in the Standard Event Status Register (SESR) when all pending operations finish. The *OPC? query places the ASCII character "1" into the output queue when all pending operations are finished. The *OPC? response is not available to read until all pending operations finish. For a complete discussion of the use of these registers and the output queue, see page 3–1.

Table 2–25 lists commands that generate an operation complete message.

**Group** Status and Error

**Related Commands** BUSY?, *WAI

**Syntax**

```
*OPC
```

```
*OPC?
```

The *OPC command allows you to synchronize the operation of the oscilloscope with your application program. Synchronization methods are described starting on page 3–7.

**Table 2–25: Commands that generate an Operation Complete message**

| | |
|---|---|
| Automatic scope adjustment | AUTOSet EXECute |
| Internal self-calibration | *CAL |
| Single sequence acquisition | ACQuire:STATE ON or<br>ACQuire:STATE RUN<br>(when ACQuire:STOPAfter is set to SEQuence) |
| Hardcopy output | HARDCopy STARt |

**Examples**

```
*OPC?
```

might return a 1, which indicates that the operation is complete.

## PASSWord (No Query Form)

Enables the *PUD and NEWpass set commands. Sending PASSWord without any arguments disables these same commands. Once the password is successfully entered, the *PUD and NEWpass commands are enabled until the oscilloscope is powered off, or until the FACtory command, the PASSWord command with no arguments, or the *RST command is issued.

To change the password, you must first enter the valid password with the PASSWord command and then change to your new password with the NEWpass command. Remember that the password is case sensitive.

**Group** Miscellaneous

**Related Commands** NEWpass, *PUD

**Syntax** `PASSWord[<QString>]`

**Arguments** `<QString>` is the password and can include up to 10 characters. The factory default password is “XYZZY”and is always valid.

**Examples** `PASSWORD "XYZZY"`
Enables the *PUD and NEWpass set commands.

`PASSWORD`
Disables the *PUD and NEWpass set commands. You can still use the query version of *PUD.

# *PSC

Sets or queries the power-on status flag that controls the automatic power-on handling of the DESER, SRER, and ESER registers. When *PSC is true, the DESER register is set to 255 and the SRER and ESER registers are set to 0 at power on. When *PSC is false, the current values in the DESER, SRER, and ESER registers are preserved in nonvolatile memory when power is shut off and are restored at power on. For a complete discussion of the use of these registers, see page 3–1.

**Group** Status and Error

**Related Commands** DESE, *ESE, FACtory, *RST, *SRE

**Syntax**

```
*PSC <NR1>
```

```
*PSC?
```

**Arguments**

<NR1> = 0 sets the power-on status clear flag to false, disables the power on clear, and allows the oscilloscope to possibly assert SRQ after power on.

<NR1> $\neq$ 0 sets the power-on status clear flag true. Sending *PSC 1, therefore, enables the power-on status clear and prevents any SRQ assertion after power-on. Using an out-of-range value causes an execution warning.

**Examples**

```
*PSC 0
```
sets the power-on status clear flag to false.

```
*PSC?
```
might return the value 1, showing that the power-on status clear flag is set to true.

## *PUD

Sets or queries a string of Protected User Data. This data is protected by the PASSWord command. You can modify it only by first entering the correct password. The password is not necessary to query the data.

**Group** Miscellaneous

**Related Commands** PASSWord

**Syntax** `*PUD <Block>`

`*PUD?`

**Arguments** `<Block>` is a string containing up to 100 characters.

**Examples** `*PUD #229This instrument belongs to me`
stores the string “This instrument belongs to me” in the user protected data area.

`*PUD?`
might return `#221Property of Company X`.

## *RCL (No Query Form)

Restores the state of the oscilloscope from a copy of its settings stored in memory. (The settings are stored using the *SAV command.) This command is equivalent to RECAll:SETUp, and performs the same function as the **Recall Saved Setup** item in the front-panel Save/Recall menu.

**Group** Save and Recall

**Related Commands** FACtory, *LRN?, RECAll:SETUp, *RST, *SAV, SAVe:SETUp

**Syntax** `*RCL <NR1>`

**Arguments** `<NR1>` is a value in the range from 1 to 10, and specifies a setup storage location. Using an out-of-range value causes an execution error (222, “Data out of range”).

**Examples** `*RCL 3`
restores the oscilloscope from a copy of the settings stored in memory location 3.

## RECAll:SETUp (No Query Form)

Restores a stored or factory front-panel setup of the oscilloscope. This command is equivalent to selecting **Recall Saved Setup** or **Recall Factory Setup** in the Save/Recall menu.

***NOTE.*** *TDS 300 Series oscilloscope setup files are not compatible with TDS 400, TDS 500, TDS 600, TDS 700, or TDS 800 Series oscilloscopes setup files. Refer to* Setup and Waveform File Formats *on page D–2 for more information*

**Group** Save and Recall

**Related Commands** FACtory, *RCL, *RST, *SAV, SAVe:SETUp, TEKSecure

**Syntax** `RECAll:SETUp { FACtory | <NR1> | <QString> }`

**Arguments** `FACtory` selects the factory setup. This is the same as the FACtory command.

`<NR1>` is a value in the range from 1 to 10 and specifies a setup storage location. Using an out-of-range value causes an execution error (222, “Data out of range”).

`<QString>` is the name of a file that contains setup information.

**Examples**

```
RECALL:SETUP FACTORY
```

recalls the front-panel setup to its factory defaults.

## RECAll:WAVEform (No Query Form)

Recalls a stored waveform into a reference location.

***NOTE.*** *TDS 300 Series oscilloscope waveform files are not compatible with TDS 400, TDS 500, TDS 600, TDS 700, or TDS 800 Series oscilloscopes waveform files. Refer to* Setup and Waveform File Formats *on page D–2 for more information*

**Group** Save and Recall

**Syntax** `RECAll:WAVEform <QString>,REF<x>`

**Arguments** `<QString>` is a quoted string that defines the file name and path. Input the file path using the form `<drive>/<dir>/<filename>`. `<drive>` and one or more `<dir>`s are optional. If you do not specify them, the oscilloscope will recall the waveform from the default directory. `<filename>` stands for a filename of up to 8 characters followed by a period (".") and any 3-character extension. Do not use wild card characters.

`REF<x>` is the location in internal reference memory where the waveform is recalled from.

**Examples**
```
RECALL:WAVEFORM "TEK00000.WFM",REF1
```
recalls the waveform stored in the file named TEK00000.WFM to reference location 1.

## REM (No Query Form)

Specifies a comment. This line is ignored by the oscilloscope.

**Group** Miscellaneous

**Syntax** `REM <QString>`

**Arguments** `<QString>` is a string that can have a maximum of 80 characters.

**Examples**
```
REM "This is a comment"
```
is ignored by the oscilloscope.

## *RST (No Query Form)

(Reset) Returns the oscilloscope to a known set of oscilloscope settings, but does not purge any aliases or stored settings. This command is the same as the FACtory command. Refer to Appendix D for a listing of the factory default settings.

**Group** Miscellaneous

**Related Commands** FACtory, *PSC, *RCL, RECAll:SETUp, *SAV, SAVe:SETUp, TEKSecure

**Syntax** `*RST`

Setting the oscilloscope to factory default has the following impact on the programming interface:

- Clears the Event Status Enable Register
- Clears the Service Request Enable Register
- Sets the Device Event Status Enable Register to 255
- Sets the Power On Status Clear Flag to TRUE
- Purges all defined aliases
- Enables all Command Headers (HEADer ON)
- Sets the macro defined by *DDT to a “zero-length field”
- Clears the pending operation flag and associated operations

The FACtory command does not alter the following items:

- The state of the RS-232 or GPIB interface
- Calibration data that affects device specifications
- Protected user data
- Stored settings
- Stored waveforms or data
- The current password (if implemented)
- Hard copy parameters

## RS232? (Query Only)

Queries the RS232 settings.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232: PARITY, RS232:SOFT-FLAGGING, RS232: STOPBITS

**Syntax** `RS232?`

**Arguments** None

**Examples** `RS232?`
might return: `RS232:BAUD 9600;MODE RAW;PACE XON;PARITY NONE;SBITS 2;CONTROL:RTS RFR;RS232:TRANSMIT;TERMINATOR LF;DELAY 0.0`

## RS232:BAUd

Sets or queries the RS-232 interface transmission speed. If no flow control (flagging) is used, commands may be received faster than the oscilloscope can process them. Also, if another command is sent immediately after this command, without waiting for the baud rate to be programmed, the first couple of characters may be lost.

**Group** Miscellaneous

**Related Commands** RS232: HARDFLAGGING, RS232: PARITY, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

**Syntax** `RS232:BAUd <NR1>`

`RS232:BAUd?`

**Arguments** `<NR1>` where `<NR1>` can be 300, 600, 1200, 2400, 4800, 9600, 19200, or 38400.

**Examples** `RS232:BAUD 9600`
sets the transmission rate to 9600 baud.

## RS232:CONTROl:DCD

The oscilloscope accepts but ignores this command.

**Group** Miscellaneous

**Related Commands** RS232: HARDFLAGGING, RS232: PARITY, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

**Syntax** `RS232:CONTROl:DCD { OFF | 0 }`

`RS232:DCD?`

**Arguments** `OFF or 0` turn off DCD monitoring.

**Examples** `RS232:CONTROl:DCD OFF`
turns off DCD monitoring. This is essentially a no-op since DCD monitoring is always off.

## RS232:CONTROl:RTS

Sets or queries the state of RS232 hard flagging. This command performs the same function as RS232:HARDFlagging.

**Group** Miscellaneous

**Related Commands** RS232: HARDFLAGGING, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

**Syntax**

```
RS232:CONTrol:RTS { ON | RFR | IBFull }
```

```
RS232:CONTrol:RTS?
```

**Arguments** <ON> asserts RTS (Request to Send).

<RFR> enables hard flagging.

<IBFull> enables hard flagging.

**Examples**

```
RS232:CONTrol:RTS RFR
```

enables hard flagging.

## RS232:HARDFlagging

Sets or queries the state of RS232 hard flagging. When hard flagging is enabled, the oscilloscope sends data as long as CTS (Clear To Send) is asserted. When receiving data, the oscilloscope asserts RTS (Request To Send) until the input buffer is almost full. When the oscilloscope no longer asserts RTS, it continues to read incoming data until the input buffer is full and then reports an input overrun error. The oscilloscope asserts DTR (Data Terminal Ready) when oscilloscope power is on.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: PARITY, RS232:SOFTFLAGGING, RS232: STOP-BITS, RS232?

**Syntax**

```
RS232:HARDFlagging { ON | OFF | <NR1> }
```

```
RS232:HARDFlagging?
```

**Arguments**

<ON> or <NR1>≠ 0 turn on hardflagging.

<OFF> or <NR1> = 0 turn off hardflagging (RTS always asserted).

**Examples**

```
RS232:HARDFLAGGING ON
```

turns on hard flagging.

## RS232:MODE

The oscilloscope accepts but ignores this command. The query always returns RAW.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232: PARITY, RS232:SOFTFLAGGING, RS232?

**Syntax** `RS232:MODE RAW`

`RS232:MODE?`

**Arguments** `RAW` (GPIB emulation) mode.

**Examples** `RS232:MODE?`
always returns RAW.

## RS232:PACE

Sets or queries the input and output soft flagging over the RS-232 port. This command performs the same function as `RS232:SOFTFlagging`.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232: PARITY, RS232: STOPBITS, RS232?

**Syntax** `RS232:PACE { XON | NONe }`

`RS232:PACE?`

**Arguments** `<XON>` turn on soft flagging.

`<NONE>` turn off soft flagging.

**Examples** `RS232:PACE XON`
turns on soft flagging.

## RS232:PARity

Sets or queries the parity used for all RS-232 data transfers. When parity is odd or even, the oscilloscope generates the selected parity on output and checks all input against the selected parity. When parity is none, the oscilloscope performs no input parity error checks and generates no output parity. When the parity (9th) bit does not match the parity type, the oscilloscope reports a parity error. If another command is sent immediately after this command, without waiting for the parity to be programmed, the first few characters may be lost.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

**Syntax**

```
RS232:PARity { EVEN | ODD | NONe }
RS232:PARity?
```

**Arguments**

<EVEN> sets even parity.

<ODD> sets odd parity.

<NONe> sets no parity (no ninth bit transmitted).

**Examples**

```
RS232:PARITY EVEN
```

sets even parity.

## RS232:PRESet (No Query Form)

Sets RS-232 parameters to default values. The RS232? query will show the new settings.

RS232:MODE RAW
RS232:CONTrol:RTS RFR
RS232:CONTrol:DCD OFF
RS232:PACE NONE
RS232:BAUD 9600
RS232:PARity NONE
RS232:SBITs 1
RS232:TRANsmit:DELay 0

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

**Syntax** `RS232:PRESet`

**Arguments** None.

**Examples** `RS232:PRESET`
sets RS232 parameters to the default values.

## RS232:SBITS

Sets or queries the number of transmission stop bits sent with each character to identify the end of data for that character. This command performs the same function as RS232:STOPBits.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232: PARITY, RS232:SOFT-FLAGGING, RS232?

**Syntax**

```
RS232:SBITS <NR1>
```

```
RS232:SBITS?
```

**Arguments** <NR1> where <NR1> can either be 1 or 2.

**Examples**

```
RS232:SBITS 1
```

sets the number of stop bits to 1.

## RS232:SOFTFlagging

Sets or queries the input and output soft flagging over the RS-232 port. After receiving an XOFF (DC3), the oscilloscope sends two or less characters. The oscilloscope sends an XOFF character when its input buffer is running out of space. After sending an XOFF character it can receive at least 20 more bytes. The oscilloscope begins transmitting data again when it receives an XON (DC1) character. It sends XON when its input buffer has an acceptable number of free bytes.

When soft flagging is enabled and binary data is transferred, data transmission will lock up if the data contains XOFF or XON characters.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232: PARITY, RS232: STOPBITS, RS232?

**Syntax**

```
RS232:SOFTFlagging { ON | OFF | <NR1> }
RS232:SOFTFlagging?
```

**Arguments**

<ON> or <NR1>≠ 0 turns on softflagging.

<OFF> or <NR1> = 0 turns off softflagging.

**Examples**

```
RS232:SOFTFLAGGING ON
```

turns on soft flagging.

## RS232:STOPBits

Sets or queries the number of transmission stop bits sent with each character to identify the end of data for that character. The standard setting for most computer equipment is 1 stop bit. If another command is sent immediately after this command without waiting for it to complete, the first couple of characters may be lost.

**Group** Miscellaneous

**Related Commands** RS232: BAUD, RS232: HARDFLAGGING, RS232: PARITY, RS232:SOFT-FLAGGING, RS232?

**Syntax**

```
RS232:STOPBits <NR1>
```

```
RS232:STOPBits?
```

**Arguments** <NR1> where <NR1> can either be 1 or 2.

**Examples**

```
RS232:STOPBITS 1
```

sets the number of stop bits to 1.

## RS232:TRANsmit:DELay

Sets or queries the minimum amount of time to wait after receiving a query command before sending the response. This is provided for old terminals and computers that cannot accept data immediately after sending data.

**Group** Miscellaneous

**Related Commands** RS232: HARDFLAGGING, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

**Syntax**

```
RS232:TRANsmit:DELay { <NR1> }
RS232:TRANsmit:DELay?
```

**Arguments** <NR1> the delay value from 0 s to 60 s.

**Examples**

```
RS232:TRANsmit:DELay 0
```

sets the transmit delay to 0 s.

## RS232:TRANsmit:TERMinator

Sets or queries the end-of-line (EOL) terminator. When transmitting, the oscilloscope appends the terminator to the end of each message. When receiving, the oscilloscope accepts all four terminators, regardless of the currently selected terminator. When a combination of multiple characters is selected (CRLF or LFCR), the oscilloscope interprets the first character as the terminator; it treats the second character as a null command.

CR represents an ASCII carriage return character (0x0D) and LF represents an ASCII linefeed character (0x0A).

**Group** Miscellaneous

**Related Commands** RS232: HARDFLAGGING, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

**Syntax**

```
RS232:TRANsmit:TERMinator { CR | LF | CRLf | LFCr }
```

```
RS232:TRANsmit:TERMinator?
```

**Arguments**

<CR> selects the carriage return character as the EOL terminator.

<LF> selects the line feed character as the EOL terminator.

<CRLf> selects the carriage return and line feed characters as the EOL terminator.

<LFCr> selects the line feed and carriage return characters as the EOL terminator.

**Examples**

```
RS232:TRANsmit:TERMinator CR
```
sets the carriage return as the EOL terminator.

## *SAV (No Query Form)

(Save) Stores the state of the oscilloscope into a specified memory location. You can later use the *RCL command to restore the oscilloscope to this saved state. This is equivalent to selecting **Save Current Setup** in the Save/Recall menu.

**Group** Save and Recall

**Related Commands** FACtory, *RCL, RECAll:SETUp, SAVe:SETUp

**Syntax** `*SAV <NR1>`

**Arguments** `<NR1>` is a value in the range from 1 to 10 and specifies a location. Using an out-of-range value causes an execution error. Any settings that have been stored previously at this location will be overwritten.

**Examples** `*SAV 2`
saves the current settings in memory location 2.

## SAVe:SETUp (No Query Form)

Saves the current front-panel setup into the specified memory location or file. This is equivalent to selecting **Save Current Setup** in the Save/Recall menu. This command is the same as *SAV.

***NOTE.*** *TDS 300 Series oscilloscope setup files are not compatible with TDS 400, TDS 500, TDS 600, TDS 700, or TDS 800 Series oscilloscope setup files. Refer to* Setup and Waveform File Formats *on page D–2 for more information*

**Group** Save and Recall

**Related Commands** RECAll:SETUp, *RCL, *SAV

**Syntax** `SAVe:SETUp { <NR1> | <QString> }`

**Arguments** `<NR1>` is a value in the range from 1 to 10 and specifies a location. Using an out-of-range value causes an execution error. Any settings that have been stored previously at this location will be overwritten.

`<QString>` is a quoted string that defines the file name and path. Enter the file path using the form `<drive>/<dir>/<filename>`. `<drive>` and one or more `<dir>`s are optional. If you do not specify them, the oscilloscope will save the settings in the current directory. `<filename>` stands for a filename of up to 8 characters followed by a period (".") and any 3-character extension. Do not use wild card characters.

**Examples**

```
SAVE:SETUP 5
```

saves the current front-panel setup in memory location 5.

## SAVe:WAVEform (No Query Form)

Stores a waveform in one of two reference memory locations or a file.

***NOTE.*** *TDS 300 Series oscilloscope waveform files are not compatible with TDS 400, TDS 500, TDS 600, TDS 700, or TDS 800 Series oscilloscope waveform files. Refer to* Setup and Waveform File Formats *on page D–2 for more information*

**Group** Save and Recall

**Syntax** `SAVe:WAVEform <wfm><Comma>REF<x> | <QString>`

**Arguments** `<wfm>` is CH<x>, MATH1, or REF<x>, and is the waveform that will be saved.

`REF<x>` is the location where the waveform will be stored.

`<QString>` is a quoted string that defines the file name and path. Enter the file path using the form `<drive>/<dir>/<filename>`. `<drive>` and one or more `<dir>`s are optional. If you do not specify them, the oscilloscope will save the waveform to the current directory. `<filename>` stands for a filename of up to 8 characters followed by a period (".") and any 3-character extension. Do not use wild card characters.

**Examples** `SAVE:WAVEFORM MATH1,REF1`
saves the math 1 waveform in reference memory location 1.

## SAVe:WAVEform:FILEFormat

Specifies the file format for saved waveforms.

**Group** Save and Recall

**Related Commands** SAVE:WAVEFORM

**Syntax**

```
SAVe:WAVEform:FILEFormat{ INTERNal | SPREADSheet | MATHCad }
```

```
SAVe:WAVEform:FILEFormat?
```

**Arguments**

`INTERNal` specifies the internal format. Internal format files have a .isf extension.

`SPREADSheet` specifies the spreadsheet format. Spreadsheet format files have a .CSV extension.

`MATHCad` specifies the MathCad format. MathCad format files have a .DAT extension.

**Examples**

```
SAVE:WAVEFORM:FILEFORMAT SPREADSHEET
```

specifies the waveform, when saved, will be stored in a spreadsheet-compatible format.

## SELect? (Query Only)

Returns the selected waveform and the display status of all waveforms.

**Group** Vertical

**Syntax** SELect?

**Examples** SELECT?
might return :SELECT:CH1 1;CH2 0;MATH1 0;REF1 0;REF2 0;CONTROL CH1

## SELect: <wfm>

Controls the display and selection of waveforms. There can be up to five waveforms displayed at one time, but only one waveform can be selected at a time. The selected waveform is the waveform that was most recently turned on. This command is equivalent to pressing a front-panel channel button (**CH 1**, **CH 2**, **MATH**, **REF 1**, or **REF 2**). <wfm> can be CH<x>, MATH1, or REF<x>.

**Group** Vertical

**Syntax** SELect:<wfm> { OFF | ON | <NR1> }

SELect:<wfm>?

**Arguments** OFF or <NR1> = 0 turns off the display of the specified waveform.

ON or <NR1> $\neq$ 0 turns on the display of the specified waveform. The waveform also becomes the selected waveform.

**Examples** SELECT:CH2 ON
turns the channel 2 display on and selects channel 2.

SELECT:REF1?
returns either 0 or 1, indicating whether the REF1 waveform is displayed.

## SELect:CONTROl

Sets or queries the waveform that is currently measured and adjusted by the cursor and vertical commands.

**Group** Vertical

**Syntax** `SELect:CONTROl <wfm>`

`SELect:CONTROl?`

**Arguments** <wfm> is CH<x>, MATH1, or REF<x>, and is the selected waveform.

**Examples** `SELECT:CONTROL?`
might return `CH1` as the selected waveform.

## SET? (Query Only)

Returns a string listing the oscilloscope settings, except for the calibration values. You can use this string to return the oscilloscope to the state it was in when you sent SET?. This command is identical to the *LRN? command.

**Group** Miscellaneous

**Related Commands** HEADer, *LRN?, VERBos

**Syntax** `SET?`

***NOTE**. The SET? query always returns a string with command headers, regardless of the setting of the HEADer command. This is because the returned string is intended to be useable as a command string that can sent back to the oscilloscope . The VERBos command can still be used to specify whether the returned headers should be abbreviated or full length.*

**Examples** `SET?`
a partial return string may look like this:

```
:ACQUIRE:STOPAFTER RUNSTOP;STATE 1;MODE SAMPLE;NUMENV 8;NUMAVG
16;:HEADER 1;:VERBOS 1; :ALIAS:STATE 0;:DISPLAY:FORMAT YT;STYLE
VECTORS;PERSISTENCE 500.0E-3;GRATICULE FULL;TRIGT 1;...;
```

# *SRE

(Service Request Enable) sets and queries the bits in the Service Request Enable Register (SRER). For a complete discussion of the use of these registers, see page 3–1.

**Group** Status and Error

**Related Commands** *CLS, DESE, *ESE, *ESR?, EVENT?, EVMSg?, FACtory, *PSC, *STB?

**Syntax** *SRE <NR1>

*SRE?

**Arguments** <NR1> is a value in the range from 0 to 255. The binary bits of the SRER are set according to this value. Using an out-of-range value causes an execution error. The power-on default for SRER is 0 if *PSC is 1. If *PSC is 0, the SRER maintains its value through a power cycle.

**Examples** *SRE 48
sets the bits in the SRER to 00110000 binary.

*SRE?
might return a value of 32, showing that the bits in the SRER have the binary value 00100000.

## *STB? (Query Only)

(Read Status Byte) query returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. For a complete discussion of the use of these registers, see page 3–1.

**Group** Status and Error

**Related Commands** *CLS, DESE, *ESE, *ESR?, EVENT?, EVMSg?, FACtory, *SRE

**Syntax** `*STB?`

**Returns** `<NR1>`

**Examples** `*STB?`
might return the value `96`, showing that the SBR contains the binary value 01100000.

## TEKSecure

Initializes both waveform and setup memories. This overwrites any previously stored data: it nulls all waveform reference memory and puts all setups in the factory init state. It then verifies that the waveform and setup memory are in the desired state and displays a pass or a fail notifier on completion.

**Group** Miscellaneous

**Syntax** `TEKSecure`

## TIMe

Sets or queries the time that the oscilloscope can display.

**Group** Miscellaneous

**Related Commands** DATE, DISplay: CLOCk

**Syntax** `TIMe <QString>`

`TIMe?`

**Arguments** `<QString>` is a date in the form `"hh:mm:ss"`.
`hh` refers to the hour number from 01 to 24.
`mm` refers to the minute number in the hour from 00 to 59.
`ss` refers to the seconds number in the minute from 00 to 59.
There must be a colon after the `hh` and after the `mm`.
Use two digits for each of the `hh`, `mm`, and `ss`.

**Examples** `TIME "01:24:00"`
specifies that the time is set to 01:24 AM.

## *TRG (No Query Form)

(Trigger) Executes commands that are defined by *DDT.

The Group Execute Trigger (GET) interface message has the same effect as the *TRG command.

**Group** Miscellaneous

**Related Commands** Alias commands, *DDT

**Syntax** `*TRG`

**Examples** `*TRG`
immediately executes all commands that have been defined by *DDT.

## TRIGger

Forces a trigger event to occur and the TRIGger query returns the current trigger parameters.

**Group** Trigger

**Syntax** `TRIGger FORCe`

`TRIGger?`

**Arguments** `FORCe` creates a trigger event. If TRIGger:STATE is REAdy, the acquisition will complete, otherwise this command will be ignored. This is equivalent to pressing the front-panel **FORCE TRIGGER** button.

**Examples** `TRIGGER FORCe`
forces a trigger event to occur.

`TRIGGER?`
might return `:TRIGGER:MAIN:MODE AUTO;TYPE EDGE;LEVEL -4.8E-1;HOLDOFF:VALUE 5.0E-7;:TRIGGER:MAIN:EDGE:SOURCE CH1; COUPLING DC;SLOPE RISE;:TRIGGER:MAIN:VIDEO:SOURCE CH1;HOLDOFF:VALUE 5.0E-7;:TRIGGER:MAIN:VIDEO:FIELD FIELD1;SCAN RATE1`

## TRIGger:MAIn

Sets the main trigger level and returns the current main trigger parameters.

**Group** Trigger

**Syntax** `TRIGger:MAIn SETLevel`

`TRIGger:MAIn?`

**Arguments** `SETLevel` sets the main trigger level to half way between the MIN and MAX amplitudes of the trigger source input. This is equivalent to pressing the front-panel **SET LEVEL TO 50%** button.

**Examples** `TRIGGER:MAIN SETLEVEL`
sets the main trigger level mid way between MAX and MIN.

## TRIGger:MAIn:EDGE? (Query Only)

Returns the trigger coupling, source, and slope for the main edge trigger.

**Group** Trigger

**Syntax** `TRIGger:MAIn:EDGE?`

**Examples** `TRIGGER:MAIN:EDGE?`
might return `SOURCE CH1;COUPLING DC;SLOPE RISE`

## TRIGger:MAIn:EDGE:COUPling

Sets or queries the type of coupling for the main edge trigger. This is equivalent to setting **Coupling** in the Trigger menu.

**Group** Trigger

**Syntax**

```
TRIGger:MAIn:EDGE:COUPling { AC | DC | HFRej | LFRej | NOISErej }
TRIGger:MAIn:EDGE:COUPling?
```

**Arguments**

`AC` selects AC trigger coupling.

`DC` selects DC trigger coupling.

`HFRej` coupling removes the high-frequency components of the DC signal.

`LFRej` coupling removes the low-frequency components of the AC signal.

`NOISErej` selects DC low sensitivity. It requires added signal amplitude for more stable, less false triggering.

**Examples**

```
TRIGGER:MAIN:EDGE:COUPLING DC
```
sets the main edge trigger coupling to DC.

## TRIGger:MAIn:EDGE:SLOpe

Selects a rising or falling slope for the main edge trigger. This is equivalent to setting **Slope** in the Trigger menu.

**Group** Trigger

**Syntax**

```
TRIGger:MAIn:EDGE:SLOpe { FALL | RISe }
TRIGger:MAIn:EDGE:SLOpe?
```

**Arguments**

`FALL` specifies to trigger on the falling or negative edge of a signal.

`RISe` specifies to trigger on the rising or positive edge of a signal.

**Examples**

```
TRIGGER:MAIN:EDGE:SLOPE RISE
```
sets the main edge trigger to occur on the rising slope.

## TRIGger:MAIn:EDGE:SOUrce

Sets or queries the source for the main edge trigger. This is equivalent to setting **Source** in the Trigger menu.

**Group** Trigger

**Syntax** `TRIGger:MAIn:EDGE:SOUrce { EXT | EXT10 | CH<x> | LINE }`

`TRIGger:MAIn:EDGE:SOUrce?`

**Arguments** `EXT` specifies an external trigger using the **EXT TRIG** connector.

`EXT10` specifies an external trigger using the **EXT TRIG** connector, with x10 attenuation.

`CH<x>` specifies one of the input channels.

`LINE` specifies AC line voltage.

**Examples** `TRIGGER:MAIN:EDGE:SOURCE LINE`
specifies the AC line voltage as the main edge trigger source.

`TRIGGER:MAIN:EDGE:SOURCE?`
might return `CH2` for the main edge trigger source.

## TRIGger:MAIn:HOLdoff? (Query Only)

Returns the main trigger holdoff value.

**Group** Trigger

**Syntax** `TRIGger:MAIn:HOLdoff?`

**Examples** `TRIGGER:MAIN:HOLDOFF?`
might return `:TRIGGER:MAIN:HOLDOFF:VALUE 5.0E-7`.

## TRIGger:MAIn:HOLdoff:VALue

Sets or queries the main trigger holdoff value.

**Group** Trigger

**Syntax** `TRIGger:MAIn:HOLdoff:VALue <NR3>`

`TRIGger:MAIn:HOLdoff:VALue?`

**Arguments** `<NR3>` is the holdoff, from 500 ns to 10 s.

**Examples** `TRIGGER:MAIN:HOLDOFF:VALUE 10`
sets the holdoff value to 10 s.

## TRIGger:MAIn:LEVel

Sets the main trigger level. This command is equivalent to adjusting the front-panel **TRIGGER LEVEL** knob.

**Group** Trigger

**Syntax** `TRIGger:MAIn:LEVel { ECL | TTL | <NR3> }`

`TRIGger:MAIn:LEVel?`

**Arguments** `ECL` specifies a preset ECL level of –1.3 V.

`TTL` specifies a preset TTL level of 1.4 V.

`<NR3>` specifies the main trigger level, in volts.

**Examples** `TRIGGER:MAIN:LEVEL?`
might return `1.4`, indicating that the main edge trigger is set to 1.4 V.

## TRIGger:MAIn:MODe

Sets or queries the main trigger mode.

**Group** Trigger

**Syntax** `TRIGger:MAIn:MODe { AUTO | NORMal }`

`TRIGger:MAIn:MODe?`

**Arguments** `AUTO` generates a trigger if a trigger is not detected within a specific time period. Auto also enables roll-mode when sweep speeds are slower than 100ms/div.

`NORMal` waits for a valid trigger event.

**Examples** `TRIGGER:MAIN:MODE AUTO`
specifies that a trigger event is automatically generated.

## TRIGger:MAIn:TYPe

Sets or queries the type of main trigger. This is equivalent to setting **Type** in the Trigger menu.

**Group** Trigger

**Syntax** `TRIGger:MAIn:TYPe { EDGE | VIDeo }`

`TRIGger:MAIn:TYPe?`

**Arguments** `EDGE` is a normal trigger. A trigger event occurs when a signal passes through a specified voltage level in a specified direction. Edge trigger is controlled by the TRIGger:MAIn:EDGE commands.

`VIDeo` specifies that a trigger occurs when a specified signal is found. Video trigger is controlled by the TRIGger:MAIn:VIDeo commands.

**Examples** `TRIGGER:MAIN:TYPE?`
might return `VIDEO` indicating that the main trigger type is a video trigger.

## TRIGger:MAIn:VIDeo:FIELD

Sets or queries the field the video trigger detects.

**Group** Trigger

**Syntax** `TRIGger:MAIn:VIDeo:FIELD { FIELD1 | FIELD2 | ANY }`

`TRIGger:MAIn:VIDeo:FIELD?`

**Arguments** `FIELD1` specifies interlaced video field 1.

`FIELD2` specifies interlaced video field 2.

`ANY` specifies any line.

**Examples** `TRIGGER:MAIN:VIDEO:FIELD1`
selects field 1.

## TRIGger:MAIn:VIDeo:HOLdoff? (Query Only)

Returns the video trigger holdoff value.

**Group** Trigger

**Syntax** `TRIGger:MAIn:VIDeo:HOLdoff?`

**Examples** `TRIGGER:MAIN:VIDEO:HOLDOFF?`
might return `:TRIGGER:MAIN:VIDEO:HOLDOFF:VALUE 5.0E-7`.

## TRIGger:MAIn:VIDeo:HOLdoff:VALue

Sets or queries the video trigger holdoff value. This is equivalent to setting **Holdoff** in the video trigger menu's **Holdoff** side menu.

**Group** Trigger

**Syntax** `TRIGger:MAIn:VIDeo:HOLdoff:VALue <NR3>`

`TRIGger:MAIn:VIDeo:HOLdoff:VALue?`

**Arguments** `<NR3>` is the holdoff, from 500 ns to 10 s.

**Examples** `TRIGGER:MAIN:HOLDOFF:VALUE 3E-03`
set the holdoff value to 3 ms.

## TRIGger:MAIn:VIDeo:SCAN

Sets or queries the video trigger scan rate. This is equivalent to setting **Scan Rate** in the video trigger menu's **Scan Rate** side menu.

**Group** Trigger

**Syntax** `TRIGger:MAIn:VIDeo:SCAN { RATE1 | RATE2 | RATE3 | RATE4 | RATE5 }`

`TRIGger:MAIn:VIDeo:SCAN?`

**Arguments** `RATE1` specifies a 15 to 20 kHz line rate range.

`RATE2` specifies a 20 to 25 kHz line rate range.

`RATE3` specifies a 25 to 35 kHz line rate range.

`RATE4` specifies a 35 to 50 kHz line rate range.

`RATE5` specifies a 50 to 65 kHz line rate range.

**Examples** `TRIGGER:MAIN:VIDeo:SCAN RATE1`
selects rate 1.

## TRIGger:MAIn:VIDeo:SOUrce

Sets or queries the source for the main video trigger. This is equivalent to selecting the source in the video trigger menu's **Source** side menu.

**Group** Trigger

**Syntax**

```
TRIGger:MAIn:VIDeo:SOUrce { CH<x> | EXT | EXT10 | LINE }
```

```
TRIGger:MAIn:VIDeo:SOUrce?
```

**Arguments**

`CH<x>` specifies one of the input channels (CH1 or CH2).

`EXT` specifies the external trigger.

`EXT10` specifies the external trigger attenuated by x10.

`LINE` specifies triggering from the power line frequency.

**Examples**

```
TRIGGER:MAIN:VIDEO:SCAN RATE1
```

specifies the line rate range for commercial broadcast television.

## TRIGger:STATE? (Query Only)

Returns the current state of the triggering system.

**Group** Trigger

**Syntax** TRIGger:STATE?

**Returns** READy indicates that all pretrigger information has been acquired and the oscilloscope is ready to accept a trigger.

PARTial indicates that the main trigger has occurred and the oscilloscope is waiting for the runs-after delay to complete.

TRIGger indicates that the oscilloscope has accepted a trigger and is acquiring the posttrigger information.

AUTO indicates that the oscilloscope is in auto mode and acquires data even in the absence of a trigger.

SAVE indicates that acquisition is stopped or that all channels are off.

ARMed indicates that the oscilloscope is acquiring pretrigger information. All triggers are ignored when TRIGger:STATE is ARMed.

**Examples** TRIGGER:STATE?
might return READY, indicating that pretrigger data has been acquired and the oscilloscope is waiting for a trigger.

## *TST? (Query Only)

(Self-Test) Tests the GPIB or RS-232 interface and returns a 0.

**Group** Miscellaneous

**Syntax** *TST?

## UNLock (No Query Form)

Unlocks the front panel. This command is equivalent to LOCk NONe.

***NOTE.*** *If the oscilloscope is in the Remote With Lockout State (RWLS), the UNLock command has no effect. For more information see the ANSI-IEEE Std. 488.1-1987 Standard Digital Interface for Programmable Instrumentation, section 2.8.3 on RL State Descriptions.*

**Group** Miscellaneous

**Related Commands** LOCk

**Syntax** `UNLock ALL`

**Arguments** `ALL` specifies all front-panel buttons and knobs.

## VERBos

Sets and queries the verbose state that controls the length of keywords on query responses. Keywords can be both headers and arguments. This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk).

**Group** Miscellaneous

**Related Commands** HEADer, *LRN?, SET?

**Syntax**

```
VERBos { OFF | ON | <NR1> }
VERBos?
```

**Arguments**

ON or <NR1> $\neq$ 0 sets the verbose state true, which returns full-length keywords for applicable setting queries.

OFF or <NR1> = 0 sets the verbose state false, which returns minimum-length keywords for applicable setting queries.

**Examples**

```
VERBOS ON
```
sets the verbose state true.

```
VERBOS?
```
might return the value 1, showing that the verbose state is true.

## *WAI (No Query Form)

(Wait) Prevents the oscilloscope from executing further commands or queries until all pending operations finish. This command allows you to synchronize the operation of the oscilloscope with your application program. Synchronization methods are described on page 3–7.

**Group** Status and Error

**Related Commands** BUSY?, *OPC

**Syntax** `*WAI`

## WAVFrm? (Query Only)

Returns WFMPre? and CURVe? data for the waveform or waveforms as specified by the DATa:SOUrce command. This command is equivalent to sending WFMPre?; CURVe?.

**Group** Waveform

**Related Commands** CURVe?, DATa:SOUrce, WFMPre?

**Syntax** `WAVFrm?`

## WFMPre? (Query Only)

Returns the waveform formatting data for the waveform or waveforms as specified by the DATa:SOUrce command. Channel and math waveforms specified by the DATA:SOUrce command must be displayed.

**Group** Waveform

**Related Commands** WAVFrm?

**Syntax** `WFMPre?`

**Returns** The format of the response is:

```
BYT_Nr <NR1>;BIT_Nr <NR1>;ENCdg { ASC | BIN };
BN_Fmt { RI | RP };BYT_Or { LSB | MSB };
<wfm>:WFID <Qstring>;NR_PT <NR1>;PT_FMT { ENV | Y };
XUNit <QString>;XINcr <NR3>;PT_Off <NR1>;YUNit <QString>;YMUlt
<NR3>; YOFf <NR3>;YZEro<NR3>[;<wfm>:
WFID <Qstring>;NR_PT <NR1>;PT_FMT{ ENV | Y };
XUNit<QString>;XINcr <NR3>;PT_Off <NR1>;YUNit <QString>; YMUlt
<NR3>; YOFf <NR3>;YZEro <NR3>...]
```

## WFMPre:BIT_Nr

Returns the number of bits per binary waveform point for the waveform or waveforms as specified by the DATa:SOUrce command. The WFMPre:BIT_Nr command is ignored on input.

**Group** Waveform

**Related Commands** DATa:WIDth, WFMPre:BYT_Nr

**Syntax**

```
WFMPre:BIT_Nr <NR1>
```

```
WFMPre:BIT_Nr?
```

**Arguments** <NR1> is either 8 or 16, and is equivalent to WFMPre:BYT_Nr * 8.

**Examples**

```
WFMPRE:BIT_NR?
```

might return 8, indicating that there are 8 bits per waveform point.

## WFMPre:BN_Fmt

Sets or queries the format of binary data for the waveform or waveforms specified by the DATa:SOUrce command.

**Group** Waveform

**Related Commands** DATa:ENCdg, WFMPre:BYT_Or, WFMPre:ENCdg

**Syntax**

```
WFMPre:BN_Fmt { RI | RP }
```

```
WFMPre:BN_Fmt?
```

**Arguments** RI specifies signed integer data-point representation.

RP specifies positive integer data-point representation.

**Examples**

```
WFMPRE:BN_FMT RP
```

specifies that the binary waveform data are positive integer data-points.

## WFMPre:BYT_Nr

Sets or queries the binary field data width for the first ordered waveform as specified by the DATa:SOUrce command. This command is equivalent to the DATa:WIDth command.

**Group** Waveform

**Related Commands** DATa:WIDth, WFMPre:BIT_Nr

**Syntax** `WFMPre:BYT_Nr <NR1>`

`WFMPre:BYT_Nr?`

**Arguments** `<NR1>` is the number of bytes per point and can be 1 or 2.

**Examples** `WFMPRE:BYT_NR 2`
specifies that there are 2 bytes per waveform data point.

## WFM_Pre:BYT_Or

Selects which byte of binary waveform data is transmitted first during a waveform data transfer when DATa:WIDth (or WFMPre:BYT_Nr) is set to 2.

**Group** Waveform

**Related Commands** DATa:ENCdg, WFMPre:BN_Fmt, WFMPre:ENCdg

**Syntax** `WFMPre:BYT_Or { LSB | MSB }`

`WFMPre:BYT_Or?`

**Arguments** `LSB` selects the least significant byte to be transmitted first.

`MSB` selects the most significant bye to be transmitted first.

**Examples** `WFMPRE:BYT_OR MSB`
specifies that the most significant byte in the waveform data is transferred first.

## WFM_Pre:ENCdg

Sets or queries the type of encoding for waveform data transferred with the CURVe command.

**Group** Waveform

**Related Commands** DATa:ENCdg, WFMPre:BYT_Or, WFMPre:BN_Fmt

**Syntax**

```
WFMPre:ENCdg { ASC | BIN }
```

```
WFMPre:ENCdg?
```

**Arguments**

`ASC` specifies ASCII curve data.

`BIN` specifies binary curve data.

**Examples**

```
WFMPRE:ENCDG ASC
```
specifies that the waveform data is in ASCII format.

```
WFMPRE:ENCDG?
```
might return `BIN`, indicating that the waveform data is in binary format.

## WFM_Pre:PT_Fmt (No Query Form)

Selects the point format of the waveform data for the first ordered waveform as specified by the DATa:SOUrce command.

**Group** Waveform

**Syntax** `WFMPre:PT_Fmt { ENV | Y }`

**Arguments** `ENV` specifies that the waveform is transmitted as maximum and minimum point pairs. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

$$X_n = 0 + XINcr\ (n - PT\_Off)$$

$$Y_{n_{max}} = YZEro + YMUlt\ (y_{n_{max}} - YOFf)$$

$$Y_{n_{min}} = YZEro + YMUlt\ (y_{n_{min}} - YOFf)$$

`Y` specifies a normal waveform where one ASCII or binary data point is transmitted for each point in the waveform record. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

$$X_n = 0 + XINcr\ (n - PT\_Off)$$

$$Y_n = YZEro + YMUlt\ (y_n - YOFf)$$

**Examples** `WFMPRE:PT ENV`
sets the waveform data point format to enveloped.

## WFM_Pre:PT_Off (No Query Form)

Specifies the trigger point within the waveform record for the reference waveform specified by the DATa:DESTination command.

**Group** Waveform

**Related Commands** HORizontal:TRIGger:POsition

**Syntax** `WFMPre:PT_Off <NR1>`

**Arguments** <NR1> = 0 to the record length (1000), and is the position of the trigger point relative to DATa:STARt (<nr1> can be negative).

**Examples** `WFMPRE:PT_OFF 1`
specifies that the trigger point is the first point in the waveform record.

## WFMPre:XINcr (No Query Form)

Specifies the horizontal sampling interval for the reference waveform specified by the DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:XINcr <NR3>`

**Arguments** <NR3> is the sampling interval, in seconds per point.

## WFMPre:YMUlt (No Query Form)

Specifies the vertical scale factor for the reference waveform specified by the DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:YMUlt <NR3>`

**Arguments** `<NR3>` is the vertical scale factor, in YUNits (usually volts) per division.

## WFMPre:YOFf (No Query Form)

Specifies the offset of the vertical component for the reference waveform specified by the DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:YOFf <NR3>`

**Arguments** `<NR3>` is the vertical offset in digitizing levels.

## WFMPre:YZEro (No Query Form)

Specifies the offset for the reference waveform specified by the DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:YZEro <NR3>`

**Arguments** <NR3> is of the offset, in YUNits (usually volts).

Table 2–26 lists additional WFMPre commands that are included for compatibility purposes.

***NOTE.*** *These commands do not support a query form, and all information is ignored.*

**Table 2–26: Additional WFMPre commands**

| Command | Argument | Description |
|---|---|---|
| WFMPre:NR_PT | <NR1> | Number of waveform points |
| WFMPre:WFId | <QString> | Waveform identifier |
| WFMPre:XUNit | <QString> | Horizontal units |
| WFMPre:XMUlt | <NR3> | Horizontal (X-axis) scale factor |
| WFMPre:XOFf | <NR3> | Horizontal (X-axis) offset |
| WFMPre:XZEro | <NR3> | Horizontal (X-axis) origin offset |
| WFMPre:YUNit | <QString> | Vertical units |
| WFMPre:ZMUlt | <NR3> | Z-axis scale factor |
| WFMPre:ZOFf | <NR3> | Z-axis offset |
| WFMPre:ZUNit | <QString> | Z-axis units |
| WFMPre:ZZEro | <NR3> | Z-axis origin offset |

***NOTE.*** *When returning WFMPRE:<wfm> information from the oscilloscope, <wfm> specifies the waveform source (CH<x>, MATH1, or REF<x>). The source must also be set using the DAta:SOUrce command. When sending WFMPRE:<wfm> information to the oscilloscope, the <wfm> specification is ignored and the reference location specified by DATa:DESTination is used instead.*

## WFMPre:<wfm>? (Query Only)

Returns the waveform formatting data for the waveform specified by the DATa:SOUrce command. Channel and math waveforms must be displayed before they can be queried. Querying an invalid reference waveform generates an execution error.

**Group** Waveform

**Syntax** `WFMPre:<wfm>?`

**Returns** The format of the response is:

```
<wfm>:WFID <Qstring>;NR_PT <NR1>;PT_FMT { ENV | Y };
XUNit <QString>;XINcr <NR3>;PT_Off <NR1>;YUNit <QString>;YMUlt
<NR3>;YOFf <NR3>;YZEro <NR3>[;<wfm>:WFID <Qstring>;NR_PT <NR1>;
PT_FMT { ENV | Y };XUNit <QString>;XINcr <NR3>;
PT_Off <NR1>;YUNit <QString>;YMUlt <NR3>;YOFf <NR3>;
YZEro <NR3>...]
```

## WFMPre:<wfm> :NR_Pt

Sets or queries the number of points that are in the transmitted waveform record. This value is ignored on input.

**Related Commands** DATa:DESTination

**Group** Waveform

**Syntax** `WFMPre:<wfm>:NR_Pt <NR1>`

`WFMPre:<wfm>:NR_Pt?`

**Arguments** <NR1> is the number of data points. If DATa:WIDth is 2, then there are twice as many bytes.

<NR1> = 0 means that the waveform record is of an unspecified length.

**Examples** `WFMPRE:CH1:NR_Pt?`
might return 1000 as the number of data points in the waveform record transferred from channel 1.

## WFMPre:<wfm>:PT_Fmt

Selects the data point format for the waveform selected by the DATa:SOUrce command. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

**Group** Waveform

**Related Commands** DATa:DESTination

**Syntax**

```
WFMPre:<wfm>:PT_Fmt { ENV | Y }
```

```
WFMPre:<wfm>:PT_Fmt?
```

**Arguments** ENV specifies that the waveform is transmitted as maximum and minimum point pairs. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

$$X_n = 0 + XINcr\ (n - PT\_Off)$$

$$Y_{n_{max}} = YZEro + YMUlt\ (y_{n_{max}} - YOFf)$$

$$Y_{n_{min}} = YZEro + YMUlt\ (y_{n_{min}} - YOFf)$$

Y specifies a normal waveform where one ASCII or binary data point is transmitted for each point in the waveform record. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

$$X_n = 0 + XINcr\ (n - PT\_Off)$$

$$Y_n = YZEro + YMUlt\ (y_n - YOFf)$$

**Examples**

```
WFMPRE:MATH1:PT_FMT?
```

might return ENV, indicating that the MATH1 waveform data format is enveloped.

## WFMPre:<wfm>:PT_Off

Returns the trigger point within the waveform record. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:PT_Off <NR1>`

`WFMPre:<wfm>:PT_Off?`

**Arguments** <NR1> = 0 to the record length (1000), and is the position of the trigger point relative to DATa:STARt when queried.

**Examples** `WFMPRE:CH1:PT_OFF?`
returns 0 indicating the trigger position within the waveform record.

## WFMPre:<wfm>:WFId

Returns information about the waveform such as input coupling, volts/division, time/division, acquisition mode, and record length.

The WFMPre:<wfm>:WFId command is ignored on input.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:WFId <QString>`

`WFMPre:<wfm>:WFId?`

**Arguments** <QString> is the waveform identifier string.

## WFMPre:<wfm>:XINcr

Sets or queries the horizontal sampling interval of the specified waveform. The command and query form ignores the <wfm> parameter; instead, the instrument uses the reference location specified by DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:XINcr <NR3>`

`WFMPre:<wfm>:XINcr?`

**Arguments** `<NR3>` is the sampling interval.

## WFMPre:<wfm>:XUNit

Sets or queries the horizontal (X-axis) units of the waveform data.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:XUNit <QString>`

`WFMPre:<wfm>:XINcr?`

**Arguments** `<QString>` is `"s"` for seconds or `"Hz"` for Hertz.

**Examples** `WFMPRE:CH1:XUNIT?`
returns `"s"`, indicating that the horizontal units for channel 1 are seconds.

## WFMPre:<wfm>:YMUlt

Sets or queries the vertical scale factor, in YUNit(s) per digitizing level. The command and query form ignores the <wfm> parameter; instead, the instrument uses the reference location specified by DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:YMUlt <NR3>`

`WFMPre:<wfm>:YMUlt?`

**Arguments** <NR3> is the scale factor, in YUNits (usually volts), per digitizing level.

## WFMPre:<wfm>:YOFf

Sets or queries the vertical position of the waveform. On input, <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent. The command and query form ignores the <wfm> parameter; instead, the instrument uses the reference location specified by DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:YOFf <NR3>`

`WFMPre:<wfm>:YOFf?`

**Arguments** <NR3> is the position in digitizing levels.

## WFMPre:<wfm>:YUNit

Sets or queries the vertical (Y-axis) units of the waveform data.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:YUNit <QString>`

`WFMPre:<wfm>:YUNit?`

**Arguments** `<QString>` is `"V"` for volts `"VV"` for volts$^2$, or `"dB"` for decibels, and specifies the units.

**Examples** `WFMPRE:CH2:YUNIT?`
might return `"V"`, meaning that the units for the vertical component of the channel 2 waveform data are volts.

## WFMPre:<wfm>:YZEro

Sets or queries the vertical (Y-axis) offset voltage. On input, <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent. The command and query form ignores the <wfm> parameter; instead, the instrument uses the reference location specified by DATa:DESTination command.

**Group** Waveform

**Syntax** `WFMPre:<wfm>:YZEro <NR3>`

`WFMPre:<wfm>:YZEro?`

**Arguments** `<NR3>` is the offset in YUNits (usually volts).

## ZOOm:VERTical:POSition

Sets or queries the vertical position of waveforms.

***NOTE.*** *ZOOm commands affect only REF and MATH waveforms.*

**Group** Vertical

**Syntax** `ZOOm:VERTical:POSition <NR3>`

`ZOOm:VERTical:POSition?`

**Arguments** `<NR3>` is the vertical position, in divisions.

**Examples** `ZOOM:VERTICAL:POSITION?`
might return `:ZOOM:VERTICAL:POSITION 0`

## ZOOm:VERTical:SCALe

Sets or queries the vertical compression or expansion factor.

***NOTE.*** *ZOOm commands affect only REF and MATH waveforms.*

**Group** Vertical

**Syntax** `ZOOm:VERTical:SCALe <NR3>`

`ZOOm:VERTical:SCALe?`

**Arguments** `<NR3>` is the amount of vertical compression or expansion.

**Examples** `ZOOM:VERTICAL:SCALE?`
might return `:ZOOM:VERTICAL:SCALE 1.0E0`

# Status and Events

The oscilloscope provides a status and event reporting system for the GPIB and RS-232 interfaces. This system informs you of certain significant events that occur within the oscilloscope.

The oscilloscope status handling system consists of five 8-bit registers and two queues. This section describes these registers and components and explains how the event handling system operates.

## Registers

The registers in the event handling system fall into two functional groups:

- The Standard Event Status Register (SESR) and the Status Byte Register (SBR) contain information about the status of the oscilloscope. These registers are the Status Registers.
- The Device Event Status Enable Register (DESER), the Event Status Enable Register (ESER), and the Service Request Enable Register (SRER) determine whether selected types of events are reported to the Status Registers and the Event Queue. These three registers are the Enable Registers.

### Status Registers

The Standard Event Status Register (SESR) and the Status Byte Register (SBR) record certain types of events that may occur while the oscilloscope is in use. IEEE Std 488.2–1987 defines these registers.

Each bit in a Status Register records a particular type of event, such as an execution error or service request. When an event of a given type occurs, the oscilloscope sets the bit that represents that type of event to a value of one. (You can disable bits so that they ignore events and remain at zero. See the Enable Registers section on page 3–3.) Reading the status registers tells you what types of events have occurred.

**The Standard Event Status Register (SESR).** The SESR, shown in Figure 3–1, records eight types of events that can occur within the oscilloscope. Use *ESR? to read the SESR register. Reading the register clears the bits of the register so that the register can accumulate information about new events. Table 3–1 shows SESR bit functions.

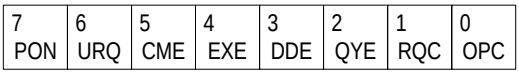



**Figure 3–1: The Standard Event Status Register (SESR)**

**Table 3–1: SESR bit functions**

| Bit | Function |
|---|---|
| 7 (MSB) | **PON** (Power On). Shows that the oscilloscope was powered on. |
| 6 | **URQ** (User Request). Not used. |
| 5 | **CME** (Command Error). Shows that an error occurred while the oscilloscope was parsing a command or query. Command error messages are listed in Table 3–4 on page 3–13. |
| 4 | **EXE** (Execution Error). Shows that an error occurred while the oscilloscope was executing a command or query. Execution error messages are listed in Table 3–5 on page 3–13. |
| 3 | **DDE** (Device Error). Shows that a device error occurred. Device error messages are listed in Table 3–6 on page 3–16. |
| 2 | **QYE** (Query Error). Shows that either an attempt was made to read the Output Queue when no data was present or pending, or that data in the Output Queue was lost. |
| 1 | **RQC** (Request Control). Not used. |
| 0 (LSB) | **OPC** (Operation Complete). Shows that the operation is complete. This bit is set when all pending operations complete following a *OPC command. |

**The Status Byte Register (SBR).** The SBR, shown in Figure 3–2, records whether output is available in the Output Queue, whether the oscilloscope requests service, and whether the SESR has recorded any events.

Use a Serial Poll (GPIB only) or *STB? to read the contents of the SBR. The bits in the SBR are set and cleared depending on the contents of the SESR, the Event Status Enable Register (ESER), and the Output Queue. When you use a Serial Poll to obtain the SBR, bit 6 is the RQS bit. When you use the *STB? query to obtain the SBR, bit 6 is the MSS bit. Reading the SBR does not clear the bits. Table 3–2 shows the SBR bit functions.

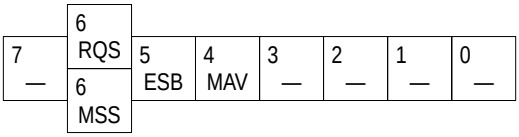



**Figure 3–2: The Status Byte Register (SBR)**

**Table 3–2: SBR bit functions**

| Bit | Function |
|---|---|
| 7 (MSB) | Not used. |
| 6 | **RQS** (Request Service), obtained from a serial poll. Shows that the oscilloscope requests service from the GPIB controller. |
| 6 | **MSS** (Master Status Summary), obtained from *STB?. Summarizes the ESB and MAV bits in the SBR. |
| 5 | **ESB** (Event Status Bit). Shows that status is enabled and present in the SESR. |
| 4 | **MAV** (Message Available). Shows that output is available in the Output Queue. |
| 3 – 0 | Not used. |

## Enable Registers

The DESER, ESER, and SRER allow you to select which events are reported to the Status Registers and the Event Queue. Each Enable Register acts as a filter to a Status Register (the DESER also acts as a filter to the Event Queue), and can prevent information from being recorded in the register or queue.

Each bit in an Enable Register corresponds to a bit in the Status Register it controls. In order for an event to be reported to its bit in the Status Register, the corresponding bit in the Enable Register must be set to one. If the bit in the Enable Register is set to zero, the event is not recorded.

The bits in the Enable Registers are set using various commands. The Enable Registers and the commands used to set them are described below.

**The Device Event Status Enable Register (DESER).** The DESER, shown in Figure 3–3, controls which types of events are reported to the SESR and the Event Queue. The bits in the DESER correspond to those in the SESR, as described earlier.

Use the DESE command to enable and disable the bits in the DESER. Use the DESE? query to read the DESER.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PON | URQ | CME | EXE | DDE | QYE | RQC | OPC |

**Figure 3–3: The Device Event Status Enable Register (DESER)**

**The Event Status Enable Register (ESER).** The ESER, shown in Figure 3–4, controls which types of events are summarized by the Event Status Bit (ESB) in the SBR.

Use the *ESE command to set the bits in the ESER, and use the *ESE? query to read it.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PON | URQ | CME | EXE | DDE | QYE | RQC | OPC |

**Figure 3–4: The Event Status Enable Register (ESER)**

**The Service Request Enable Register (SRER).** The SRER, shown in Figure 3–5, controls which bits in the SBR generate a Service Request (GPIB only) and are summarized by the Master Status Summary (MSS) bit.

Use the *SRE command to set the SRER. Use the *SRE? query to read it. The RQS bit remains set to one until either the Status Byte Register is read by a Serial Poll (GPIB only) or the MSS bit changes back to a zero.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| — | — | ESB | MAV | — | — | — | — |

**Figure 3–5: The Service Request Enable Register (SRER)**

## The Enable Registers and the *PSC Command

The *PSC command controls the contents of the Enable Registers at power-on. Sending *PSC 1 sets the Enable Registers at power on as follows:

- DESER 255 (equivalent to a DESe 255 command)
- ESER 0 (equivalent to an *ESE 0 command)
- SRER 0 (equivalent to an *SRE 0 command)

Sending *PSC 0 lets the Enable Registers maintain their values in nonvolatile memory through a power cycle.

***NOTE**. To enable the PON (Power On) event to generate a Service Request (GPIB only), send *PSC 0, use the DESe and *ESE commands to enable PON in the DESER and ESER, and use the *SRE command to enable bit 5 in the SRER. Subsequent power-on cycles will generate a Service Request (GPIB only).*

# Queues

The oscilloscope status and event reporting system contains two queues: the Output Queue and the Event Queue.

## The Output Queue

The Output Queue stores query responses waiting to be output. The oscilloscope empties the Output Queue each time it receives a new command or query message. This means you must read any query response before you send the next command or query, or you will lose responses to earlier queries. Also, an error may result.

---

***NOTE.*** *When a controller sends a query, an <EOM>, and a second query, the digitizing oscilloscope normally clears the first response and outputs the second while reporting a Query Error (QYE bit in the ESER) to indicate the lost response. A fast controller, however, may receive a part or all the first response as well. To avoid this situation, the controller should always read the response immediately after sending any terminated query message or send a DCL (Device Clear) before sending the second query.*

---

## The Event Queue

The Event Queue stores detailed information on up to 20 events. If more than 20 events stack up in the Event Queue, the 20th event is replaced by event code 350, “Too many events.”

Read the Event Queue with EVENT? (which returns only the event number), with EVMsg? (which returns the event number and a text description of the event), or with ALLEV? (which returns all the event numbers along with a description of the event). Reading an event removes it from the queue.

Before reading an event from the Event Queue, you must use *ESR? to read the summary of the event from the SESR. This makes the events summarized by *ESR? available to EVENT? and EVMSG?, and empties the SESR.

Reading the SESR erases any events that were summarized by previous *ESR? reads but not read from the Event Queue. Events that follow an *ESR? read are put in the Event Queue but are not available until *ESR? is used again.

## Event Handling Sequence

Figure 3–6 shows how to use the status and event handling system. In the explanation that follows, numbers in parentheses refer to numbers in Figure 3–6.

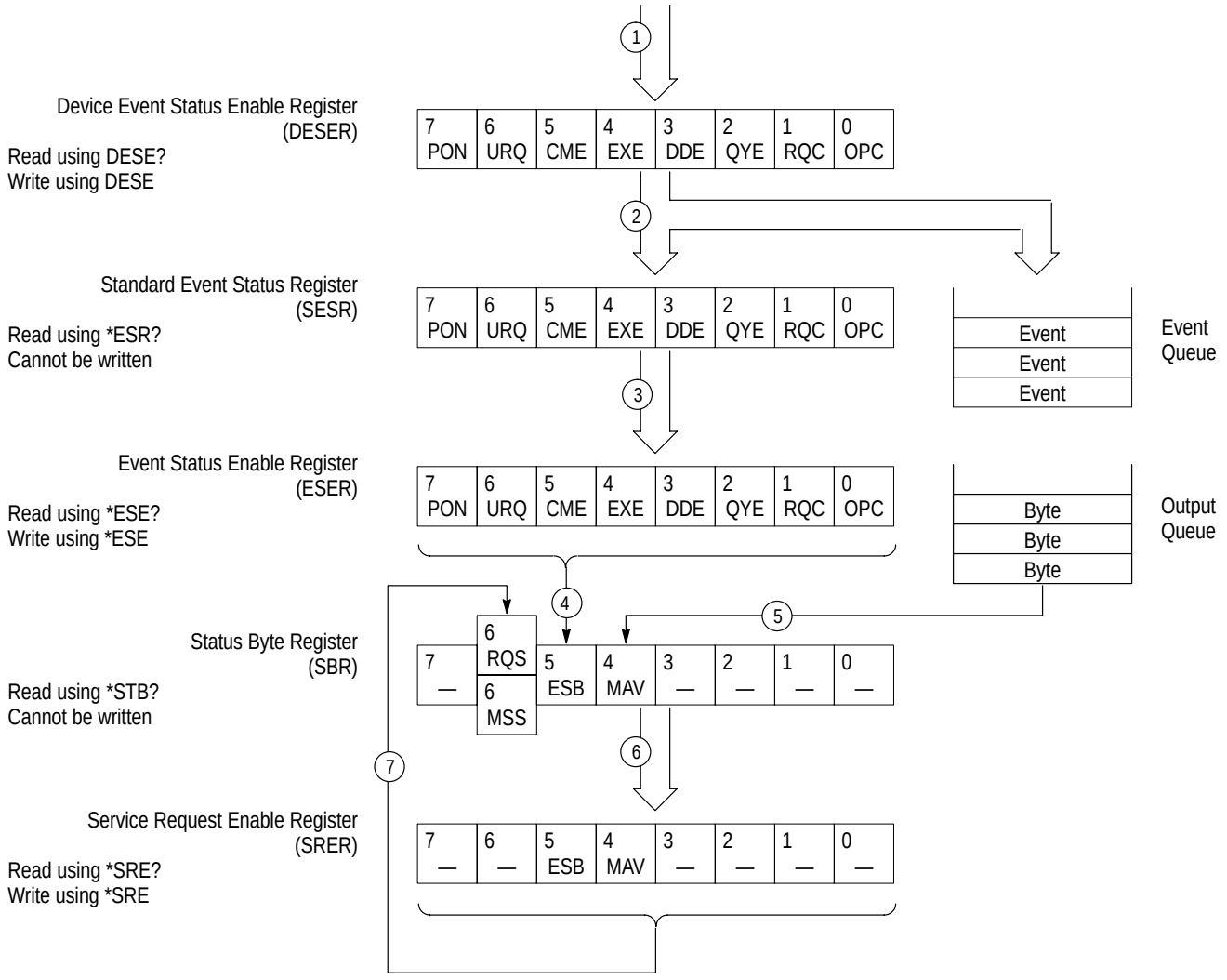



**Figure 3–6: Status and event handling process**

When an event occurs, a signal is sent to the DESER (1). If that type of event is enabled in the DESER (that is, if the bit for that event type is set to 1), the appropriate bit in the SESR is set to one and the event is recorded in the Event Queue (2). If the corresponding bit in the ESER is also enabled (3), then the ESB bit in the SBR is set to one (4).

When output is sent to the Output Queue, the MAV bit in the SBR is set to one (5).

When a bit in the SBR is set to one and the corresponding bit in the SRER is enabled (6), the MSS bit in the SBR is set to one and a service request (GPIB only) is generated (7).

## Synchronization Methods

Although most commands are completed almost immediately after being received by the oscilloscope, some commands start a process that requires more time. For example, once a HARDCOPY START command is executed, it may be a few seconds before the hardcopy operation is complete. Rather than remain idle while the operation is in process, the oscilloscope continues processing other commands. This means that some operations are not completed in the order that they were sent.

There may be times when the result of an operation is dependent on the result of an earlier one, and you must be assured that the first operation has completed before processing the next one. The status and event reporting system provides ways to do this.

For example, a typical application would be to acquire a single-sequence waveform, and then take a measurement on the acquired waveform. You could use the following command sequence:

/** Set up single-sequence acquisition **/

```
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
```

/** Acquire waveform data **/

```
ACQUIRE:STATE ON
```

/** Set up the measurement parameters **/

```
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
```

/** Take amplitude measurement on acquired data **/

```
MEASUREMENT:IMMED:VALUE?
```

The acquisition of the waveform requires extended processing time and may not complete before the amplitude measurement is taken (See Figure 3–7). This will result in an incorrect amplitude value.

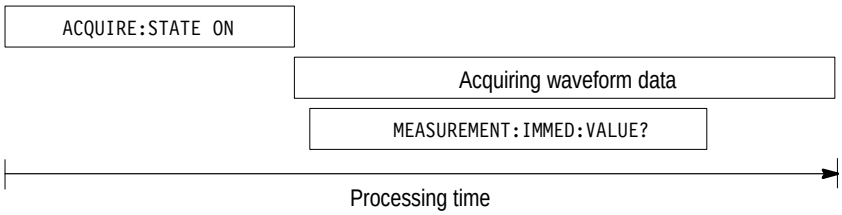



**Figure 3–7: Command processing without using synchronization**

The acquisition of the waveform must be completed before the measurement can be taken on the acquired data. This is achieved by synchronizing the program so that the measurement command is not processed by the oscilloscope until the acquisition is complete. Figure 3–8 shows the desired processing sequence.

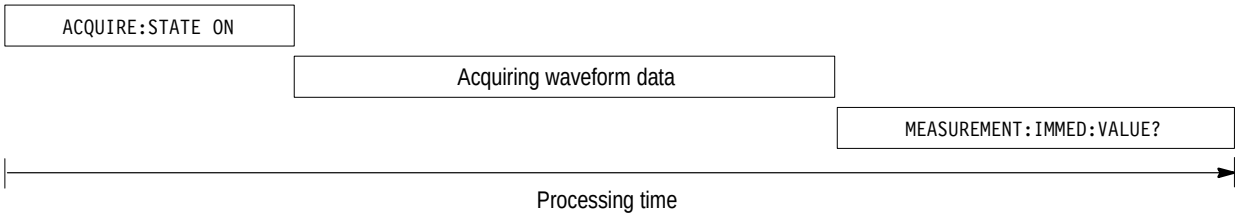



**Figure 3–8: Processing sequence with synchronization**

Four commands can be used to synchronize the operation of the oscilloscope with your application program: *WAI, BUSY?, *OPC, and *OPC?.

## Using the *WAI Command

You can force commands to execute sequentially by using the *WAI command. This command forces completion of the previous commands before processing new ones.

The same command sequence using the *WAI command for synchronization looks like this:

/* Set up single-sequence acquisition */

```
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
```

```
/* Acquire waveform data */

    ACQUIRE:STATE ON

/* Set up the measurement parameters */

    MEASUREMENT:IMMED:TYPE AMPLITUDE
    MEASUREMENT:IMMED:SOURCE CH1

/* Wait until the acquisition is complete before taking the measurement */

    *WAI

/* Take amplitude measurement on acquired data */

    MEASUREMENT:IMMED:VALUE?
```

Though *WAI is one of the easiest ways to achieve synchronization, it is also the most costly. The processing time of the oscilloscope is slowed, since it is processing a single command at a time. This time could be spent doing other tasks.

The controller can continue to write commands to the input buffer, but the commands are not processed by the oscilloscope until all operations in process are complete. If the input buffer becomes full, the controller will be unable to write any more commands to the buffer and will result in a time-out.

### Using the BUSY Query

BUSY? allows you to find out whether the oscilloscope is busy processing a command that has an extended processing time, such as single-sequence acquisition.

The same command sequence using BUSY? for synchronization looks like this:

```
/* Set up single-sequence acquisition */

    SELECT:CH1 ON
    ACQUIRE:MODE SAMPLE
    ACQUIRE:STOPAFTER SEQUENCE

/* Acquire waveform data */

    ACQUIRE:STATE ON

/* Set up the measurement parameters */

    MEASUREMENT:IMMED:TYPE AMPLITUDE
    MEASUREMENT:IMMED:SOURCE CH1

/* Wait until the acquisition is complete before taking the measurement */

    While BUSY? keep looping
```

/* Take amplitude measurement on acquired data */

```
MEASUREMENT:IMMED:VALUE?
```

This sequence lets you create your own wait loop rather than using the *WAI command. An advantage to using BUSY? is that you eliminate the possibility of a time-out caused by writing too many commands to the input buffer. The controller is still tied up, though, and the repeated BUSY? results in more bus traffic.

## Using the *OPC Command

If the corresponding status registers are enabled, the *OPC command sets the OPC bit in the Standard Event Status Register (SESR) when an operation is complete. You can use this command in conjunction with either a serial poll or service request handler to achieve synchronization.

**Serial Poll Method (GPIB Only).** Enable the OPC bit in the Device Event Status Enable Register (DESER) and the Event Status Enable Register (ESER) using the DESE and *ESE commands. When the operation is complete, the OPC bit in the Standard Event Status Register (SESR) is enabled, and the Event Status Bit (ESB) in the Status Byte Register is enabled.

The same command sequence using the *OPC command for synchronization with serial polling looks like this:

/* Set up single-sequence acquisition */

```
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
```

/* Enable the status registers */

```
DESE 1
*ESE 1
*SRE 0
```

/* Acquire waveform data */

```
ACQUIRE:STATE ON
```

/* Set up the measurement parameters */

```
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
```

/* Wait until the acquisition is complete before taking the measurement */

```
*OPC
While serial poll = 0, keep looping
```

/* Take amplitude measurement on acquired data */

```
MEASUREMENT:IMMED:VALUE?
```

This technique requires less bus traffic than did looping on BUSY?.

**Service Request Method (GPIB Only).** Enable the OPC bit in the Device Event Status Enable Register (DESER) and the Event Status Enable Register (ESER) using the DESE and *ESE commands. Also, enable service requests by setting the ESB bit in the Service Request Enable Register (SRER) using the *SRE command. When the operation is complete, a Service Request is generated.

The same command sequence using the *OPC command for synchronization looks like this:

/* Set up single-sequence acquisition */

```
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
```

/* Enable the status registers */

```
DESE 1
*ESE 1
*SRE 32
```

/* Acquire waveform data */

```
ACQUIRE:STATE ON
```

/* Set up the measurement parameters */

```
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
```

/* Wait until the acquisition is complete before taking the measurement */

```
*OPC
```

Program can now do different tasks such as talk to other devices. The SRQ, when it comes, interrupts those tasks and returns control to this task.

/* Take amplitude measurement on acquired data */

```
MEASUREMENT:IMMED:VALUE?
```

This technique is more efficient but requires more sophisticated programming.

### Using the *OPC Query

*OPC? places a 1 in the Output Queue once an operation is complete. A time-out could occur if you try to read the output queue before there is any data in it.

The same command sequence using *OPC? for synchronization looks like this:

/* Set up single-sequence acquisition */

```
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
```

/* Acquire waveform data */

```
ACQUIRE:STATE ON
```

/* Set up the measurement parameters */

```
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
```

/* Wait until the acquisition is complete before taking the measurement */

```
*OPC?
Wait for read from Output Queue.
```

/* Take amplitude measurement on acquired data */

```
MEASUREMENT:IMMED:VALUE?
```

This is the simplest approach. It requires no status handling or loops. However, you must set the controller time-out for longer than the acquisition operation.

## Messages

Tables 3–3 through 3–9 list all the programming interface messages the oscilloscope generates in response to commands and queries.

For most messages, a secondary message from the oscilloscope gives more detail about the cause of the error or the meaning of the message. This message is part of the message string and is separated from the main message by a semicolon.

Each message is the result of an event. Each type of event sets a specific bit in the SESR and is controlled by the equivalent bit in the DESER. Thus, each message is associated with a specific SESR bit. In the message tables that follow, the associated SESR bit is specified in the table title.

Table 3–3 shows the messages when the system has no events or status to report. These have no associated SESR bit.

**Table 3–3: No event messages**

| Code | Message |
|---|---|
| 0 | No events to report – queue empty |
| 1 | No events to report – new events pending *ESR? |

Table 3–4 shows the error messages generated by improper command syntax. Check that the command is properly formed and that it follows the rules in the Command Syntax chapter starting on page 2–1.

**Table 3–4: Command error messages — CME Bit 5**

| Code | Message |
|---|---|
| 100 | Command error |
| 102 | Syntax error |
| 103 | Invalid separator |
| 104 | Data type error |
| 105 | GET not allowed |
| 108 | Parameter not allowed |
| 110 | Command header error |
| 111 | Header separator error |
| 112 | Program mnemonic too long |
| 113 | Undefined header |
| 161 | Invalid block data (indefinite length blocks are not allowed over the RS-232) |

Table 3–5 lists the errors that are detected during execution of a command. In these error messages, you should read “macro” as “alias.”

**Table 3–5: Execution error messages — EXE Bit 4**

| Code | Message |
|---|---|
| 200 | Execution error |
| 201 | Invalid while in local |
| 210 | Trigger error |
| 211 | Trigger ignored |
| 212 | Arm ignored |
| 220 | Parameter error |

**Table 3–5: Execution error messages — EXE Bit 4 (Cont.)**

| Code | Message |
|---|---|
| 221 | Settings conflict |
| 222 | Data out of range |
| 223 | Too much data |
| 224 | Illegal parameter value |
| 230 | Data corrupt or stale |
| 240 | Hardware error |
| 241 | Hardware missing |
| 242 | Hardware configuration error |
| 243 | Hardware I/O device error |
| 250 | Mass storage error |
| 251 | Missing mass storage |
| 252 | Missing media |
| 253 | Corrupt media |
| 254 | Media full |
| 255 | Directory full |
| 256 | File name not found |
| 257 | File name error |
| 258 | Media protected |
| 260 | Expression error |
| 261 | Math error in expression |
| 2200 | Measurement error, Measurement system error |
| 2201 | Measurement error, Zero period |
| 2202 | Measurement error, No period found |
| 2203 | Measurement error, No period, second waveform |
| 2204 | Measurement error, Low signal amplitude |
| 2205 | Measurement error, Low amplitude, second waveform |
| 2206 | Measurement error, Invalid gate |
| 2207 | Measurement error, Measurement overflow |
| 2208 | Measurement error, Waveform does not cross Mid Ref |
| 2209 | Measurement error, No second Mid Ref crossing |
| 2210 | Measurement error, No Mid Ref crossing, second waveform |
| 2211 | Measurement error, No backwards Mid Ref crossing |
| 2212 | Measurement error, No negative crossing |

**Table 3–5: Execution error messages — EXE Bit 4 (Cont.)**

| Code | Message |
|---|---|
| 2213 | Measurement error, No positive crossing |
| 2214 | Measurement error, No crossing |
| 2215 | Measurement error, No crossing, second waveform |
| 2216 | Measurement error, No crossing, target waveform |
| 2217 | Measurement error, Constant waveform |
| 2218 | Measurement error, Unused |
| 2219 | Measurement error, No valid edge – No arm sample |
| 2220 | Measurement error, No valid edge – No arm cross |
| 2221 | Measurement error, No valid edge – No trigger cross |
| 2222 | Measurement error, No valid edge – No second cross |
| 2223 | Measurement error, waveform mismatch |
| 2224 | Measurement error, WAIT calculating |
| 2225 | Measurement error, No waveform to measure |
| 2226 | Null Waveform |
| 2227 | Positive and Negative Clipping |
| 2228 | Measurement error, Positive Clipping |
| 2229 | Measurement error, Negative Clipping |
| 2230 | Measurement error, High Ref < Low Ref |
| 2235 | Math error, Invalid math description |
| 2240 | Invalid password |
| 2241 | Waveform request is invalid |
| 2242 | Data start and stop > record length |
| 2243 | Waveform requested is not a data source |
| 2244 | Waveform requested is not turned on |
| 2245 | Saveref error, Selected channel is turned off |
| 2246 | Saveref error, Selected channel data invalid |
| 2248 | Saveref error, Source reference data invalid |
| 2260 | Calibration error |
| 2270 | Alias error |
| 2271 | Alias syntax error |
| 2272 | Alias execution error |
| 2273 | Illegal alias label |
| 2274 | Alias parameter error |

**Table 3–5: Execution error messages — EXE Bit 4 (Cont.)**

| Code | Message |
|---|---|
| 2275 | Alias definition too long |
| 2276 | Alias expansion error |
| 2277 | Alias redefinition not allowed |
| 2278 | Alias header not found |
| 2279 | Alias label too long |
| 2280 | Alias table full |
| 2285 | Tek Secure® Pass |
| 2286 | Tek Secure® Fail |
| 2301 | Cursor error, Off screen |

Table 3–6 lists the device errors that can occur during oscilloscope operation. These errors may indicate that the oscilloscope needs repair.

**Table 3–6: Device error messages — DDE Bit 3**

| Code | Message |
|---|---|
| 300 | Device-specific error |
| 310 | System error |
| 311 | Memory error |
| 312 | PUD memory lost |
| 313 | Calibration memory lost |
| 314 | Save/recall memory lost |
| 315 | Configuration memory lost |
| 350 | Queue overflow (does not set DDE bit) |
| 361 | Parity error in program message (check parity) |
| 362 | Framing error in program message (check baud rate) |
| 363 | Input buffer overrun (check flagging) |

Table 3–7 lists the system event messages. These messages are generated whenever certain system conditions occur.

**Table 3–7: System event messages**

| Code | Message |
|---|---|
| 400 | Query event |
| 401 | Power on (PON bit 7 set) |
| 402 | Operation complete (OPC bit 0 set) |
| 403 | User request (URQ bit 6 set) |
| 404 | Power fail (DDE bit 3 set) |
| 405 | Request control |
| 410 | Query INTERRUPTED (QYE bit 2 set) |
| 420 | Query UNTERMINATED (QYE bit 2 set) |
| 430 | Query DEADLOCKED (QYE bit 2 set) |
| 440 | Query UNTERMINATED after indefinite response (QYE bit 2 set) |

Table 3–8 lists warning messages that do not interrupt the flow of command execution. These messages notify you that you may get unexpected results.

**Table 3–8: Execution warning messages — EXE Bit 4**

| Code | Message |
|---|---|
| 500 | Execution warning |
| 510 | String data too long, truncated |
| 525 | Parameter underrange |
| 526 | Parameter overrange |
| 527 | Parameter rounded |
| 528 | Parameter out of range |
| 530 | Data stop > stop, Values swapped internally |
| 531 | Data stop > record length, Curve truncated |
| 532 | Curve data too long, Curve truncated |
| 540 | Measurement warning |
| 541 | Measurement warning, Low signal amplitude |
| 542 | Measurement warning, Unstable histogram |
| 543 | Measurement warning, Low resolution |
| 544 | Measurement warning, Uncertain edge |

**Table 3–8: Execution warning messages — EXE Bit 4 (Cont.)**

| Code | Message |
|---|---|
| 545 | Measurement warning, Invalid in minmax |
| 546 | Measurement warning, Need 3 edges |
| 547 | Measurement warning, Clipping positive/negative |
| 548 | Measurement warning, Clipping positive |
| 549 | Measurement warning, Clipping negative |

Table 3–9 shows internal errors that indicate an internal fault in the oscilloscope.

**Table 3–9: Internal warning messages**

| Code | Message |
|---|---|
| 600 | Internal warning |

# Programming Examples

The example programs illustrate methods you can use to control the oscilloscope from the GPIB or RS-232 interface. The diskettes that come with this manual contain listings for these programs.

The programs run on a PC compatible system equipped with a Tektronix (National Instruments) GPIB board and associated drivers or an RS-232 (COM) serial port. For example, the GPIB programs work with a Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB package (see Figure 4–1).

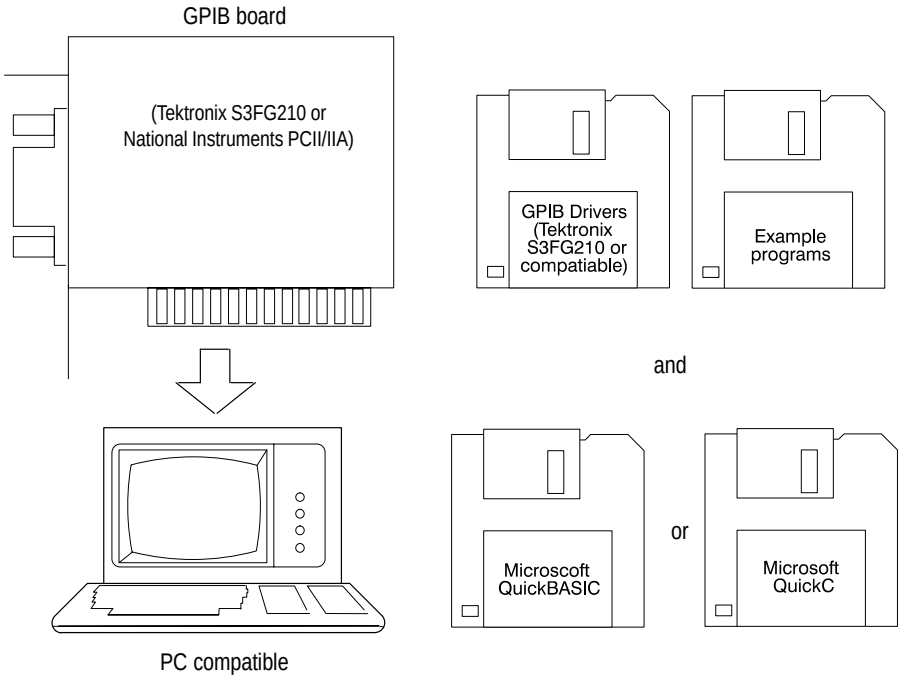



**Figure 4–1: Equipment needed to run the GPIB and RS-232 example programs**

## GPIB Examples

All the example GPIB programs assume that the GPIB system recognizes the oscilloscope as DEV1 and the PC (controller) as GPIB0. You can assign these names using the IBCONF.EXE program.

The example GPIB software includes:

MEAS: measures a parameter of an oscilloscope waveform.

COMM: shows communication between the controller and the oscilloscope.

GETWFM: reads a waveform from an oscilloscope and stores its time and voltage values in a file.

TL: a talker-listener program.

### Compiling the Example programs

The example programs diskette contains programs written in Microsoft QuickBASIC 4.5 and Microsoft QuickC 2.5.

Executable versions of the programs are in the PROGRAMS directory. Source versions are in the SOURCES directory. Within this directory, the QuickBASIC programs are in the Q-BASIC subdirectory and the QuickC programs are in the Quick-C subdirectory.

A README file in each directory explains how to build executable code from the source files provided.

The QuickC directory also comes with sample MAKE files and sample executable files. These have the suffix .MAK.

If you wish to develop code, you will need to use files that come with the GPIB system. Specifically, the QuickBASIC programs use QBDECL.BAS and QBIB.OBJ. The QuickC programs use DECL.H and MCIB.OBJ.

---

***NOTE.*** *The GPIB programs you compile in the Sources directory work with the Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB system. It may take extra steps or changes to get them to work with older Tektronix GURU and other GPIB systems.*

---

**Compiling and Linking Your Example Quick-C Programs.** To make an executable for any of the following files, perform the following steps:

1. Install QuickC. Select the SMALL memory model. Be sure to set up your path so DOS can access the Quick-C directory.

2. Install the Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB board and drivers. Remember to identify the GPIB device as DEV1. This identifier is defined using the `IBCONF.EXE` program.

3. Copy the files from the examples diskette to your hard disk. You might also create a special directory to store them. For example, if the current drive is hard disk C, you want to store the examples in drive C, and the examples diskette is in drive B, type:

   ```
   mkdir examples

   cd examples

   copy B:\gpib\quick-c\*.* .
   ```

4. For this installation, you also want to copy `DECL.H` and `MCIB.OBJ` from your Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB drivers directory to this directory. For example, if the GPIB drivers are in the `gpib-pc` directory and you are in the example programs directory, type:

   ```
   copy \gpib-pc\decl.h .

   copy \gpib-pc\mcib.obj .
   ```

5. To compile and link your TDS sample "C" programs, simply type: `nmake <file name>.mak`

   where <file name> refers to the name of the example program you wish to compile and link. Specifically:

   To compile and link `MEAS.C`, type: `nmake meas.mak`

   To compile and link `COMM.C`, type: `nmake comm.mak`

   To compile and link `GETWFM.C`, type: `nmake getwfm.mak`

   To compile and link `TL.C`, type: `nmake tl.mak`

6. Run the program by typing the program name.

   To run `meas`, type: `meas`

   To run `comm`, type: `comm`

   To run `getwfm`, type: `getwfm`

   To run `tl`, type: `tl`

**Compiling and Linking Your Example QuickBASIC Programs.** To make an executable for any of the following files, perform the following steps:

1. Install QuickBASIC.
2. Install the Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB board and drivers. Remember to reboot your PC to initialize the GPIB drivers.
3. Copy the files from the examples diskette to your hard disk. You might also create a special directory to store them. For example, if the current drive is hard disk C, you want to store the examples in drive C, and the examples diskette is in drive B, type:

   ```
   mkdir examples

   cd examples

   copy b:\gpib\q-basic\*.* .
   ```

4. For this installation, you also want to copy QBDECL.BAS and QBIB.OBJ from your Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB drivers directory to the directory your example programs are in. For example, if the GPIB drivers are in the gpib-pc directory and you are in the example programs directory, type:

   ```
   copy \gpib-pc\qbdecl.bas .

   copy \gpib-pc\qbib.obj .
   ```

5. Perform the following two steps for example programs:

   **a.** Compile the program using the following command:

   ```
   bc /o <file>.bas;
   ```

   where <file> is one of the example program names.

   To compile MEAS.BAS, type: `bc /o meas.bas;`

   To compile COMM.BAS, type: `bc /o comm.bas;`

   To compile GETWFM.BAS, type: `bc /o getwfm.bas;`

   To compile TL.BAS, type: `bc /o tl.bas;`

**b.** Link the compiled program with the qbib.obj module to create the executable program (file.EXE) using the following command:

`link <file>.obj+qbib.obj;`

where `<file>` is one of the above program names.

To link `MEAS.OBJ`, type: `link meas.obj+qbib.obj;`

To link `COMM.OBJ`, type: `link comm.obj+qbib.obj;`

To link `GETWFM.OBJ`, type: `link getwfm.obj+qbib.obj;`

To link `TL.OBJ`, type: `link tl.obj+qbib.obj;`

The `GPIBIO.BAS` file is a collection of input/output routines used by the other programs and is included for proper file compilation.

**6.** Run the program by typing the program name.

To run `meas`, type: `meas`

To run `comm`, type: `comm`

To run `getwfm`, type: `getwfm`

To run `tl`, type: `tl`

---

***NOTE.*** *The example programs disable front-panel operation while they are running, and reenable it when they terminate. If your program terminates prematurely, front-panel operation may remain disabled. To re-enable front-panel operation, do one of the following: cycle power on the oscilloscope or send the GPIB command* `UNLOCK ALL` *to unlock the front panel. You can send the* `UNLOCK ALL` *command with the TL program included in your sample programs disk.*

---

## RS-232 Examples

The example RS-232 programs use the COM port of a PC. The example RS-232 software includes:

MEAS: measures a parameter of an oscilloscope waveform.

COMM: shows communication between the personal computer and the oscilloscope.

GETWFM: reads a waveform from an oscilloscope and stores its time and voltage values in a file.

TL: a talker-listener program.

### Compiling the Example programs

The example programs diskette contains programs written in Microsoft QuickBASIC 4.5.

Executable versions of the programs are in the PROGRAMS directory. Source versions are in the SOURCES directory. Within this directory, the QuickBASIC programs are in the Q-BASIC subdirectory.

A README file in each directory explains how to build executable code from the source files provided.

***NOTE.*** *The programs you compile in the Sources directory may require extra steps or changes to get them to work with your system.*

**Compiling and Linking Your Example QuickBASIC Programs.** To make an executable for any of the following files, perform the following:

1. Install QuickBASIC.
2. Connect the oscilloscope to the COM2 port of the personal computer. Set the oscilloscope RS-232 parameters to the default values.
3. Copy the files from the examples diskette to your hard disk. You might also create a special directory to store them. For example, if the current drive is hard disk C, you want to store the examples in drive C, and the examples diskette is in drive B, type:

```
mkdir examples

cd examples

copy b:\rs232\q-basic\*.* .
```

4. Perform the following two steps for example programs:

   a. Compile the programs using the following commands:

      `bc /o <file>.bas`

      `bc /o /v rs232io.bas`

      where `<file>` is one of the example program names.

      To compile `MEAS.BAS`, type: `bc /o meas.bas`

      To compile `COMM.BAS`, type: `bc /o comm.bas`

      To compile `GETWFM.BAS`, type: `bc /o getwfm.bas`

      To compile `TL.BAS`, type: `bc /o tl.bas`

   b. Link the compiled program with the rs232io.obj module to create the executable program (file.EXE) using the following command:

      `link <file>.obj+rs232io.obj`

      where `<file>` is one of the above program names.

      To link `MEAS.OBJ`, type: `link meas.obj+rs232io.obj`

      To link `COMM.OBJ`, type: `link comm.obj+rs232io.obj`

      To link `GETWFM.OBJ`, type: `link getwfm.obj+rs232io.obj`

      To link `TL.OBJ`, type: `link tl.obj+rs232io.obj`

      The `rs232IO.BAS` file is a collection of input/output routines used by the other programs and is included for proper file compilation.

5. Run the program by typing the program name.

   To run `meas`, type: `meas`

   To run `comm`, type: `comm`

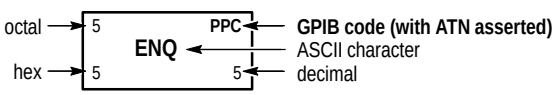
   To run `getwfm`, type: `getwfm`

   To run `tl`, type: `tl`

# Appendix A: ASCII & GPIB Code Chart

| B7 B6 B5 BITS B4 B3 B2 B1 | 0 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1 0 0 | 1 0 1 | 1 1 0 | 1 1 1 |
|---|---|---|---|---|---|---|---|---|
| | CONTROL | | NUMBERS SYMBOLS | | UPPER CASE | | LOWER CASE | |
| 0 0 0 0 | 0 NUL 0 0 | 20 DLE 10 16 | 40 LA0 SP 20 32 | 60 LA16 0 30 48 | 100 TA0 @ 40 64 | 120 TA16 P 50 80 | 140 SA0 ` 60 96 | 160 SA16 p 70 112 |
| 0 0 0 1 | 1 GTL SOH 1 1 | 21 LL0 DC1 11 17 | 41 LA1 ! 21 33 | 61 LA17 1 31 49 | 101 TA1 A 41 65 | 121 TA17 Q 51 81 | 141 SA1 a 61 97 | 161 SA17 q 71 113 |
| 0 0 1 0 | 2 STX 2 2 | 22 DC2 12 18 | 42 LA2 " 22 34 | 62 LA18 2 32 50 | 102 TA2 B 42 66 | 122 TA18 R 52 82 | 142 SA2 b 62 98 | 162 SA18 r 72 114 |
| 0 0 1 1 | 3 ETX 3 3 | 23 DC3 13 19 | 43 LA3 # 23 35 | 63 LA19 3 33 51 | 103 TA3 C 43 67 | 123 TA19 S 53 83 | 143 SA3 c 63 99 | 163 SA19 s 73 115 |
| 0 1 0 0 | 4 SDC EOT 4 4 | 24 DCL DC4 14 20 | 44 LA4 $ 24 36 | 64 LA20 4 34 52 | 104 TA4 D 44 68 | 124 TA20 T 54 84 | 144 SA4 d 64 100 | 164 SA20 t 74 116 |
| 0 1 0 1 | 5 PPC ENQ 5 5 | 25 PPU NAK 15 21 | 45 LA5 % 25 37 | 65 LA21 5 35 53 | 105 TA5 E 45 69 | 125 TA21 U 55 85 | 145 SA5 e 65 101 | 165 SA21 u 75 117 |
| 0 1 1 0 | 6 ACK 6 6 | 26 SYN 16 22 | 46 LA6 & 26 38 | 66 LA22 6 36 54 | 106 TA6 F 46 70 | 126 TA22 V 56 86 | 146 SA6 f 66 102 | 166 SA22 v 76 118 |
| 0 1 1 1 | 7 BEL 7 7 | 27 ETB 17 23 | 47 LA7 ' 27 39 | 67 LA23 7 37 55 | 107 TA7 G 47 71 | 127 TA23 W 57 87 | 147 SA7 g 67 103 | 167 SA23 w 77 119 |
| 1 0 0 0 | 10 GET BS 8 8 | 30 SPE CAN 18 24 | 50 LA8 ( 28 40 | 70 LA24 8 38 56 | 110 TA8 H 48 72 | 130 TA24 X 58 88 | 150 SA8 h 68 104 | 170 SA24 x 78 120 |
| 1 0 0 1 | 11 TCT HT 9 9 | 31 SPD EM 19 25 | 51 LA9 ) 29 41 | 71 LA25 9 39 57 | 111 TA9 I 49 73 | 131 TA25 Y 59 89 | 151 SA9 i 69 105 | 171 SA25 y 79 121 |
| 1 0 1 0 | 12 LF A 10 | 32 SUB 1A 26 | 52 LA10 * 2A 42 | 72 LA26 : 3A 58 | 112 TA10 J 4A 74 | 132 TA26 Z 5A 90 | 152 SA10 j 6A 106 | 172 SA26 z 7A 122 |
| 1 0 1 1 | 13 VT B 11 | 33 ESC 1B 27 | 53 LA11 + 2B 43 | 73 LA27 ; 3B 59 | 113 TA11 K 4B 75 | 133 TA27 [ 5B 91 | 153 SA11 k 6B 107 | 173 SA27 { 7B 123 |
| 1 1 0 0 | 14 FF C 12 | 34 FS 1C 28 | 54 LA12 , 2C 44 | 74 LA28 < 3C 60 | 114 TA12 L 4C 76 | 134 TA28 \ 5C 92 | 154 SA12 l 6C 108 | 174 SA28 ¦ 7C 124 |
| 1 1 0 1 | 15 CR D 13 | 35 GS 1D 29 | 55 LA13 – 2D 45 | 75 LA29 = 3D 61 | 115 TA13 M 4D 77 | 135 TA29 ] 5D 93 | 155 SA13 m 6D 109 | 175 SA29 } 7D 125 |
| 1 1 1 0 | 16 SO E 14 | 36 RS 1E 30 | 56 LA14 . 2E 46 | 76 LA30 > 3E 62 | 116 TA14 N 4E 78 | 136 TA30 ^ 5E 94 | 156 SA14 n 6E 110 | 176 SA30 ~ 7E 126 |
| 1 1 1 1 | 17 SI F 15 | 37 US 1F 31 | 57 LA15 / 2F 47 | 77 UNL ? 3F 63 | 117 TA15 O 4F 79 | 137 UNT – 5F 95 | 157 SA15 o 6F 111 | 177 RUBOUT (DEL) 7F 127 |
| | ADDRESSED COMMANDS | UNIVERSAL COMMANDS | LISTEN ADDRESSES | | TALK ADDRESSES | | SECONDARY ADDRESSES OR COMMANDS | |

**KEY**




**Tektronix**
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

# Appendix B: Reserved Words

The following is a list of the reserved words of the digitizing oscilloscope. Do not use these words for aliases. The list starts on the next page

*CAL
*CLS
*DDT
*ESE
*ESR
*IDN
*IST
*LRN
*OPC
*PSC
*PUD
*RCL
*RST
*SAV
*SRE
*STB
*TRG
*TST
*WAI
ABOrt
ABSolute
AC
ACQuire
ACCUMDots
ACCUMVectors
ACQUISition
ACQuire
ALIas
ALL
ALLev
ALWAYS
AMPlitude
ANY
ANYFIELD
ASC
ASCII
AUTO
AUTOSet
AVErage
BANdwidth
BAUd
BIN
BIT_Nr
BMENU1
BMENU2
BMENU3
BMENU4
BMENU5
BMENU6
BMENU7
BMP
BN_Fmt
BRIght
BURst
BUSY
BYT_Nr
BYT_Or
CALC
CALibrate
CATALOG
CENtronics
CH1
CH1PROBE
CH2
CH2PROBE
CLEAR
CLEARMenu
CLEARSpool
CLOCk
CMEan
CONTRast
CONTROl
COPY
COUPling
CPU
CR
CRLf
CRMs
CROSSHair
CURSor
CURSOR1
CURSOR2
CURVe
CWD
DATE
DATa
DC
DCD
DEFINE
DELay
DELAYEd
DELEte
DELTa
DELWarn
DESE
DESKJet
DESTination
DEVelop
DIAg
DIM
DINk
DIR
DISplay
DOTs
DPU411
DPU412
ECL
EDGE
ENCdg
ENVelope
EPSImage
EPSon
ERASEFactory
ERRLOG
EVEN
EVEnt
EVMsg
EVQty
EXECute
EXT
EXT10
EXT10TRIGHIGH
EXT10TRIGLOW
EXTTRIGCALC
EXTTRIGHIGH
EXTTRIGLOW
FACtory
FAIL
FALL
FIELD
FIELD1
FIELD2
FILE
FILEFormat
FILEName
FILESystem
FIRST
FITtoscreen
FLAg
FORCe
FORMat
FPAnel
FRAme
FREESpace
FREQuency
FULl
FUNCtion
GATing
GND
GPIB
GPKNOB
GRAticule
GRId
HARDCopy
HARDFlagging
HBArs
HDELta
HDR
HEADer
HERtz
HFRej
HIGH
HIStogram
HOLdoff
HORZMENU
HORZPOS
HORZSCALE
HORizontal
HPOS1
HPOS2
IBFull
ID
IMMed
INDEpendent
INFInite
INIT
INITACQREF
INTENSIFied
INTENSITy
INTERLeaf
INTERNal
INVert
LANDscape
LASERJet
LAYout
LEVel
LF
LFCr
LFRej
LINE
LOCk
LOG
LOOP
LOW
LSB
MAIn
MATH
MATH1
MATHCad
MAXimum
MEAN
MEAS1
MEAS2
MEAS3
MEAS4
MEASMENU
MEASUrement
METHod
MID
MINImum
MINMax
MKDir
MODe
MSB
NAME
NDUTY
NEWpass
NEXT
NOISErej
NOISYGAIN
NONE
NORMal
NOVershoot
NR_Pt
NUMACq
NUMAVg
NUMERRors
NUMEnv
NWIdth
ODD
OFF
OFFSet
ON
ONCE
OPTion
OVERAll
OVERWrite
PACE
PAIRed
PARity
PASSWord
PCX
PDUTy
PEAKdetect
PERCent
PERIod
PERSistence
PITBULL
PK2pk
PORT
PORTRait
POSition
POSITION1
POSITION2
POVershoot
PRESet
PREss
PRInt
PRObe
PT_Fmt
PT_Off
PWIdth
RATE1
RATE2
RATE3
RATE4
RATE5
RAW
REBOOT
RECAll
RECOrdlength
REF1
REF1HIGH
REF1LOW
REF2
REF2HIGH
REF2LOW
REF3HIGH
REF3LOW
REF4HIGH
REF4LOW
REFLevel
REM
REName
REPET
RESUlt
RFR
RI
RIBinary
RISe
RMDir
RMENU1
RMENU2
RMENU3
RMENU4
RMENU5
RMS
RP
RPBinary
RS232
RTS
RUN
RUNSAfter
RUNSTop
SAMple
SAVe
SBITS
SCAle
SCAN
SECOnds
SECdiv
SELect
SEQuence
SET
SETLevel
SETUp
SHOWColzero
SHOWFirst
SLOpe
SNAp
SOFTFlagging
SOURCE
SOURCE1
SPREADSheet
SRIbinary
SRPbinary
SRSETUP
STARt
STATE
STATUS
STATUSMSG
STOP
STOPAfter
STOPBits
STYle
SYSTEM
TARget
TEKSecure
TERMinator
TEXt
THInkjet
TIFf
TIMe
TIMING
TOGGLE
TRANsmit
TRIGger
TRIGLEVEL
TRIGMENU
TRIGT
TTL
TURN
TWEnty
TYPe
UNIts
UNLOCk
UTILITY
VALue
VBArs
VDELta
VECtors
VERBose
VERTMENU
VERTPOS
VERTSCALE
VERTical
VIDeo
VOLts
WAVEFORM
WAVFrm
WFid
WFMOFF
WFMPre
WIDth
XINcr
XMULt
XOFf
XON
XUNit
XY
XZEro
Y
YMULt
YOFf
YT
YUNit
YZEro
ZMULt
ZOFf
ZOOm
ZUNit
ZZEro

# Appendix C: Interface Specifications

This appendix describes the oscilloscope GPIB remote interface. The information is useful when connecting to GPIB controllers of unusual configuration.

## GPIB Function Subsets

The oscilloscope supports many GPIB function subsets, as described in the list below. Some of the listings describe subsets that the oscilloscope does not support.

- SH1 (Source Handshake). The oscilloscope can transmit multiline messages across the GPIB.
- AH1 (Acceptor Handshake). The oscilloscope can receive multiline messages across the GPIB.
- T5 (Talker). The oscilloscope becomes a talker when the controller sends its talk address with the ATN (Attention) line asserted. It can send both response data and status information when responding to a serial poll. It ceases to be a talker when the controller sends another device's talk address with ATN asserted. The oscilloscope has talk-only capability for hard copy operation.
- L4 (Listener). The oscilloscope becomes a listener when the controller sends its listen address with the ATN (Attention) line asserted. The oscilloscope does not have listen-only capability.
- SR1 (Service Request). The oscilloscope asserts the SRQ (Service Request) line to notify the controller when it requires service.
- RL1 (Remote/Local). The oscilloscope responds to both the GTL (Go To Local) and LLO (Local Lock Out) interface messages.
- PP0 (Parallel Poll). The oscilloscope has no parallel poll capability. It does not respond to the following interface messages: PPC, PPD, PPE, and PPU. The oscilloscope does not send out a status message when the ATN (Attention) and EOI (End or Identify) lines are asserted simultaneously.
- DC1 (Device Clear). The oscilloscope responds to the DCL (Device Clear) and, when made a listener, the SDC (Selected Device Clear) interface messages.

- DT1 (Device Trigger). When acting as a listener, the oscilloscope responds to the GET (Group Execute Trigger) interface message.
- C0 (Controller). The oscilloscope cannot control other devices.
- E2 (Electrical). The oscilloscope uses tristate buffers to provide optimal high-speed data transfer.

## Interface Messages

Table C–1 shows the standard interface messages that the oscilloscope supports.

**Table C–1: Standard interface messages**

| Message | GPIB |
|---|---|
| DCL | Yes |
| GET | Yes |
| GTL | Yes |
| LLO | Yes |
| PPC | No |
| PPD | No |
| PPE | No |
| PPU | No |
| SDC | Yes |
| SPD | Yes |
| SPE | Yes |
| TCT | No |
| UNL | Yes |
| UNT | Yes |
| Listen Addresses | Yes |
| Talk Addresses | Yes |

# Appendix D: Factory Initialization Settings

The factory initialization settings provide a known state for the oscilloscope. Factory initialization sets values as shown in Table D–1.

**Table D–1: Factory initialization defaults**

| Control | Factory Initialization value |
|---|---|
| Acquire mode | Sample |
| Acquire stop after | RUN/STOP button only |
| Acquire # of averages | 16 |
| Acquire # of envelopes | 8 |
| Channel selection | Channel 1 on, all others off |
| Cursor H Bar 1 position | –3.2 divisions from the center |
| Cursor H Bar 2 position | +3.2 divisions from the center |
| Cursor V Bar 1 position | 30% of the record length (1000) |
| Cursor V Bar 2 position | 70% of the record length (1000) |
| Cursor function | Off |
| Cursor time units | Seconds |
| Delay time, delayed runs after main | 1 µs |
| Display Date/Time | On |
| Display format | YT |
| Display graticule type | Full |
| Display intensity – contrast | 150% |
| Display intensity – text | DIM |
| Display intensity – waveform | BRIGHT |
| Display intensity – overall | 85% |
| Display style | Vectors |
| Display trigger “T” | On |
| Display variable persistence | 500 ms |
| Edge trigger coupling | DC |
| Edge trigger level | 0.0 V |
| Edge trigger slope | Rising |
| Edge trigger source | Channel 1 |
| Horizontal – main trigger position | 50% |

**Table D–1: Factory initialization defaults (Cont.)**

| Control | Factory Initialization value |
|---|---|
| Horizontal – main time/div. | 500 µs |
| Horizontal – fit to screen | Off |
| Horizontal – delay time/div | 50 µs |
| Horizontal – time base | Main only |
| Main trigger holdoff | 500 ns |
| Main trigger mode | Auto |
| Main trigger type | Edge |
| Math waveform function | CH1 + CH2 |
| Measure 1–4 | Off |
| Measure Gating | Off |
| Measure High-Low Setup | Histogram |
| Measure High Ref | 90% and 0 V (units) |
| Measure Low Ref | 10% and 0 V (units) |
| Measure Mid Ref | 50% and 0 V (units) |
| Ref ufm lock | Locked |
| Saved setups | No change |
| Saved waveforms | No change |
| Vertical bandwidth (both channels) | Full |
| Vertical coupling (both channels) | DC |
| Vertical offset (both channels) | 0 V |
| Vertical position (both channels) | 0 div |
| Vertical volts/div. (both channels) | 100 mV/div |
| Video trigger on | Field1 |
| Video trigger scan rate | 15 – 20kHz |

## Setup and Waveform File Formats

TDS 300 Series setup and waveform files are not compatible with TDS 400, TDS 500, TDS 600, TDS 700, or TDS 800 Series setup or waveform files. The TDS 300 Series oscilloscopes write information in Tek Codes and Formats (ASCII) structure, while the other TDS-Series oscilloscopes write information in Smalltalk data structures.

# Glossary

**ASCII**
Acronym for the American Standard Code for Information Interchange. Controllers transmit commands to the digitizing oscilloscope using ASCII character encoding.

**Address**
A 7-bit code that identifies an instrument on the communication bus. The digitizing oscilloscope must have a unique address for the controller to recognize and transmit commands to it.

**Backus-Naur Form (BNF)**
A standard notation system for command syntax diagrams. The syntax diagrams in this manual use BNF notation.

**Controller**
A computer or other device that sends commands to and accepts responses from the digitizing oscilloscope.

**EOI**
A mnemonic referring to the control line "End or Identify" on the GPIB interface bus. One of the two possible end-of-message terminators.

**EOM**
A generic acronym referring to the end-of-message terminator. The end-of-message terminator is either an EOI or the ASCII code for line feed (LF).

**GPIB**
Acronym for General Purpose Interface Bus, the common name for the communications interface system defined in IEEE Std 488.

**IEEE**
Acronym for the Institute for Electrical and Electronic Engineers.

**QuickBASIC**
A computer language (distributed by Microsoft) that is based on the Beginner's All-Purpose Symbolic Instruction Code.

**QuickC**
A computer language (distributed by Microsoft) that is based on C.

**RS-232**
A serial, full-duplex, asynchronous communication port that follows ANSI/EIA/TIA–562–1989[1], ANSI/EIA/TIA–574–1990[2], and CCITT V.24–1989[3] standards.

**TEKSecure**
A Tektronix custom command that initializes both waveform and setup memories. This overwrites any previously stored data.

# Index

## A



## B



## C

## D



## E

## F



## G



## H



## I



## L



## M

## N



## O



## P



## Q

## R



## S

## T



## U



## V



## W

## Z