Programmer Manual

Tektronix

TDS 310, TDS 320 & TDS 350
Two Channel Oscilloscopes

070-8571-04

Use this manual for TDS 310, TDS 320, and
TDS 350 oscilloscopes with firmware
versions 2.04 and above.

Fifth Edition: December 1994
WARRANTY

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of three (3) years from the date of shipment. If any such 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; or c) 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 WITH RESPECT TO THIS PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESSED 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

Safety Summary ................................................................. vii
Preface ................................................................................. ix

Getting Started

Manual Structure ................................................................. 1-1
Installing The Option 14 Module .......................................... 1-3
Choosing an Interface .......................................................... 1-3
Setting Up Remote Communications ..................................... 1-4

Syntax and Commands

Command Syntax ................................................................. 2-1
Command and Query Structure ............................................. 2-1
Clearing the Oscilloscope .................................................... 2-4
Command Entry .................................................................... 2-4
Constructed Mnemonics ...................................................... 2-6
Argument Types .................................................................... 2-7
Syntax Diagrams ..................................................................... 2-10
Command Groups .................................................................. 2-11
Acquisition Commands ......................................................... 2-11
Alias Commands .................................................................... 2-11
Calibration and Diagnostic Commands .................................. 2-12
Cursor Commands .................................................................. 2-12
Display Commands .................................................................. 2-13
Hard Copy Commands ............................................................ 2-14
Horizontal Commands ............................................................ 2-14
Measurement Commands ....................................................... 2-15
Miscellaneous Commands ..................................................... 2-17
Save and Recall Commands .................................................. 2-18
Status and Error Commands .................................................. 2-19
Trigger Commands ................................................................. 2-19
Vertical Commands ............................................................... 2-20
Waveform Commands ............................................................ 2-22
Command Descriptions .......................................................... 2-27
List of Figures

Figure 1-1: The Command Syntax Section Describes Common Message Elements ................................................. 1-1
Figure 1-2: The Commands Section Lists and Explains Commands ................................................................. 1-2
Figure 1-3: GPIB Service Requests (SRQs) Provide for Event (Interrupt) Driven Programs ............................. 1-2
Figure 1-4: The Disks That Accompany This Manual ................................................................. 1-3
Figure 1-5: GPIB, RS-232, and Centronics Connector Locations ............................................................... 1-4
Figure 1-6: How to Stack GPIB Connectors ......................................................................................................... 1-5
Figure 1-7: Typical GPIB Network Configurations ............................................................................................ 1-6
Figure 1-8: Selecting the I/O System in the Main Menu ............................................................................ 1-6
Figure 1-9: Selecting the GPIB Address in the GPIB Configuration Side Menu ........................................... 1-7
Figure 1-10: Pin Assignments of the RS-232 Connector ......................................................................................... 1-8
Figure 1-11: RS-232 Parameter Settings ........................................................................................................ 1-10
Figure 1-12: RS-232 Hardcopy Menu .......................................................................................................... 1-11
Figure 2-1: Command Message Elements ..................................................................................................... 2-2
Figure 2-2: Block Argument Example ............................................................................................................ 2-9
Figure 2-3: Typical Syntax Diagrams ............................................................................................................... 2-10
Figure 3-1: The Standard Event Status Register (SESR) ................................................................................ 3-1
Figure 3-2: The Status Byte Register (SBR) ................................................................................................. 3-2
Figure 3-3: The Device Event Status Enable Register (DESER) ................................................................. 3-3
Figure 3-4: The Event Status Enable Register (ESER) .................................................................................. 3-4
Figure 3-5: The Service Request Enable Register (SRER) ........................................................................ 3-4
Figure 3-6: Status and Event Handling Process ............................................................................................ 3-6
Figure 3-7: Command Processing Without Using Synchronization .............................................................. 3-7
Figure 3-8: Processing Sequence With Synchronization ............................................................................. 3-8
Figure 4-1: Equipment Needed to Run the GPIB and RS-232 Example Programs ........................................... 4-1
List of Tables

Table 1-1: GPIB and RS-232 Comparison .......................... 1-3
Table 1-2: RS-232 Cables ........................................... 1-9
Table 1-3: RS-232 Troubleshooting ............................... 1-13
Table 2-1: BNF Symbols and Meanings ............................. 2-1
Table 2-2: Command Message Elements ........................... 2-2
Table 2-3: Comparison of Header Off and On Responses ........ 2-3
Table 2-4: Acquisition Commands ................................. 2-11
Table 2-5: Alias Commands .......................................... 2-11
Table 2-6: Calibration and Diagnostic Commands ................ 2-12
Table 2-7: Cursor Commands ........................................ 2-12
Table 2-8: Display Commands ....................................... 2-13
Table 2-9: Hard Copy Commands ................................. 2-14
Table 2-10: Horizontal Commands ................................. 2-14
Table 2-11: Measurement Commands ............................... 2-16
Table 2-12: Miscellaneous Commands ............................. 2-17
Table 2-13: Save and Recall Commands ......................... 2-18
Table 2-14: Status and Error Commands ......................... 2-19
Table 2-15: Trigger Commands ..................................... 2-20
Table 2-16: Vertical Commands .................................... 2-20
Table 2-17: Waveform Commands ................................. 2-24
Table 2-18: Commands that Affect BUSY? Response ............ 2-41
Table 2-19: Offset Ranges (All Channels) ....................... 2-46
Table 2-20: DATa and WFMPre Parameter Settings ............ 2-65
Table 2-21: XY Format Pairs ...................................... 2-76
Table 2-22: Horizontal Delay Time Resolution ................. 2-98
Table 2-23: Commands that Generate an Operation Complete Message .......................... 2-129
Table 2-24: Additional WFMPre Commands ....................... 2-173
Table 3-1: SESR Bit Functions .................................... 3-2
Table 3-2: SBR Bit Functions ..................................... 3-3
Table 3-3: No Event Messages ..................................... 3-13
Table 3-4: Command Error Messages — CME Bit 5 ............. 3-13
Table 3-5: Execution Error Messages — EXE Bit 4 ............. 3-14
Table 3-6: Device Error Messages — DDE Bit 3 ................. 3-16
Table 3-7: System Event Messages ............................... 3-17
Table 3-8: Execution Warning Messages — EXE Bit 4 ........... 3-17
Table 3-9: Internal Warning Messages ........................... 3-18
List of Tables

Table A-1: The TDS 300 Series Character Set ........................................ A-1
Table A-2: The ASCII & GPIB Code Chart ............................................. A-2
Table A-3: Standard Interface Messages ............................................... A-6
Table A-4: Factory Initialization Defaults ............................................. A-7
Safety Summary

Please take a moment to review these safety precautions. They are provided for your protection and to prevent damage to the oscilloscope.

Symbols and Terms

These two terms appear in manuals:

- **CAUTION** statements identify conditions or practices that could result in damage to the equipment or other property.

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

These two terms appear on equipment:

- **CAUTION** indicates a personal injury hazard not immediately accessible as one reads the marking, or a hazard to property including the equipment itself.

- **DANGER** indicates a personal injury hazard immediately accessible as one reads the marking.

This symbol appears in manuals:

![Static-Sensitive Devices](image)

These symbols appear on equipment:

- **DANGER** High Voltage

- **Protective ground (earth) terminal**

- **ATTENTION** Refer to manual
Specific Precautions

Observe all the following precautions to ensure your personal safety and to prevent damage to either the oscilloscope or equipment connected to it.

Power Source

The oscilloscope operates from a power source that will not apply more than $250 \, \text{V}_{\text{RMS}}$ between the supply conductors or between either supply conductor and ground. A protective ground connection, through the grounding conductor in the power cord, is essential for safe system operation.

Grounding the Oscilloscope

The oscilloscope is grounded through the power cord. To avoid electric shock, plug the power cord into a properly wired receptacle with an earth ground connection. Do this before making connections to the input or output terminals of the oscilloscope.

Without the protective ground connection, all parts of the oscilloscope are potential shock hazards. This includes knobs and controls that may appear to be insulators.

Use the Proper Power Cord

Use only the power cord and connector specified for your product. Use only a power cord that is in good condition.

Use the Proper Fuse

To avoid fire hazard, use only a fuse that meets all type, voltage, and current specifications.

Do Not Remove Covers or Panels

To avoid personal injury, do not operate the oscilloscope without the panels or covers.

Do Not Operate in Explosive Atmospheres

The oscilloscope provides no explosion protection from static discharges or arcing components. Do not operate the oscilloscope in an atmosphere of explosive gasses.

Electric Overload

Never apply a voltage to a connector on the oscilloscope that is outside the range specified for that connector.
This is the Programmer Manual for the TDS 310, TDS 320, and TDS 350 Two Channel 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 310, TDS 320 & TDS 350 Instruction Manual*
- *TDS 310, TDS 320 & TDS 350 Reference*
- *The XYZs of Analog and Digital Oscilloscopes*
Preface
Getting Started

This section covers the following topics.

- **Manual Structure** describes the major sections in this manual.
- **Installing the Option 14 Module** describes how to install the communications interface.
- **Choosing an Interface** describes the advantages and disadvantages of GPIB and RS-232 interfaces.
- **Setting Up Remote Communications** describes setting up for remote control including connecting the oscilloscope and setting the appropriate front panel controls.

---

**Manual Structure**

This manual includes the following sections.

**Syntax and Commands**

The Syntax and Commands section (Section 2) describes the structure of the messages your program sends to the oscilloscope. Figure 1-1 shows a syntax diagram and command parts as described in this section.

![Syntax Diagram](image)

**Figure 1-1: The Command Syntax Section Describes Common Message Elements**

The Syntax and Commands section also describes each command used in the oscilloscope. It explains the syntax of each command and provides examples of how you might use it. The section arranges commands alphabetically and also provides a list by functional area (Figure 1-2).
Figure 1-2: The Commands Section Lists and Explains Commands

Status and Events

The Status and Events section (Section 4) starting on page 3-1 describes how to use GPIB service requests (SRQs) and various event messages in your programs.

The program requests information from the oscilloscope. The oscilloscope provides information in the form of status and error messages. Figure 1-3 illustrates the basic operation of this system.

Figure 1-3: GPIB Service Requests (SRQs) Provide for Event (Interrupt) Driven Programs

Examples

The Programming Examples section starting on page 4-1 describes example oscilloscope control programs and how to compile them. The disks that come with this manual (Figure 1-4) have an executable and a source code version of each program.
Installing The Option 14 Module

If you received your Option 14 Communications Interface in a separate shipment, you must install it before remote communications is possible.

Use the procedure in your TDS 300 Series TD3F14A I/O Interfaces Upgrade Kit to install your Option 14 module.

Choosing an Interface

Your system hardware may let you choose which interface to use with your system; if so, you should consider the comparative advantages and disadvantages of each interface. For example, the GPIB interface is an eight bit parallel bus and, therefore, it offers high-speed data transfers and multiple instrument control. In contrast, the RS-232 interface is a slower serial data bus for single instrument control, but it is easy to connect to and can be used with a low-cost controller. Table 1-1 compares the GPIB and RS-232 interface.

Table 1-1: GPIB and RS-232 Comparison

<table>
<thead>
<tr>
<th>Operating Attribute</th>
<th>GPIB</th>
<th>RS-232</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cable</td>
<td>IEEE-488 Std.</td>
<td>9-wire</td>
</tr>
<tr>
<td>Data flow control</td>
<td>Hardware, 3-wire handshake</td>
<td>Flagging: soft (XON/XOFF), hard (RTS/CTS)</td>
</tr>
<tr>
<td>Data format</td>
<td>8-bit parallel</td>
<td>8-bit serial</td>
</tr>
<tr>
<td>Interface control</td>
<td>Operator low-level control message</td>
<td>None</td>
</tr>
<tr>
<td>Interface messages</td>
<td>Most IEEE-488 Std.</td>
<td>Device clear via break signal</td>
</tr>
<tr>
<td>Interrupts reported</td>
<td>Service requests status and event code</td>
<td>None. Must be polled for status.</td>
</tr>
<tr>
<td>Message termination (Receive)</td>
<td>Hardware EOI, software LF, or both</td>
<td>Software CR, LF, or CRLF</td>
</tr>
</tbody>
</table>
Table 1-1: GPIB and RS-232 Comparison (Cont.)

<table>
<thead>
<tr>
<th>Operating Attribute</th>
<th>GPIB</th>
<th>RS-232</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message termination</td>
<td>Hardware EOI, and software LF</td>
<td>Software CR, LF, CRLF, or LFCR</td>
</tr>
<tr>
<td>(Transmit)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timing</td>
<td>Asynchronous</td>
<td>Asynchronous</td>
</tr>
<tr>
<td>Transmission path length</td>
<td>≤ 2 meters between devices; ≤ 20 meters total cabling for GPIB system</td>
<td>≤15 meters</td>
</tr>
<tr>
<td>Speed</td>
<td>200 kBytes/sec</td>
<td>19,200 bits/sec</td>
</tr>
<tr>
<td>System environment</td>
<td>Multiple devices (≤ 15)</td>
<td>Single terminal (point to point connection)</td>
</tr>
</tbody>
</table>

Setting Up Remote Communications

The oscilloscope has GPIB, RS-232, and Centronics connectors on its rear panel, as shown in Figure 1-5. These connectors have D-type shells. The GPIB connector conforms to IEEE Std. 488.1 – 1987. The RS-232 connector conforms to ANSI/EIA/TIA Standard 574 – 1990.

Connecting to a GPIB Device

Attach an IEEE Std. 488.1 – 1987 GPIB cable to the GPIB connector.

![Figure 1-5: GPIB, RS-232, and Centronics Connector Locations](image)

If needed, you can stack GPIB connectors as shown in Figure 1-6.

**GPIB Requirements** — Observe these rules when you use your oscilloscope with a GPIB network:
Getting Started

- Assign each device on the bus a unique device address; 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 6 feet (2 meters) of cable used.
- Do not use more than 65 feet (20 meters) of cable to connect devices to a bus.
- At least two-thirds of the devices on the network must be turned on while the network is operating.
- Connect the devices on the network in a star or linear configuration as shown in Figure 1-7. Do not use loop or parallel configurations.

Appendix C: Interface Specifications, gives additional information on the GPIB configuration of the oscilloscope.

Figure 1-6: How to Stack GPIB Connectors
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 **Hcp Port** button in the main menu; then, if you want hardcopies to be sent by way of the GPIB port, press **GPIB** in the side menu.

4. Press the **GPIB** button in the main menu to display the GPIB configuration side menu.
5. Press the Talk/Listen Address side menu button, and set the GPIB address using the general purpose knob.

The oscilloscope is set up for bidirectional communication with your controller or computer. If you wish to isolate the oscilloscope from the bus: press the Off Bus side menu button.

If you wish to enter a special mode of operation to communicate directly with non-488.2 hard copy devices press the Hardcopy side menu button. The oscilloscope sends hard copy information only when you press the HARD-COPY button.

Figure 1-9: Selecting the GPIB Address in the GPIB Configuration Side Menu

Connecting to an RS-232 Device

The RS-232 interface provides a point-to-point connection between two items of equipment such as a computer or terminal and the oscilloscope. This section tells how to connect and set up the oscilloscope for communication over the RS-232 interface.

RS-232 Interface — defines two types of devices: Data Terminal Equipment (DTE) and Data Communications Equipment (DCE).
**NOTE**

Some DTE devices may have female connectors. Also, the RS-232 ports of personal computers may be configured as DCE or DTE devices, with either a 25-pin or a 9-pin connector. Refer to the documentation that accompanies your computer or terminal to determine if it is a DTE or a DCE device.

The oscilloscope is a DTE device. A 9-pin D-type shell RS-232 connector is located on the rear panel. In standard usage, a male connector appears on DTE devices, and a female connector appears on DCE devices. A straight-through female-to-male cable of less than 50 feet is typically used for local DTE-to-DCE connection. Figure 1-10 on page 1-8 shows the oscilloscope 9-pin connector with its pin number assignments. When connecting the oscilloscope to another RS-232 device consider these suggestions:

- Many devices require a constant high signal on one or more input pins
- Do not connect the output line of one DTE device to the output line of the other.
- Ensure that the signal ground of the oscilloscope is connected to the signal ground of the external device
- Ensure that the chassis ground of the oscilloscope is connected to the chassis ground of the external device

**9-PIN D-SHELL**

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 1-10: Pin Assignments of the RS-232 Connector*

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 Cables

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

### Setting RS-232 Parameters

To set the RS-232 parameters, do the following steps:

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. Then press the **RS-232** main menu button to display the RS-232C side menu (see Figure 1-11). You may set the following parameters:

   - **Baud Rate** — sets the data transmission rate. You can set rates of 300, 600, 1200, 2400, 4800, 9600, or 19200 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. Use hard flagging for binary data transfers.
   - **Set RS-232 Parameters to Default Values** — sets default values for RS-232 parameters (for a list of default settings see page 2-139).
   - **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 2-142).
   - **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.
Getting Started

- **Delay** — sets the delay time before responding to a query. You can set times from 0 to 60 seconds.

3. 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.

4. Press the **Hcp Port** main menu button to display the Hard Copy Port side menu. Then, if you want hardcopies to be sent via the RS-232 port, press the **RS-232C** side menu button to select the RS-232 port as a remote interface port (see Figure 1-12).

After these parameters are set, the RS-232 interface is ready to operate.

![Figure 1-11: RS-232 Parameter Settings](image)

---

**Figure 1-11**: RS-232 Parameter Settings
RS-232 Conventions

You should be aware of the processing conventions that are specific to the RS-232 interface. These conventions pertain to:

- Transferring binary data
- Processing break signals
- RS-232 I/O errors
- Service Requests
- Hardcopies

**When Transferring Binary Data** — to the oscilloscope via the RS-232 port, note the following:

- Do not use binary data transfers with soft flagging unless you can ensure that the data does not contain XON or XOFF characters. Using RTS/CTS (hard) flagging guarantees correct data transfer.
- All eight bits of binary data contain meaningful information. To ensure that all eight bits are received or transmitted, an RS-232 device which is connected to the oscilloscope must be configured to receive and transmit eight-bit characters (set the RS-232 word length to eight bits).
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 behaves as if a GPIB <DCL> command was received (input and output buffers are flushed, and the oscilloscope waits 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 is sent in the middle of a character stream, a couple of characters immediately preceding or following the BREAK may be lost. The controller should wait until the DCL and end of line terminator string is received before sending more characters.

RS-232 I/O 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 (see 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 UTILITY Diag Error Log.

You can use the RS232 Line Snapshot entry of the error log to help establish an RS-232 connection. The snapshot reports if the oscilloscope is waiting to receive a control Q (yes/no), the state of the hardware CTS line (high/low), and if characters have been received (yes/no). If Waiting For ^Q is Yes, the oscilloscope must receive an XON character before it will transmit any more data. If CTS is Low and hard flagging is enabled, the oscilloscope will not transmit any data. If hard flagging is not enabled, 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.

Service Requests — are part of the GPIB interface. There is no counterpart for RS-232. However, 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.

Hardcopies — The Centronics, GPIB, and RS-232 ports are always active. However, only one port at a time can be used for hardcopies. The hardcopy port is selected in the UTILITY I/O menu.

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 and that it 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. Verify that the cable is wired correctly. Try your program or printer again.
2. Verify that the settings on both pages of the UTILITY I/O RS-232C menu match the settings used by your printer or the program on your personal computer. Start by pressing **Set RS-232 Parameters to Default Values** in the RS-232C menu. Then, change only those menu items that you know need to be changed, like perhaps the baud rate. You should not need to change the settings on the second page of the menu because they are standard on most printers and personal computers. Try your printer or computer program again.

3. If you are trying to generate a hardcopy using the RS-232 port, verify that the UTILITY I/O Hcp Port is set to RS-232C.

4. If you are trying to control the oscilloscope using a personal computer or other computer, look at the UTILITY Diag Error Log and examine the RS232 Line Snapshot and the RS232 Errors. Use Table 1-3 to troubleshoot your setup. 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.

### Table 1-3: RS-232 Troubleshooting

<table>
<thead>
<tr>
<th>Symptom</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>Your personal computer program tried to send characters to the oscilloscope, but the oscilloscope error log displays Chars Rcvd: No</td>
<td>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.</td>
</tr>
<tr>
<td>The oscilloscope shows a non-zero number of Framing errors.</td>
<td>There is a baud rate mismatch between the oscilloscope and the personal computer. There is an incorrect number of bits sent by the personal computer (the oscilloscope expects 8 data bits). The personal computer may be sending a parity bit when the oscilloscope is expecting no parity.</td>
</tr>
<tr>
<td>The oscilloscope shows a non-zero number of Parity errors.</td>
<td>There is a parity mismatch between the oscilloscope and the personal computer.</td>
</tr>
<tr>
<td>The oscilloscope shows a non-zero number of Overrun errors.</td>
<td>Flagging is not being used by the oscilloscope or the personal computer, or they are using different types of flagging. Ideally, hard flagging should be used unless you are using a telephone modem.</td>
</tr>
<tr>
<td>Transmissions are incomplete, or the oscilloscope does not process all commands from the personal computer.</td>
<td>Flagging is not being used by the oscilloscope or the personal computer, or they are using different types of flagging. Also, the end of line terminator may not match what the personal computer program expects.</td>
</tr>
<tr>
<td>The oscilloscope error log displays Waiting for ^Q: Yes.</td>
<td>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.</td>
</tr>
<tr>
<td>Soft flagging is being used, and transmissions stop.</td>
<td>Verify that both the personal computer and the oscilloscope agree that soft flagging is being used. Also, 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.</td>
</tr>
</tbody>
</table>
### Table 1-3: RS-232 Troubleshooting (Cont.)

<table>
<thead>
<tr>
<th>Symptom</th>
<th>Possible Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>The oscilloscope error log displays CTS: Low, and the oscilloscope is using hard flagging.</td>
<td>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.</td>
</tr>
<tr>
<td>After the personal computer program sends a BREAK, the first message fails.</td>
<td>Verify that the personal computer program is waiting for and reading the DCL and end of line terminator response sent by the oscilloscope.</td>
</tr>
</tbody>
</table>
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 on page A-1 contains a chart of the ASCII character set.

This manual uses Backus-Naur Form (BNF) notation and syntax diagrams to describe commands and queries.

This manual uses the following BNF symbols.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; &gt;</td>
<td>Defined element</td>
</tr>
<tr>
<td>::=</td>
<td>Is defined as</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>( )</td>
<td>Group; one element is required</td>
</tr>
<tr>
<td>[ ]</td>
<td>Optional; can be omitted</td>
</tr>
<tr>
<td>. . .</td>
<td>Previous element(s) may be repeated</td>
</tr>
<tr>
<td>( )</td>
<td>Comment</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Header&gt;</td>
<td>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 (*).</td>
</tr>
<tr>
<td>&lt;Mnemonic&gt;</td>
<td>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.</td>
</tr>
<tr>
<td>&lt;Argument&gt;</td>
<td>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 &lt;Space&gt;. Arguments are separated from each other by a &lt;Comma&gt;.</td>
</tr>
<tr>
<td>&lt;Comma&gt;</td>
<td>A single comma between arguments of multiple-argument commands. It may optionally have white space characters before and after the comma.</td>
</tr>
<tr>
<td>&lt;Space&gt;</td>
<td>A white space character between command header and argument. It may optionally consist of multiple white space characters.</td>
</tr>
</tbody>
</table>

**Figure 2-1: Command Message Elements**

```
SAVE:WAVEform CH1,REF2
```

```
SAVE <Space> WAVEform <Comma> <Space> <String> <Comma> REF <Comma> <x>
```
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>?]`  
- `[:<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>
<thead>
<tr>
<th>Query</th>
<th>Header Off Response</th>
<th>Header On Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURSOR:VBARs:DELTA?</td>
<td>1.064E-03</td>
<td>CURSOR:VBARs:DELTA 1.064E-03</td>
</tr>
<tr>
<td>ACQUIRE:NUMAVG?</td>
<td>16</td>
<td>ACQUIRE:NUMAVG 16</td>
</tr>
</tbody>
</table>
Clearing the Oscilloscope

To clear the Output Queue and reset the oscilloscope to accept a new command or query, 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 through 09 and 0B through 20 hexadecimal (0 through 9 and 11 through 32 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 so this is legal:

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 number of acquisitions for averaging, and the 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** — may 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 — may 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

#### Cursor Position Mnemonics

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>POSITION&lt;x&gt;</td>
<td>A cursor selector; &lt;x&gt; is either 1 or 2.</td>
</tr>
</tbody>
</table>

#### 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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEAS&lt;x&gt;</td>
<td>A measurement specifier; &lt;x&gt; is either 1, 2, 3, or 4.</td>
</tr>
</tbody>
</table>

#### Channel Mnemonics

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH&lt;x&gt;</td>
<td>A channel specifier; &lt;x&gt; is either 1 or 2.</td>
</tr>
</tbody>
</table>

#### Math Waveform Mnemonics

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATH&lt;x&gt;</td>
<td>A math waveform specifier; &lt;x&gt; is 1.</td>
</tr>
</tbody>
</table>
Reference Waveform Mnemonics

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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>REF&lt;x&gt;</td>
<td>A reference waveform specifier; &lt;x&gt; is either 1 or 2.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;wfm&gt;</td>
<td>Can be CH&lt;x&gt;, MATH, MATH1, or REF&lt;x&gt;</td>
</tr>
</tbody>
</table>

Argument Types

The argument of a command may be in one of several forms. The individual descriptions of each command tell which 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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;NR1&gt;</td>
<td>Signed integer value</td>
</tr>
<tr>
<td>&lt;NR2&gt;</td>
<td>Floating point value without an exponent</td>
</tr>
</tbody>
</table>

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"
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 on page A-1).

- 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:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;NZDig&gt;</td>
<td>A non-zero digit character, in the range 1–9</td>
</tr>
<tr>
<td>&lt;Dig&gt;</td>
<td>A digit character, in the range 0–9</td>
</tr>
<tr>
<td>&lt;DChar&gt;</td>
<td>A character with the hex equivalent of 00 through FF hexadecimal (0 through 255 decimal)</td>
</tr>
<tr>
<td>&lt;Block&gt;</td>
<td>A block of data bytes, defined as:</td>
</tr>
<tr>
<td></td>
<td>&lt;Block&gt; ::=</td>
</tr>
<tr>
<td></td>
<td>{ #&lt;NZDig&gt;&lt;Dig&gt;[&lt;Dig&gt;...][&lt;DChar&gt;...]</td>
</tr>
</tbody>
</table>

<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.

Aliasing: DEFINE "SETUp1", #231AUTOSet EXECute; :SELect:REF1

Figure 2-2: Block Argument Example
The syntax diagrams in this manual use the following symbols and notation:

- Circles and ovals contain literal elements. You must send most elements exactly as shown. The diagrams show command mnemonics in both upper and lower case to distinguish between complete and abbreviated spellings. These elements are not case sensitive and you can omit the lower case portion of the mnemonic.

- Boxes contain the defined elements described earlier in this section, such as <NR3> or <QString>.

- Elements are connected by arrows that show the allowed paths through the diagram, and thus the orders in which you can send the elements. Parallel paths show that you must take one and only one of the paths. A path around a group of elements shows that those elements are optional. Loops show elements that you can repeat.

Figure 2-3 shows the structure of a few typical syntax diagrams.

![Syntax Diagrams](image-url)
Command Groups

This section lists commands by functional groups. The following section, Command Descriptions, lists commands alphabetically. The alphabetical list starts on page 2-27.

The oscilloscope GPIB and RS232 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 control mode, averaging, enveloping, and single-waveform acquisition.

<table>
<thead>
<tr>
<th>Table 2-4: Acquisition Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
</tr>
<tr>
<td>ACQuire?</td>
</tr>
<tr>
<td>ACQuire:MODe</td>
</tr>
<tr>
<td>ACQuire:NUMACq?</td>
</tr>
<tr>
<td>ACQuire:NUMAVg</td>
</tr>
<tr>
<td>ACQuire:NUMEnv</td>
</tr>
<tr>
<td>ACQuire:STATE</td>
</tr>
<tr>
<td>ACQuire:STOPAAfter</td>
</tr>
</tbody>
</table>

 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>
<thead>
<tr>
<th>Table 2-5: Alias Commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
</tr>
<tr>
<td>ALIas</td>
</tr>
<tr>
<td>ALIas:CATalog?</td>
</tr>
<tr>
<td>ALIas:DEFINE</td>
</tr>
<tr>
<td>ALIas:DELETE</td>
</tr>
</tbody>
</table>
**Table 2-5: Alias Commands (Cont.)**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALIas:DELEte:ALL</td>
<td>Remove all aliases</td>
</tr>
<tr>
<td>ALIas:DELEte:NAME</td>
<td>Remove a named alias</td>
</tr>
<tr>
<td>ALIas:STATE</td>
<td>Turn the alias state on and off</td>
</tr>
</tbody>
</table>

**Calibration and Diagnostic Commands**

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

**Table 2-6: Calibration and Diagnostic Commands**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*CAL?</td>
<td>Perform an internal self-calibration</td>
</tr>
<tr>
<td>DIAg:RESULT:FLAG?</td>
<td>Return diagnostic tests status</td>
</tr>
<tr>
<td>DIAg:RESULT:LOG?</td>
<td>Return diagnostic test sequence results</td>
</tr>
<tr>
<td>DIAg:SELect:ALL</td>
<td>Diagnostic test sequence for Acquisition, Processor, Display, and Front panel</td>
</tr>
<tr>
<td>DIAg:STATE</td>
<td>Control of diagnostic tests</td>
</tr>
</tbody>
</table>

**Cursor Commands**

Cursor commands provide control over cursor display and readout.

**Table 2-7: Cursor Commands**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURSor?</td>
<td>Returns cursor settings</td>
</tr>
<tr>
<td>CURSor:FUNCTION</td>
<td>Cursors on or off; select cursor type</td>
</tr>
<tr>
<td>CURSor:HBArs?</td>
<td>Return H bar settings</td>
</tr>
<tr>
<td>CURSor:HBArs:DELTa?</td>
<td>Return distance between H bars</td>
</tr>
<tr>
<td>CURSor:HBArs:POSITION&lt;x&gt;</td>
<td>Position a horizontal cursor</td>
</tr>
<tr>
<td>CURSor:HBArs:SELect</td>
<td>Set which cursor the knob controls</td>
</tr>
</tbody>
</table>
### Table 2-7: Cursor Commands (Cont.)

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURSor:PAIred:HDELTA?</td>
<td>Query horizontal distance between first and second paired cursors</td>
</tr>
<tr>
<td>CURSor:PAIred:HPOS1?</td>
<td>Query horizontal position of first paired cursor</td>
</tr>
<tr>
<td>CURSor:PAIred:HPOS2?</td>
<td>Query horizontal position of second paired cursor</td>
</tr>
<tr>
<td>CURSor:PAIred:POSITION1?</td>
<td>Set vbar position of first paired cursor</td>
</tr>
<tr>
<td>CURSor:PAIred:POSITION2?</td>
<td>Set vbar position of the second paired cursor</td>
</tr>
<tr>
<td>CURSor:PAIred:SESelect?</td>
<td>Select active paired cursor</td>
</tr>
<tr>
<td>CURSor:PAIred:VDELTA?</td>
<td>Query vertical distance between first and second paired cursors</td>
</tr>
<tr>
<td>CURSor:VBArs</td>
<td>Position vertical bar cursors</td>
</tr>
<tr>
<td>CURSor:VBArs:DELTA?</td>
<td>Horizontal distance between cursors</td>
</tr>
<tr>
<td>CURSor:VBArs:POSITION&lt;x&gt;</td>
<td>Position a vertical cursor</td>
</tr>
<tr>
<td>CURSor:VBArs:SESelect</td>
<td>Set which cursor the knob controls</td>
</tr>
<tr>
<td>CURSor:VBArs:UNIts</td>
<td>Set vertical cursors to period or frequency</td>
</tr>
</tbody>
</table>

### Display Commands

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

### Table 2-8: Display Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEARMenu</td>
<td>Clear menus from display</td>
</tr>
<tr>
<td>DISplay?</td>
<td>Returns display settings</td>
</tr>
<tr>
<td>DISplay:FORMat</td>
<td>YT or XY display</td>
</tr>
<tr>
<td>DISplay:GRAticule</td>
<td>Graticule style</td>
</tr>
<tr>
<td>DISplay:INTENSITy?</td>
<td>Returns intensity settings</td>
</tr>
<tr>
<td>DISplay:INTENSITy:CONTRast</td>
<td>Waveform intensified zone brightness</td>
</tr>
<tr>
<td>DISplay:INTENSITy:OVERALL</td>
<td>Main brightness</td>
</tr>
</tbody>
</table>
Table 2-8: Display Commands (Cont.)

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISplay:INTENSITY:TEXt</td>
<td>Text brightness</td>
</tr>
<tr>
<td>DISplay:INTENSITY:WAVEform</td>
<td>Waveform brightness</td>
</tr>
<tr>
<td>DISplay:STYle</td>
<td>Waveform dots, vectors, dot accumulate or vector accumulate</td>
</tr>
<tr>
<td>DISplay:TRIGT</td>
<td>Controls the display of the trigger indicator on screen</td>
</tr>
</tbody>
</table>

### 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 2-9: Hard Copy Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARDCopy</td>
<td>Start or terminate hard copy</td>
</tr>
<tr>
<td>HARDCopy:CLEARSpool</td>
<td>Empty hard copy spooler</td>
</tr>
<tr>
<td>HARDCopy:FORMAT</td>
<td>Hard copy output format</td>
</tr>
<tr>
<td>HARDCopy:LAYout</td>
<td>Hard copy orientation</td>
</tr>
<tr>
<td>HARDCopy:PORT</td>
<td>Hard copy port for output</td>
</tr>
</tbody>
</table>

### Horizontal Commands

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

Table 2-10: Horizontal Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HORizontal?</td>
<td>Return horizontal settings</td>
</tr>
<tr>
<td>HORizontal:DElay?</td>
<td>Return delay time base settings</td>
</tr>
<tr>
<td>HORizontal:DElay:MODE</td>
<td>Delay time base mode</td>
</tr>
<tr>
<td>HORizontal:DElay:SCAle</td>
<td>Delay time base time/division</td>
</tr>
<tr>
<td>HORizontal:DElay:SECdiv</td>
<td>Same as HORizontal:DElay:SCAle</td>
</tr>
<tr>
<td>HORizontal:DElay:TIME</td>
<td>Delay time</td>
</tr>
<tr>
<td>HORizontal:DElay:TIME?</td>
<td>Return delay time parameters</td>
</tr>
</tbody>
</table>
**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. Because they are computed only when they are needed, immediate measurements slow the waveform update rate less than displayed measurements.

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 2-10: Horizontal Commands (Cont.)**

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HORizontal:DELay:TIME:RUNSAfter</td>
<td>Time to wait in delay-runs-after-main mode</td>
</tr>
<tr>
<td>HORizontal:MAIn?</td>
<td>Returns time/division of main time base</td>
</tr>
<tr>
<td>HORizontal:MAIn:SCAle</td>
<td>Main time base time/division</td>
</tr>
<tr>
<td>HORizontal:MAIn:SECdiv</td>
<td>Same as HORizontal:MAIn:SCAle</td>
</tr>
<tr>
<td>HORizontal:MODE</td>
<td>Turn delay time base on or off</td>
</tr>
<tr>
<td>HORizontal:POSITION</td>
<td>Portion of waveform to display</td>
</tr>
<tr>
<td>HORizontal:RECOOrdlength?</td>
<td>Return number of points in waveform record</td>
</tr>
<tr>
<td>HORizontal:REF&lt;x&gt;</td>
<td>Position lock for REF waveforms</td>
</tr>
<tr>
<td>HORizontal:SCAle</td>
<td>Same as HORizontal:MAIn:SCAle</td>
</tr>
<tr>
<td>HORizontal:SECdiv</td>
<td>Same as HORizontal:MAIn:SCAle</td>
</tr>
<tr>
<td>HORizontal:TRIGger?</td>
<td>Return trigger position</td>
</tr>
<tr>
<td>HORizontal:TRIGger:POSITION</td>
<td>Main time base trigger position</td>
</tr>
</tbody>
</table>
### Table 2-11: Measurement Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEASurement?</td>
<td>Returns all measurement parameters</td>
</tr>
<tr>
<td>MEASurement:GATING</td>
<td>Set or query measurement gating</td>
</tr>
<tr>
<td>MEASurement:IMMed?</td>
<td>Return immediate measurement parameters</td>
</tr>
<tr>
<td>MEASurement:IMMed:SOUrce[1]</td>
<td>Channel to take measurement from</td>
</tr>
<tr>
<td>MEASurement:IMMed:TYPe</td>
<td>The measurement to be taken</td>
</tr>
<tr>
<td>MEASurement:IMMed:UNIts?</td>
<td>Return measurement units</td>
</tr>
<tr>
<td>MEASurement:IMMed:VALue?</td>
<td>Return measurement result</td>
</tr>
<tr>
<td>MEASurement:MEAS&lt;x&gt;??</td>
<td>Return parameters on measurement</td>
</tr>
<tr>
<td>MEASurement:MEAS&lt;x&gt;:SOUrce[1]</td>
<td>Channel to take measurement from</td>
</tr>
<tr>
<td>MEASurement:MEAS&lt;x&gt;:STATE</td>
<td>Turn measurement display on or off</td>
</tr>
<tr>
<td>MEASurement:MEAS&lt;x&gt;:TYPe</td>
<td>The measurement to be taken</td>
</tr>
<tr>
<td>MEASurement:MEAS&lt;x&gt;:UNIts?</td>
<td>Units to use for measurement</td>
</tr>
<tr>
<td>MEASurement:MEAS&lt;x&gt;:VALue?</td>
<td>Measurement result query</td>
</tr>
<tr>
<td>MEASurement:METHod</td>
<td>Method for calculating reference levels</td>
</tr>
<tr>
<td>MEASurement:REFLevel?</td>
<td>Returns reference levels</td>
</tr>
<tr>
<td>MEASurement:REFLevel:ABSolute:HIGH</td>
<td>The top level for risetime (90% level)</td>
</tr>
<tr>
<td>MEASurement:REFLevel:ABSolute:LOW</td>
<td>The low level for risetime (10% level)</td>
</tr>
<tr>
<td>MEASurement:REFLevel:ABSolute:MID</td>
<td>Mid level for measurements</td>
</tr>
<tr>
<td>MEASurement:REFLevel:METHod</td>
<td>Method to assign HIGH and LOW levels: either % or absolute volts</td>
</tr>
<tr>
<td>MEASurement:REFLevel:PERCent:HIGH</td>
<td>The top level for risetime (90% level)</td>
</tr>
<tr>
<td>MEASurement:REFLevel:PERCent:LOW</td>
<td>The low level for risetime (10% level)</td>
</tr>
<tr>
<td>MEASurement:REFLevel:PERCent:MID</td>
<td>Mid level for measurements</td>
</tr>
</tbody>
</table>
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 2-12: Miscellaneous Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOSet</td>
<td>Automatic oscilloscope setup</td>
</tr>
<tr>
<td>*DDT</td>
<td>Define group execute trigger (GET)</td>
</tr>
<tr>
<td>FACTory</td>
<td>Reset to factory default</td>
</tr>
<tr>
<td>HDR</td>
<td>Same as HEADer</td>
</tr>
<tr>
<td>HEADer</td>
<td>Return command header with query</td>
</tr>
<tr>
<td>*IDN?</td>
<td>Identification</td>
</tr>
<tr>
<td>*LRN?</td>
<td>Learn device setting</td>
</tr>
<tr>
<td>LOCK</td>
<td>Lock front panel (local lockout)</td>
</tr>
<tr>
<td>NEWpass</td>
<td>Change password for User Protected Data</td>
</tr>
<tr>
<td>PASSWord</td>
<td>Access to change User Protected Data</td>
</tr>
<tr>
<td>REM</td>
<td>No action; remark only</td>
</tr>
<tr>
<td>*RST</td>
<td>Return most settings to factory default</td>
</tr>
<tr>
<td>RS232?</td>
<td>Query RS232 parameters</td>
</tr>
<tr>
<td>RS232:BAUd</td>
<td>Set or query baud rate</td>
</tr>
<tr>
<td>RS232:CONTrol:RTS</td>
<td>Set or query hard flagging</td>
</tr>
<tr>
<td>RS232:DCD</td>
<td>Turns off DCD monitoring</td>
</tr>
<tr>
<td>RS232:HARDFlagging</td>
<td>Set or query hard flagging</td>
</tr>
<tr>
<td>RS232:MODE</td>
<td>Query always returns RAW</td>
</tr>
<tr>
<td>RS232:PACE</td>
<td>Set or query soft flagging</td>
</tr>
<tr>
<td>RS232:PARity</td>
<td>Set or query parity type</td>
</tr>
<tr>
<td>RS232:PRESet</td>
<td>Set default RS-232 parameters</td>
</tr>
<tr>
<td>RS232:SBITS</td>
<td>Set or query number of stop bits</td>
</tr>
</tbody>
</table>
### Save and Recall Commands

Save and Recall commands allow you to store and retrieve internal waveforms and settings. When you “save a setting,” you save all the settings of the oscilloscope. When you then “recall a setting,” the oscilloscope restores itself to the state it was in when you originally saved that setting.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*RCL</td>
<td>Recall setting</td>
</tr>
<tr>
<td>RECALL:SETUp</td>
<td>Recall saved oscilloscope setting</td>
</tr>
<tr>
<td>*SAV</td>
<td>Save setting</td>
</tr>
<tr>
<td>SAVe:SETUp</td>
<td>Save oscilloscope setting</td>
</tr>
<tr>
<td>SAVe:WAVEform</td>
<td>Save waveform</td>
</tr>
</tbody>
</table>
Status and Error Commands

Table 2-14 lists the status and error commands the oscilloscope supports. These commands let you determine the status of the oscilloscope, and control events.

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 2-14: Status and Error Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLEv?</td>
<td>Return all events</td>
</tr>
<tr>
<td>BUSY?</td>
<td>Return scope status</td>
</tr>
<tr>
<td>*CLS</td>
<td>Clear status</td>
</tr>
<tr>
<td>DESE</td>
<td>Device event status enable</td>
</tr>
<tr>
<td>*ESE</td>
<td>Event status enable</td>
</tr>
<tr>
<td>*ESR?</td>
<td>Return standard event status register</td>
</tr>
<tr>
<td>EVENT?</td>
<td>Return event code</td>
</tr>
<tr>
<td>EVMsg?</td>
<td>Return event code and message</td>
</tr>
<tr>
<td>EVQty?</td>
<td>Return number of events in queue</td>
</tr>
<tr>
<td>ID?</td>
<td>Identification</td>
</tr>
<tr>
<td>*OPC</td>
<td>Operation complete</td>
</tr>
<tr>
<td>*PSC</td>
<td>Power-on status clear</td>
</tr>
<tr>
<td>*PUD</td>
<td>Query or set User Protected Data</td>
</tr>
<tr>
<td>*SRE</td>
<td>Service request enable</td>
</tr>
<tr>
<td>*STB?</td>
<td>Read status byte</td>
</tr>
<tr>
<td>*WAI</td>
<td>Wait to continue</td>
</tr>
</tbody>
</table>

Trigger Commands

Trigger commands control all aspects of oscilloscope triggering.

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

Edge triggering lets you display a waveform at or near the point where 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.
### Command Groups

#### Table 2-15: Trigger Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRIGger</td>
<td>Force trigger event; return parameters</td>
</tr>
<tr>
<td>TRIGger:MAIn</td>
<td>Main trigger level to 50%</td>
</tr>
<tr>
<td>TRIGger:MAIn:EDGE?</td>
<td>Return main edge trigger parameters</td>
</tr>
<tr>
<td>TRIGger:MAIn:EDGE:COUPling</td>
<td>Main trigger coupling</td>
</tr>
<tr>
<td>TRIGger:MAIn:EDGE:SLOpe</td>
<td>Main trigger slope</td>
</tr>
<tr>
<td>TRIGger:MAIn:EDGE:SOURce</td>
<td>Main trigger source</td>
</tr>
<tr>
<td>TRIGger:MAIn:HOLDoff?</td>
<td>Main trigger holdoff value</td>
</tr>
<tr>
<td>TRIGger:MAIn:HOLDoff:VALue</td>
<td>Main trigger holdoff value</td>
</tr>
<tr>
<td>TRIGger:MAIn:LEVEL</td>
<td>Main trigger level</td>
</tr>
<tr>
<td>TRIGger:MAIn:MODE</td>
<td>Main trigger mode</td>
</tr>
<tr>
<td>TRIGger:MAIn:TYPE</td>
<td>Main trigger edge or video</td>
</tr>
<tr>
<td>TRIGger:MAIn:VIDeo:FIELD</td>
<td>Video trigger field</td>
</tr>
<tr>
<td>TRIGger:MAIn:VIDeo:HOLDoff?</td>
<td>Return video trigger holdoff</td>
</tr>
<tr>
<td>TRIGger:MAIn:VIDeo:HOLDoff:VALue</td>
<td>Video trigger holdoff value</td>
</tr>
<tr>
<td>TRIGger:MAIn:VIDeo:SCAN</td>
<td>Video trigger scan rate</td>
</tr>
<tr>
<td>TRIGger:MAIn:VIDeo:SOURce</td>
<td>Video trigger source</td>
</tr>
<tr>
<td>TRIGger:STATE?</td>
<td>Trigger system status</td>
</tr>
</tbody>
</table>

#### Vertical Commands

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

#### Table 2-16: Vertical Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH&lt;x&gt;?</td>
<td>Return vertical parameters</td>
</tr>
<tr>
<td>CH&lt;x&gt;:BANdwidth</td>
<td>Channel bandwidth</td>
</tr>
<tr>
<td>CH&lt;x&gt;:COUPling</td>
<td>Channel coupling</td>
</tr>
<tr>
<td>CH&lt;x&gt;:INVert</td>
<td>Invert channel</td>
</tr>
</tbody>
</table>
Table 2-16: Vertical Commands (Cont.)

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH&lt;x&gt;:OFFSET</td>
<td>Channel offset</td>
</tr>
<tr>
<td>CH&lt;x&gt;:POSITION</td>
<td>Channel position</td>
</tr>
<tr>
<td>CH&lt;x&gt;:PRObe?</td>
<td>Return channel probe attenuation</td>
</tr>
<tr>
<td>CH&lt;x&gt;:SCAle</td>
<td>Channel volts/div</td>
</tr>
<tr>
<td>CH&lt;x&gt;:VOLts</td>
<td>Same as CH&lt;x&gt;:SCAle</td>
</tr>
<tr>
<td>MATH1?</td>
<td>Return math waveform definition</td>
</tr>
<tr>
<td>MATH1:DEFINE</td>
<td>Math waveform</td>
</tr>
<tr>
<td>SESelect?</td>
<td>Return selected waveform</td>
</tr>
<tr>
<td>SESelect:&lt;wfm&gt;</td>
<td>Set selected waveform</td>
</tr>
<tr>
<td>SESelect:CONTROL</td>
<td>Front-panel channel selector</td>
</tr>
</tbody>
</table>
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 min/max pair. Before you can transfer waveform data, you must specify the data format, record length, and waveform locations.

Waveform Data Formats

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 HIRes or AVERAGE modes use 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** — 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** — 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: RIIBinary, RPBinary, SRIBinary, and SRPbinary.

RIIBinary is signed integer where the most significant byte is transferred first, and RPBinary is positive integer where the most significant byte is transferred first. SRIBinary and SRPBinary correspond to RIIBinary and RPBinary respectively but use a swapped byte order where 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 500th 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 the record length 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 TDS 310, TDS 320, and TDS 350 accept 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 2-164 for more information about the waveform preamble.

Scaling Waveform Data

Once the oscilloscope has transferred the waveform data to the controller, the oscilloscope converts the waveform data points into voltage values for analysis using information from the waveform preamble.
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 2-17: Waveform Commands

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURVe</td>
<td>Transfer waveform data</td>
</tr>
<tr>
<td>DATa</td>
<td>Waveform data format and location</td>
</tr>
<tr>
<td>DATa:DESTination</td>
<td>Destination for waveforms sent to oscilloscope</td>
</tr>
<tr>
<td>DATa:ENCdg</td>
<td>Waveform data encoding method</td>
</tr>
<tr>
<td>DATa:SOURce</td>
<td>Source of CURVe? data</td>
</tr>
<tr>
<td>DATa:STARt</td>
<td>Starting point in waveform transfer</td>
</tr>
<tr>
<td>DATa:STOP</td>
<td>Ending point in waveform transfer</td>
</tr>
<tr>
<td>DATa:TARget</td>
<td>Same as DATa:DESTination</td>
</tr>
<tr>
<td>DATa:WIDth</td>
<td>Byte width of waveform points</td>
</tr>
<tr>
<td>WAVFrm?</td>
<td>Returns waveform preamble and data</td>
</tr>
<tr>
<td>Header</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>WAVPre?</td>
<td>Returns waveform format data</td>
</tr>
<tr>
<td>WFMPre:BIT_Nr</td>
<td>Preamble bit width of waveform points</td>
</tr>
<tr>
<td>WFMPre:BN_Fmt</td>
<td>Preamble binary encoding type</td>
</tr>
<tr>
<td>WFMPre:BYT_Nr</td>
<td>Preamble byte width of waveform points</td>
</tr>
<tr>
<td>WFMPre:BYT_Or</td>
<td>Preamble byte order of waveform points</td>
</tr>
<tr>
<td>WFMPre:ENCdg</td>
<td>Preamble encoding method</td>
</tr>
<tr>
<td>WFMPre:NR_Pt</td>
<td>Number of points in the curve</td>
</tr>
<tr>
<td>WFMPre:PT_Fmt</td>
<td>Format of curve points</td>
</tr>
<tr>
<td>WFMPre:PT_Off</td>
<td>Trigger position</td>
</tr>
<tr>
<td>WFMPre:WFIId</td>
<td>Curve identifier</td>
</tr>
<tr>
<td>WFMPre:XINcr</td>
<td>Horizontal sampling interval</td>
</tr>
<tr>
<td>WFMPre:XMUlt</td>
<td>Horizontal scale factor</td>
</tr>
<tr>
<td>WFMPre:XOFF</td>
<td>Horizontal offset</td>
</tr>
<tr>
<td>WFMPre:XUNit</td>
<td>Horizontal units</td>
</tr>
<tr>
<td>WFMPre:XZErO</td>
<td>Horizontal origin offset</td>
</tr>
<tr>
<td>WFMPre:YMUlt</td>
<td>Vertical scale factor</td>
</tr>
<tr>
<td>WFMPre:YOFF</td>
<td>Vertical offset</td>
</tr>
<tr>
<td>WFMPre:YUNit</td>
<td>Vertical units</td>
</tr>
<tr>
<td>WFMPre:YZErO</td>
<td>Offset voltage</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:NR_Pt</td>
<td>Number of points in the curve</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:PT_Fmt</td>
<td>Format of curve points</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:PT_Off</td>
<td>Trigger position</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:WFIId</td>
<td>Curve identifier</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:XINcr</td>
<td>Horizontal sampling interval</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:XUNit</td>
<td>Horizontal units</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:YMUlt</td>
<td>Vertical scale factor</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:YOFF</td>
<td>Vertical offset</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:YUNit</td>
<td>Vertical units</td>
</tr>
<tr>
<td>WFMPre:&lt;wfm&gt;:YZErO</td>
<td>Offset voltage</td>
</tr>
</tbody>
</table>
Command Groups
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?

![ACQuire?](image)

**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 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 upper left corner 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> \(\neq 0\) 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> ≠ 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> ≠ 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?

![Diagram](attachment:image.png)

**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>

```
ALIAS -> DELETE -> <Space> -> <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

```
ALIAS -> DELETE -> ALL ->
```

**Examples:** ALIAS:DELETE:ALL

deletes all aliases.
Command Descriptions

**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> )

**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?, EVQt?, *SRE, *STB?

**Syntax:** ALLEv?

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

\[
\text{<Event Code><Comma><QString>[<Comma><Event Code><Comma><QString>...]}
\]

\[
\text{<QString>::=}<\text{Message}>;[<\text{Command}>]
\]

The command string 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:**

\[
\text{ALLEV? might return the string :ALLEV 2225,“Measurement error, No waveform to measure; ”,420,”Query UNTERMINATED; ”.}
\]
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 310, TDS 320, & TDS 350 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 displayed 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 2-18.

\(<NR1> = 1 \) means that the oscilloscope is busy processing one of the commands listed in Table 2-18.

**Table 2-18: Commands that Affect BUSY? Response**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single sequence acquisition</td>
<td>ACQuire:STATE ON or ACQuire:STATE RUN (when ACQuire:STOPAfter is set to SEQuence)</td>
</tr>
<tr>
<td>Hard copy output</td>
<td>HARDCopy START</td>
</tr>
<tr>
<td>Signal path compensation</td>
<td>*CAL? or CALibrate</td>
</tr>
</tbody>
</table>

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

*CAL? (Query Only)

Instructs the oscilloscope to perform an internal self-calibration and return its calibration status.

**NOTE**

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

**Syntax:** *CAL?

**Returns:**

\(<\text{NR1}\> = 0\) indicates that the calibration completed without any errors detected.  
\(<\text{NR1}\> \neq 0\) 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)

The CALibrate command instructs the oscilloscope to perform an internal self-calibration.

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:** *CAL?

**Syntax:** CALibrate INTERNAL

**Arguments:** INTERNAL specifies an internal self-calibration.

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

---

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 10.0E-3;POSITION 0.0E+0;
OFFSET 0.0E+0;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:**

\[
\text{CH<x>:COUPLing} \{ \text{AC | DC | GND} \}
\]

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.
Command Descriptions

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 2-19: Offset Ranges (All Channels)

<table>
<thead>
<tr>
<th>CH&lt;x&gt;:SCAle</th>
<th>OFFSet Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 mV/div – 99.5 mV/div</td>
<td>± 1 V</td>
</tr>
<tr>
<td>100 mV/div – 995 mV/div</td>
<td>±10 V</td>
</tr>
<tr>
<td>1 V/div – 10 V/div</td>
<td>±100 V</td>
</tr>
</tbody>
</table>

Examples: CH1:OFFSET 0.5E+00

lowers the channel 1 displayed waveform by 0.5 V.

CH1:OFFSET?

might return 500.0E-3, 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 1.3E+00
positions the channel 2 input signal 1.3 divisions above the center of the display.

CH1:POSITION?
might return −1.3E+00, 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 1.0E+1 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:VOLt

**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 100E-03
  sets the channel 1 gain to 100 mV/div.
- CH2:SCALE?
  might return 1.00E+0, 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 100E-03
```

sets the channel 1 gain to 100 mV/div.

```
CH2:VOLTS?
```

might return 1.00E+0, 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. 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.
**Command Descriptions**

---

**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:**

```plaintext
CURSOR?
might return :CURSOR:FUNCTION OFF;VBARS:UNITS SECONDS;
POSITION1 500.0E-6;POSITION2 4.50E-3;SELECT CURSOR1;
:CURSOR:HBARS:POSITION1 3.20E+0;POSITION2 -3.20E+0;
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 volts.
- **OFF** removes the cursors from the display.
- **VBArS** specifies vertical bar cursors that measure time or frequency.
- **PAIred** specifies paired cursors that show both time and volts.

**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.
Command Descriptions

CURSor:HBArS:DELTa? (Query Only)

Returns the voltage difference between the two horizontal bar cursors.

**Group:** Cursor

**Syntax:** CURSor:HBArS:DELTa?

**Returns:** <NR3>

**Examples:** CURSOR:HBARS:DELTA?
might return 5.08E+0 for the voltage 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 volts.

**Examples:** CURSOR:HBARS:POSITION1 25.0E-3
positions one of the horizontal cursors at 25.0 mV.

CURSOR:HBARS:POSITION2?
might return –64.0E-3, 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 when the cursor menu is active. The unselected cursor is displayed as a dashed horizontal line. This command is equivalent to pressing the TOGGLE button on the front panel when the Cursor menu is displayed.

**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.
Command Descriptions

CURSor:PAIred:HDELTA (Query Only)

Queries the hbar (voltage) 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?

```
CURSor 1 PAIred 2 HDELTA 3
```

**Examples:** CURSOR:PAIRED:HDELTA?
might return 5.08E+0 for the voltage difference between the two cursors.

CURSor:PAIred:HPOS1 (Query Only)

Queries the horizontal bar (voltage) position of the first paired cursor.

**Group:** Cursor

**Related Commands:** CURSor:FUNCtion

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

```
CURSor 1 PAIred 2 HPOS1 3
```

**Examples:** CURSOR:PAIRED:HPOS1?
might return -64.0E-3, indicating that the first cursor is at -64.0 mV.
CURSor:PAIred:HPOS2 (Query Only)

Queries the horizontal bar (voltage) position of the second paired cursor.

**Group:** Cursor

**Related Commands:** CURSor:FUNCtion

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

**Examples:**

CURSOR:PAIRED:HPOS2?

might return $-64.0\times10^{-3}$, indicating the second cursor is at $-64.0$ mV.

CURSor:PAIred:POSITION1

Sets or queries the vertical bar (time) 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 9.00E-6

specifies the first paired cursor is at 9 μs.

CURSOR:POSITION1?

might return 1.00E-6, indicating that the first paired cursor is at 1 μs.
Command Descriptions

**CURSor:PAIred:POSITION2**

Sets or queries the vertical bar (time) 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 1.00E-6, indicating that the second paired cursor is at 1 μs.
CURSor:PAIred:SELet

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 TOGGLE button on the front panel when the cursor menu is displayed.

**Group:** Cursor

**Syntax:**

```
CURSor:PAIred:SELet { CURSOR1 | CURSOR2 }
CURSor:PAIred:SELet?
```

**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 vbar (time) distance between paired cursors.

**Group:** Cursor

**Related Commands:** CURSor:FUNCtion

**Syntax:**

```
CURSor:PAIred:VDELTA?
```

**Examples:**

- CURSOR:PAIRED:VDELTA?
  might return 1.064E+00, indicating that the time between the paired cursors is 1.064 seconds.
Command Descriptions

**CURSor:VBArS**

Positions the vertical bar cursors and CURSor:VBArS? returns the current 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 1.00E-6;POSITION2 9.00E-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?

![Diagram](CURSor:VBArs:DELTa)

**Returns:** <NR3>

**Examples:** CURSOR:VBARS:DELTa?
 might return 1.064E+00, 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 is specified by the CURSor:VBArs:UNIts command.

**Group:** Cursor

**Related Commands:** CURSor:VBArs:UNIts

**Syntax:** CURSor:VBArs:POSITION<x>  <NR3>
CURSor:VBArs:POSITION<x>? 

![Diagram](CURSor:VBArs:POSITION)

**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 9.00E-6
 positions one of the vertical bar cursors at 9 μs.

CURSOR:VBARS:POSITION1?
 might return 1.00E-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** when the cursor menu is active. The unselected cursor is displayed as a dashed vertical line. This command is equivalent to pressing the **TOGGLE** button on the front panel when the Cursor menu is displayed.

**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.

**Group:** Cursor

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

**Syntax:**

```plaintext
CURSor:VBArS:UNIts { SECOnd | HERTz }
CURSor:VBArS:UNIts?
```

**Examples:**

```plaintext
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 2-164 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.

A description of the waveform transfer process starts on page 2-22.

**Group:** Waveform

**Related Commands:** DATA, WFMPre

**Syntax:**

```
CURVe { <Block> | <asc 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 specifics.

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

---

**2-62 Syntax and Commands**
**Examples:** CURVE?

might return, for ASCII data: CURVE

0, 0, 0, 0, -1, 1, 0, -1, 0, 0, -1, 0, -1, 0, -1, 0, 0, 0, -1, 0, 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:**

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

DATA?

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

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 2-20. 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: ASCIIi 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>
<thead>
<tr>
<th>DATa:ENCdg Setting</th>
<th>WFMPre Setting</th>
<th>WFMPre Setting</th>
<th>WFMPre Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ENCDg</td>
<td>BN_Fmt</td>
<td>BYT_Or</td>
</tr>
<tr>
<td>ASCIIi</td>
<td>ASC</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>RIBinary</td>
<td>BIN</td>
<td>RI</td>
<td>MSB</td>
</tr>
<tr>
<td>RPBinary</td>
<td>BIN</td>
<td>RP</td>
<td>MSB</td>
</tr>
<tr>
<td>SRIBinary</td>
<td>BIN</td>
<td>RI</td>
<td>LSB</td>
</tr>
<tr>
<td>SRPbinary</td>
<td>BIN</td>
<td>RP</td>
<td>LSB</td>
</tr>
</tbody>
</table>
Command Descriptions

**Examples:**

DATA:ENC DG RP BINARY
sets the data encoding format to be positive integer where the most significant byte is transferred first.

DATA:ENC DG?
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, MATH1, REF1, REF2.

**Group:** Waveform

**Syntax:** DATa:SOUrce <wfm>[<Comma><wfm>]...
DATa:SOUrce?

**Arguments:** <wfm> is the location of the waveform data that will be 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, then REF2.

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

---

2-66 Syntax and Commands
**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 allows the transfer of 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 the record length 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, WFMP:BIT_Nr, WFMP: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.
**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.
Command Descriptions

DESE

Sets and 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 (ESER), 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”
Command Descriptions

**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:FLAgr? and the internal log is returned by DIAg:RESUlt:LOGr?. 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. This command performs a warm boot and does not return control to the instrument controller until diagnostics are complete.*

**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. If the PON event was enabled before running the tests, a Service Request is generated. When the Service Request is received, the pass/fail status of the tests can be returned by executing DIAg:RESUlt:FLAgr?.

The DIAg:STATE EXECute command clears the following:

- the Event Queue
the Input Queue
- the Status Registers (SES and SBR)

To enable a power-on event to generate a Service Request, send the following commands before running diagnostics:
- DESE 128
- *ESE 128
- *SRE 32
- *PSC 0

**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:  
**Examples:** DISPlay?

might return :DISPLAY:FORMAT YT;STYLE VECTORS;PERSISTENCE 500.0E-3;GRATICULE FULL;TRIGT 1;INTENSITY:OVERALL 85;WAVEFORM 70;TEXT 60;CONTRAST 150
Command Descriptions

**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 2-21. Displaying one source causes its corresponding source to be displayed.

**Table 2-21: XY Format Pairs**

<table>
<thead>
<tr>
<th>X-Axis Source</th>
<th>Y-Axis Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ch 1</td>
<td>Ch 2</td>
</tr>
<tr>
<td>Ref 1</td>
<td>Ref 2</td>
</tr>
</tbody>
</table>

*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.
```
**DIStplay:GRAticule**

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

**Group:** Display

**Syntax:**

```
DIStplay:GRAticule { FRAmes | FULL } 
DIStplay:GRAticule? 
```

**Arguments:**

- `FRAmes` specifies just a frame.
- `FULL` specifies a frame, a grid, and cross hairs.

**Examples:**

- `DIStplay:GRAticule FRAME`
  sets the graticule type to display the frame only.

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

**DIStplay:INTENSITTy? (Query Only)**

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

**Group:** Display

**Syntax:**

```
DIStplay:INTENSITTy? 
```

**Examples:**

- `DIStplay:INTENSITTy?`
  might return:
  ```
  :DIStplay:INTENSITTy: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 }
```

**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 { INFinite | <NR3> }
```

```
DISPLAY:PERSISTENCE?
```

**Arguments:**

- `<NR3>` specifies the length, in seconds, that the waveform points are displayed on the screen. The range is 500 ms to 10 s.
- `INFinite` specifies infinite persistence.

**Examples:**

```
DISPLAY:PERSISTENCE 3
```

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

**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> ≠ 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 and 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, URO, 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 (ESR). *ESR? also clears the ESR (since reading the ESR 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?

```
*ESR ?
```

**Examples:** *ESR?

might return the value 213, showing that the ESR 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?

```
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:**

EVMG?

might return the message :EVMG 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.
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

Setting the oscilloscope to factory default performs the following operations:

- 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-C or GPIB (IEEE 488.1) interfaces
- The selected GPIB address
- Calibration data that affects device specifications
- Protected user data
- Stored settings
- The current password (if implemented)
- Hardcopy parameters

---

**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.
- **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**

*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.*

**Examples:**
```
HARDCOPY ABORT
```

stops any hardcopy output that is in process.
**HARDCopy:FORMAT**

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

**Group:** Hardcopy

**Syntax:**

```
HARDCopy:FORMAT ( BMP | DESKJet | DPU411 | DPU412 | EPSImage | EPSOn | INTERLeaf | LASERJet | PCX | THInkjet | TIFF )
```

```
HARDCopy: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.
Command Descriptions

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}
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.

**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?
```

```
<Space>
```

might return GPIB as the selected hardcopy output port.
**HEADer**

Sets and 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> ≠ 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.
Command Descriptions

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 5.0E+0;TRIGGER:POSITION 50;:HORIZONTAL:MAIN:SCALE 1.0E–6;:HORIZONTAL:DELAY:MODE RUNS-AFTER;SCALE 1.0E–6;TIME: 16.0E–9;:HORIZONTAL:REF1 LOCK;REF2 LOCK

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


**Syntax:** HORizontal:DELay?

**Examples:**

HORIZONTAL:DELY?

might return the delay parameters :HORIZONTAL:DELY:MODE RUNS-AFTER;SCALE 1.0E–6;TIME: 16.0E–9
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.
Command Descriptions

**HORizontal:DE Lay:SCAle**

Sets the time per division for the delayed time base. This is equivalent to setting **Delayed Scale** in the Horizontal Scale side menu.

**Group:** Horizontal

**Related Commands:** HORizontal:DE Lay:SECdiv

**Syntax:**

```
HORizontal:DE Lay:SCAle <NR3>
HORizontal:DE Lay:SCAle?
```

**Arguments:** `<NR3>` is the time per division. The range is 10 ns (TDS 310), 5 ns (TDS 320), or 2.5 ns (TDS 350) to 5 s in a 1–2–5 sequence. Values that are not in a 1–2–5 sequence is 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 is 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 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

**Syntax:**  
HORizontal:DELay:TIME <NR3>

**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 2-22).

**Table 2-22: Horizontal Delay Time Resolution**

<table>
<thead>
<tr>
<th>Time Base Setting</th>
<th>Delay Time Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 µs or faster</td>
<td>16.5 ns</td>
</tr>
<tr>
<td>2.5 µs</td>
<td>49.5 ns</td>
</tr>
<tr>
<td>5 µs</td>
<td>99 ns</td>
</tr>
<tr>
<td>10 µs</td>
<td>198 ns</td>
</tr>
<tr>
<td>slower than 10 µs</td>
<td>one sample interval</td>
</tr>
<tr>
<td></td>
<td>(0.02 x the delayed time base setting)</td>
</tr>
</tbody>
</table>

**Examples:**  
HORizontal:DELAY:TIME 2.0E-3

sets the delay time between the main and delayed time base to 2 ms.
HORizontal:DELay:TI Me? (Query Only)

Returns the delay time parameters.

**Group:** Horizontal

**Related Commands:** HORizontal:DELay:TI Me:RUNSAfter?

**Syntax:** HORizontal:DELay:TI Me?

![Diagram](image)


HORizontal:DELay:TI Me: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.

**Group:** Horizontal

**Related Commands:** HORizontal:DELay:MODe

**Syntax:** HORizontal:DELay:TI Me:RUNSAfter &lt;NR3&gt;

HORizontal:DELay:TI Me:RUNSAfter?

![Diagram](image)

**Arguments:** &lt;NR3&gt;, see HORizontal DELay:TI Me.

**Examples:** HORizontal:DELay:TI Me:RUNSAfter 2.0E-3 sets the delay time between the main and delayed time base to 2 ms.
Command Descriptions

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?

![](image)

**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 Main Scale in the Horizontal Scale side menu.

**Group:** Horizontal


**Syntax:**
HORizontal:MAIN:SCAle <NR3>
HORizontal:MAIN:SCAle?

![](image)

**Arguments:**
<NR3> is the time per division. The range is 10 ns (TDS 310), 5 ns (TDS 320), or 2.5 ns (TDS 350) 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


**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. The level of intensity is set by 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 or setting the position in the Horizontal Position side menu.

**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: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.
Command Descriptions

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?

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%, and is the amount of pretrigger information in the waveform.

**Examples:**
- HORIZONTAL:TRIGGER:POSITION?
  - might return 50.
Command Descriptions

ID? (Query Only)

Returns identifying information about the oscilloscope and its firmware.

*Group: Status and Error

*Related Commands: *IDN?

*Syntax: ID?

*Returns: The oscilloscope identification in the following format:

```
TEK/<model number>,CF:91.1CT,FV:<firmware version number>
```

*Examples: ID?

might return TEK/TDS350,CF:91.1CT,FV:1.0

*IDN? (Query Only)

Returns the oscilloscope identification code.

*Group: Miscellaneous

*Related Commands: ID

*Syntax: *IDN?

*Returns: The oscilloscope identification in the following format:

```
TEKTRONIX,<model number>,0,CF:91.1CT FV:<firmware version number>
```

*Examples: *IDN?

might return the response

TEKTRONIX,TDS350,0,CF:91.1CT FV:1.0
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 }

LOCK?
```

**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 ALL
locks the front-panel controls.

LOCK?
returns NONE when the front-panel controls are enabled by this command.
```
Command Descriptions

**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?

![Diagram](image)

**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:**

*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?

![Diagram](image)
MATH1:DEFINE

Allows the user to define a new waveform using a mathematical expression.

**Group:** Vertical

**Syntax:**

MATH1:DEFINE <QString>

MATH1:DEFINE?

**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>
```

**Examples:**

MATH1:DEFINE "Ch1 + ch2"

sets the math waveform so that it displays the sum of channel 1 and channel 2.
MEASUrement? (Query Only)

Returns all measurement parameters.

**Group:** Measurement

**Syntax:** MEASUrement?

**Examples:**

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 0.0E+0;LOW 0.0E+0;MID 0.0E+0;MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0E+0;LOW 10.0E+0;MID 50.0E+0
MEASUrement: GATing

Sets or queries measurement gating.

**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` showing 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> | 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 voltage 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 (voltage).
- **MEAN** is the arithmetic mean over the entire waveform.
- **MINImum** is the lowest amplitude (voltage).
- **NDUTy** is the ratio of the negative pulse width to the signal period expressed as a percentage.
**Command Descriptions**

NO\text{vershoot} is the negative overshoot, expressed as:

\[ NO\text{vershoot} = 100 \times \left( \frac{\text{Low} - \text{Minimum}}{\text{Amplitude}} \right) \]

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.

PO\text{vershoot} is the positive overshoot, expressed as:

\[ PO\text{vershoot} = 100 \times \left( \frac{\text{Maximum} - \text{High}}{\text{Amplitude}} \right) \]

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 voltage.

**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, "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>
Command Descriptions

MEASUrement:MEAS<x>? (Query Only)

Returns all measurement parameters for the displayed measurement specified by <x>.

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> | MATH1 | REF<x> }

Arguments: CH<x> is an input channel.
            MATH1 is the math waveform.
            REF<x> is a reference waveform.

Examples: MEASUREMENT:MEAS2:SOURe1 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:CH<x> 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 deselected the source.
- ON or <NR1> ≠ 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 voltage 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 (voltage).

**MEAN** is the arithmetic mean over the entire waveform.

**MINIMUM** is the lowest amplitude (voltage).
**NDUTy** is the ratio of the negative pulse width to the signal period expressed as a percentage.

**NOVershoot** is the negative overshoot, expressed as:

$$\text{NOVershoot} = 100 \times \left( \frac{\text{Low} - \text{Minimum}}{\text{Amplitude}} \right)$$

**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:

$$\text{POVershoot} = 100 \times \left( \frac{\text{Maximum} - \text{High}}{\text{Amplitude}} \right)$$

**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 voltage.

**Examples:**

```
MEASUREMENT:MEAS3:TYPE RMS
```

specifies MEAS3 to calculate the Root Mean Square voltage.
Command Descriptions

**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, “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 reference levels.

**Group:** Measurement

**Syntax:**

```
MEASUrement:REFLevel?
```

```
MEASUrement:REFLevel:ABSoolute: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:ABSoolute:HIGH <NR3>
```

```
MEASUrement:REFLevel:ABSoolute:HIGH?
```

**Arguments:**

<NR3> is the high reference level, in volts. The default is 0.0 V.

**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 volts. The default is 0.0 V.

**Examples:**

```
MEASUREMENT:REFLEVEL:ABSOLUTE:LOW?
```

might return 0.0E+0 as the low reference level.
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 volts. The default is 0.0 V.

**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 2-23 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 2-23: Commands that Generate an Operation Complete Message**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic scope adjustment</td>
<td>AUTOSet EXECute</td>
</tr>
<tr>
<td>Internal self-calibration</td>
<td>*CAL</td>
</tr>
<tr>
<td>Single sequence acquisition</td>
<td>ACQuire:STATE ON or ACQuire:STATE RUN</td>
</tr>
<tr>
<td></td>
<td>(when ACQuire:STOPAfter is set to SEQUENCE)</td>
</tr>
<tr>
<td>Hardcopy output</td>
<td>HARDCopy STARt</td>
</tr>
</tbody>
</table>
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 and 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> ≠ 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 Setup 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 Setup menu. This command is the same as the FACTory command.

**Group:** Save and Recall

**Related Commands:** FACTory, *RCL, *RST, *SAV, SAVE:SETUp, TEKSecure

**Syntax:** RECALL:SETUp { FACTory | <NR1> }

![Syntax Diagram]

**Arguments:** FACTory selects the factory setup.

<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:**

RECALL:SETUp FACTORY

recalls the front-panel setup to its factory defaults.

---

**REM (No Query Form)**

Specifies a comment. Comments are ignored by the oscilloscope.

**Group:** Miscellaneous

**Syntax:** REM <QString>

![Syntax Diagram]

**Arguments:**<QString> is a string that can have a maximum of 80 characters.

**Examples:** REM “This is a comment” is ignored by the oscilloscope.
Command Descriptions

*RST (No Query Form)

(Reset) Returns the oscilloscope to a known set of oscilloscope settings, but
does not purge any aliases or stored settings.

Group: Status and Error


Syntax: *RST

*RST returns oscilloscope settings to the factory defaults (see Appendix D).
The *RST command does not alter the following items:

- The state of the RS-232 or IEEE 488.1 interfaces
- The selected IEEE Std 488.1 – 1987 address of the oscilloscope
- Calibration data that affect device specifications
- The Output Queue
- The Service Request Enable Register setting
- The Standard Event Status Enable Register setting
- The Power-on status clear flag setting
- Alias definitions
- Stored settings

RS232? (Query Only)

Queries the RS232 settings.

Group: Miscellaneous


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


Syntax:  RS232: BAUD <NR1>

RS232: BAUD?

Arguments:  <NR1> where <NR1> can be 300, 600, 1200, 2400, 4800, 9600 or 19200.

Examples:  RS232: BAUD 9600
sets the transmission rate to 9600 baud.

RS232: CONTroll: RTS

Sets or queries the state of RS232 hard flagging. This command performs the same function as RS232: HARDFlagging.

Group:  Miscellaneous


Syntax:  RS232: CONTroll: RTS { ON | RFR | IBFull }

RS232: CONTroll: RTS?
Command Descriptions

**Arguments:**

- `<ON>` asserts RTS (Request to Send).
- `<RFR>` enables hard flagging.
- `<IBFull>` enables hard flagging.

**Examples:**

```
RS232:CONTrol:RTS RFR
```

enables hard flagging.

---

**RS232:DCD**

The oscilloscope accepts but ignores this command.

**Group:** Miscellaneous


**Syntax:**

```
RS232:DCD { OFF | 0 }
RS232:DCD?
```

**Arguments:** OFF or 0 turn off DCD monitoring.

**Examples:**

```
RS232:DCD OFF
```

turns off DCD monitoring. This is essentially a no-op since DCD monitoring is always off.

---

**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 deasserts...
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


**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


**Syntax:**

```
RS232:MODE RAW
RS232:MODE?
```

**Arguments:**

- RAW (GPIB emulation) mode.
Command Descriptions

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


Syntax: RS232:PACE { XON | NONE }

Arguments: <XON> turn on softflagging.
<NONE> turn off softflagging.

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 couple of characters may be lost.

Group: Miscellaneous

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


Syntax:  RS232:PRESEt

Arguments:  None.
Command Descriptions

**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:**


**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:**

**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:**


**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?
```
**TERMinator:TRANsmit**

**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 the **Save Current Setup** in the Save/Recall Setup menu.

**Group:** Save and Recall

**Related Commands:** FAcitory, *RCL, RECALL:SETU, SAVe:SETU

**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.
Command Descriptions

**SAVe:SETUp (No Query Form)**

Saves the current front-panel setup into the specified memory location. This is equivalent to selecting the **Save Current Setup** in the Save/Recall Setup menu.

**Group:** Save and Recall

**Related Commands:** RECALL:SETUp, *RCL, *SAV

**Syntax:** SAVe:SETUp <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:**
```
SAVE:SETUP 5
```
saves the current front-panel setup in memory location 5.

**SAVe:WAVEform (No Query Form)**

Stores a waveform in one of four reference memory locations.

**Group:** Save and Recall

**Syntax:** SAVe:WAVEform <wfm><Comma>REF<x>

**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.

**Examples:**
```
SAVE:WAVEFORM MATH1,REF1
```
saves the math 1 waveform in reference memory location 1.
**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
```

---

**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> ≠ 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 affected 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 configuration information 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?, VERBose

**Syntax:** SET?

![SET?](image)

**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 able to be sent back to the oscilloscope as a command string. The VERBose command can still be used to specify whether the returned headers should be abbreviated or full length.

**Examples:**

a partial return string may look like this:

```
:ACQUIRE:STOPAFTER RUNSTOP;STATE 1;MODE SAMPLE;NUMENV 8;NUMAVG 16;:HEADER 1;:VERBose 1;:ALIAS:STATE 0;:DISPLAY:FORMAT YT;STYLE VECTORS;PERSISTENCE 500.0E-3;GRATICULE FULL;TRIGT 1;INTENSITY:OVERALL 85;WAVEFORM BRIGHT;TEXT DIM;CONTRAST 150;:LOCK NONE;:HARDCOPY:FORMAT EPSIMAGE;PORT GPIB;LAYOUT POR- TRAIT;
```
*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


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:**

```plaintext
*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
```
Command Descriptions

*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 -480.0E-3;HOLDOFF:VALUE 0;:TRIGGER:MAIN:EDGE:SOURCE CH1; COUPLING DC;SLOPE RISE;:TRIGGER:MAIN:VIDEO:SOURCE CH2;FIELD FIELD1
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:**

```plaintext
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:**

```plaintext
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:**

```plaintext
TRIGger:MAIn:HOLdoff?
```

**Examples:**

```plaintext
TRIGGER:MAIN:HOLDoff?
```
**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.
- **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 acts on.

*Group:* Trigger

*Syntax:* `TRIGger:MAIn:VIDeo:FIELD { FIELD1 | FIELD2 | ANYField | ANY }`

TRIGger:MAIn:VIDeo:FIELD?

*Arguments:* FIELD1 specifies interlaced video field 1.
FIELD2 specifies interlaced video field 2.
ANYField specifies any field.
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?
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 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.
- RATE2 specifies a 20 to 25 kHz line rate.
- RATE3 specifies a 25 to 35 kHz line rate.
- RATE4 specifies a 35 to 50 kHz line rate.
- RATE5 specifies a 50 to 65 kHz line rate.

**Examples:**

```
TRIGGER:MAIN:VIDeo:SCAN RATE1
```

selects rate 1.

---

**TRIGiger:MAIn:VIDeo:SOUrce**

Sets or queries the source for the main video trigger. This is equivalent to selecting the source in the Video Source side menu.

**Group:** Trigger

**Syntax:**

```
TRIGger:MAIn:VIDeo:SOUrce { CH<x> }
```

```
TRIGger:MAIn:VIDeo:SOUrce?
```
Arguments: CH<x> specifies one of the input channels (CH1 or CH2).

Examples: TRIGGER:MAIN:VIDEO:SOURCE CH1
          selects channel 1 as the source for the main video trigger.

---

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.
Command Descriptions

*TST? (Query Only)

(Self-Test) Tests the GPIB 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.
**VERBose**

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:**

```
VERBose { OFF | ON | <NR1> }
```

**Arguments:**

- **ON or <NR1> ≠ 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:**

```
VERBOSE ON
```

sets the Verbose State true.

```
VERBOSE?
```

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
```

---
Command Descriptions

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>;XINC <NR3>;PT_OFF <NR1>;YUNIT 
<Qstring>;YMULT <NR3>;YOFF <NR3>;YZERO<NR3>[<wfm>: 
WFID <Qstring>;NR_PT <NR1>;PT_FMT{ ENV | Y }; 
XUNIT/Qstring>;XINC <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:BN_FMT?

returns either RI or RP as the current waveform data format.
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.
**WFMPre: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 byte to be transmitted first.

**Examples:**

- WFMPRE:BYT_OR MSB specifies that the most significant byte in the waveform data is transferred first.
- WFMPRE:BYT_OR? returns either MSB or LSB depending on which data byte is transferred first.
WFMPRe: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.
WFMPRE: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:

  \[
  X_n = 0 + XINcr \ (n - PT\_Off)
  \]

  \[
  Y_{\text{max}} = YZErO + YMUlt \ (y_{\text{max}} - YOFI)
  \]

  \[
  Y_{\text{min}} = YZErO + YMUlt \ (y_{\text{min}} - YOFI)
  \]

- **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:

  \[
  X_n = 0 + XINcr \ (n - PT\_Off)
  \]

  \[
  Y_n = YZErO + YMUlt \ (y_n - YOFI)
  \]

**Examples:**

- WFMPRE:PT ENV

  sets the waveform data point format to enveloped.
**WFMPRE: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>

```
  WFMPRE  <Space>  PT_Off  <Space>  <NR1>
```

**Arguments:**

<NR1> = 0 to the record length, 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>

```
  WFMPRE  <Space>  XINcr  <Space>  <NR3>
```

**Arguments:**

<NR3> is the sampling interval, in seconds per point.
Command Descriptions

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>

![Diagram of WFMPre:YMUlt]

**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>

![Diagram of WFMPre:YOFf]

**Arguments:** <NR3> is the vertical offset in digitizing levels.

WFMPre:YZEro (No Query Form)

Specifies the offset voltage for the reference waveform specified by the DATa:DESTination command.

**Group:** Waveform

**Syntax:** WFMPre:YZEro <NR3>

![Diagram of WFMPre:YZEro]

**Arguments:** <NR3> is of the offset, in YUNits (usually volts).
Table 2-24 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>
<thead>
<tr>
<th>Command</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFMPre:NR_PT</td>
<td>&lt;NR1&gt;</td>
<td>Number of waveform points</td>
</tr>
<tr>
<td>WFMPre:WFId</td>
<td>&lt;QString&gt;</td>
<td>Waveform identifier</td>
</tr>
<tr>
<td>WFMPre:XUNit</td>
<td>&lt;QString&gt;</td>
<td>Horizontal units</td>
</tr>
<tr>
<td>WFMPre:XMUlt</td>
<td>&lt;NR3&gt;</td>
<td>Horizontal (X-axis) scale factor</td>
</tr>
<tr>
<td>WFMPre:XOFFf</td>
<td>&lt;NR3&gt;</td>
<td>Horizontal (X-axis) offset</td>
</tr>
<tr>
<td>WFMPre:XZERo</td>
<td>&lt;NR3&gt;</td>
<td>Horizontal (X-axis) origin offset</td>
</tr>
<tr>
<td>WFMPre:YUNit</td>
<td>&lt;QString&gt;</td>
<td>Vertical units</td>
</tr>
<tr>
<td>WFMPre:ZMUlt</td>
<td>&lt;NR3&gt;</td>
<td>Z-axis scale factor</td>
</tr>
<tr>
<td>WFMPre:ZOFFf</td>
<td>&lt;NR3&gt;</td>
<td>Z-axis offset</td>
</tr>
<tr>
<td>WFMPre:ZUNit</td>
<td>&lt;QString&gt;</td>
<td>Z-axis units</td>
</tr>
<tr>
<td>WFMPre:ZZERO</td>
<td>&lt;NR3&gt;</td>
<td>Z-axis origin offset</td>
</tr>
</tbody>
</table>

**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 DA:SO command. When sending WFMPRE:<wfm> information to the oscilloscope, the <wfm> specification is ignored and the reference location specified by DA:DEStination is used instead.*

**WFMPre:<wfm>? (Query Only)**

Returns the waveform formatting data for the waveform specified by the DA:SO command. Channel and math waveforms must be displayed before they can be queried. Querying an invalid reference waveform generates an execution error.
Command Descriptions

**Group:** Waveform

**Syntax:** WFMPre:<wfm>?

**Returns:** The format of the response is:

\[
\begin{align*}
<wfm> & : \text{WFID} <\text{Qstring}>; \text{NR_PT} <\text{NR1}>; \text{PT_FMT} \{ \text{ENV} | \text{Y} \}; \\
& \text{XUnit} <\text{QString}>; \text{XINcr} <\text{NR3}>; \text{PT_Off} <\text{NR1}>; \text{YUnit} <\text{QString}>; \text{YMUlt} <\text{NR3}>; \text{YOFF} <\text{NR3}>; \text{YZero} <\text{NR3}>; \\
& \{; <wfm> : \text{WFID} <\text{Qstring}>; \text{NR_PT} <\text{NR1}>; \\
& \text{PT_FMT} \{ \text{ENV} | \text{Y} \}; \text{XUnit} <\text{QString}>; \text{XINcr} <\text{NR3}>; \\
& \text{PT_Off} <\text{NR1}>; \text{YUnit} <\text{QString}>; \text{YMUlt} <\text{NR3}>; \text{YOFF} <\text{NR3}>; \text{YZero} <\text{NR3}> \ldots \}
\end{align*}
\]

---

**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:**

\[
\text{WFMPRe:<wfm>:PT_Fmt \{ ENV | Y \}}
\]

\[
\text{WFMPRe:<wfm>:PT_Fmt?}
\]

**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:

\[
X_n = 0 + XINcr \ (n - \text{PT\_Off})
\]

\[
Y_{n_{\text{max}}} = YZErO + YMUlt \ (y_{n_{\text{max}}} - \text{YOFF})
\]

\[
Y_{n_{\text{min}}} = YZErO + YMUlt \ (y_{n_{\text{min}}} - \text{YOFF})
\]

`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:

\[
X_n = 0 + XINcr \ (n - \text{PT\_Off})
\]

\[
Y_n = YZErO + YMUlt \ (y_n - \text{YOFF})
\]

**Examples:**

\[
\text{WFMPRE:MATH1:PT_FMT?}
\]

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, 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>:WFlId

Returns information about the waveform such as input coupling, volts/division, time/division, acquisition mode, and record length.

The WFMPRe:<wfm>:WFlId command is ignored on input.

**Group:** Waveform

**Syntax:**
WFMPRe:<wfm>:WFlId <QString>
WFMPRe:<wfm>:WFlId?

**Arguments:** <QString> is the waveform identifier string.
WFMPre:<wfm>:XINcr

Sets or queries the horizontal sampling interval. On input, <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

**Group:** Waveform

**Syntax:**

WFMPre:<wfm>:XINcr <NR3>

WFMPre:<wfm>:XINcr?

**Arguments:** <NR3> is the sampling interval.

WFMPre:<wfm>:XUNit

Returns the horizontal (X-axis) units of the waveform data at the time of creation.

The WFMPre:<wfm>:XUNit command is ignored on input.

**Group:** Waveform

**Syntax:**

WFMPre:<wfm>:XUNit <QString>

WFMPre:<wfm>:XUNit?

**Arguments:** <QString> is “s” for seconds, and specifies the units.

**Examples:**

WFMPRE:CH1:XUNIT?

returns “s”, indicating that the horizontal units for channel 1 are seconds.
Command Descriptions

**WFMPre:**<wfm>:YMUlt

Sets or queries the vertical scale factor, in YUNit per unscaled data point value. On input, <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

**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.

**Group:** Waveform

**Syntax:**

WFMPre:<wfm>:YOFf <NR3>

WFMPre:<wfm>:YOFf?

**Arguments:** <NR3> is the position in digitizing levels.
WFMPre:<wfm>:YUNit

Returns the vertical (Y-axis) units of the waveform data at the time of creation. The WFMPre:<wfm>:YUNit command is ignored on input.

**Group:** Waveform

**Syntax:**
WFMPre:<wfm>:YUNit <QString>

WFMPre:<wfm>:YUNit?

**Arguments:**
<QString> is "V" for volts or "VV" for volts², 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.

**Group:** Waveform

**Syntax:**
WFMPre:<wfm>:YZEro <NR3>

WFMPre:<wfm>:YZEro?

**Arguments:**
<NR3> is the offset in YUNits (usually volts).
ZOOm:STATE

Included for compatibility purposes only.

**Group:** Zoom

**Syntax:**

ZOOm:STATE \{ OFF | ON | <NR1> \}

ZOOm:STATE?

---

ZOOm:VERTical:POSition

Sets or queries the vertical position of waveforms.

**NOTE**

ZOOm commands affect only REF and MATH waveforms.

**Group:** Zoom

**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 expansion and compression factor.

**NOTE**

*ZOOM commands affect only REF and MATH waveforms.*

**Group:** Zoom

**Related Commands:** ACQuire:MODE

**Syntax:**

ZOOM:VERTical:SCALE <NR3>

ZOOM:VERTical:SCALE?

**Arguments:**

<NR3> is the amount of vertical expansion or compression.

**Examples:**

ZOOM:VERTICAL:SCALE?

might return :ZOOM::VERTICAL:SCALE 1.0E+0
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

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 (MSB)</td>
<td><strong>PON</strong> (Power On). Shows that the oscilloscope was powered on. The completion of the diagnostic tests also sets this bit.</td>
</tr>
<tr>
<td>6</td>
<td><strong>URQ</strong> (User Request). Shows that an Application menu button was pressed.</td>
</tr>
<tr>
<td>5</td>
<td><strong>CME</strong> (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.</td>
</tr>
<tr>
<td>4</td>
<td><strong>EXE</strong> (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 3-14.</td>
</tr>
<tr>
<td>3</td>
<td><strong>DDE</strong> (Device Error). Shows that a device error occurred. Device error messages are listed in Table 3-6 on page 3-16.</td>
</tr>
<tr>
<td>2</td>
<td><strong>QYE</strong> (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.</td>
</tr>
<tr>
<td>1</td>
<td><strong>RQC</strong> (Request Control). Not used.</td>
</tr>
<tr>
<td>0 (LSB)</td>
<td><strong>OPC</strong> (Operation Complete). Shows that the operation is complete. This bit is set when all pending operations complete following a *OPC command.</td>
</tr>
</tbody>
</table>

The Status Byte Register (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)](image-url)

Status and Events
Table 3-2: SBR Bit Functions

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 (MSB)</td>
<td>Not used.</td>
</tr>
<tr>
<td>6</td>
<td><strong>RQS</strong> (Request Service), obtained from a serial poll. Shows that the oscilloscope requests service from the GPIB controller.</td>
</tr>
<tr>
<td>6</td>
<td><strong>MSS</strong> (Master Status Summary), obtained from *STB?. Summarizes the ESB and MAV bits in the SBR.</td>
</tr>
<tr>
<td>5</td>
<td><strong>ESB</strong> (Event Status Bit). Shows that status is enabled and present in the SESR.</td>
</tr>
<tr>
<td>4</td>
<td><strong>MAV</strong> (Message Available). Shows that output is available in the Output Queue.</td>
</tr>
<tr>
<td>3 – 0</td>
<td>Not used.</td>
</tr>
</tbody>
</table>

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) — is shown in Figure 3-3. This register 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.

![Figure 3-3: The Device Event Status Enable Register (DESER)]
The Event Status Enable Register (ESER) — is shown in Figure 3-4. It 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.

![Figure 3-4: The Event Status Enable Register (ESER)](image)

The Service Request Enable Register (SRER) — is shown in Figure 3-5. It 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 with a Serial Poll (GPIB only) or the MSS bit changes back to a zero.

![Figure 3-5: The Service Request Enable Register (SRER)](image)

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

The oscilloscope stores query responses in the Output Queue. It empties this queue each time it receives a new command or query message after an <EOM>. The controller must read a query response before it sends the next command (or query) or it will lose responses to earlier queries.

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.

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 SERS 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, then take a measurement on the acquired waveform. You could use the following command sequence:

```c
/** Set up single-sequence acquisition **/
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE

/** Acquire waveform data **/
ACQUIRE:STATE ON

/** Set 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:

```c
/* 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, with exceptions noted with the error message text.

Table 3-3 shows the messages when the system has no events or status to report. These have no associated SESR bit.

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No events to report — queue empty</td>
</tr>
<tr>
<td>1</td>
<td>No events to report — new events pending *ESR?</td>
</tr>
</tbody>
</table>

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>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Command error</td>
</tr>
<tr>
<td>102</td>
<td>Syntax error</td>
</tr>
<tr>
<td>103</td>
<td>Invalid separator</td>
</tr>
<tr>
<td>104</td>
<td>Data type error</td>
</tr>
<tr>
<td>105</td>
<td>GET not allowed</td>
</tr>
<tr>
<td>108</td>
<td>Parameter not allowed</td>
</tr>
<tr>
<td>110</td>
<td>Command header error</td>
</tr>
<tr>
<td>111</td>
<td>Header separator error</td>
</tr>
<tr>
<td>112</td>
<td>Program mnemonic too long</td>
</tr>
<tr>
<td>113</td>
<td>Undefined header</td>
</tr>
<tr>
<td>161</td>
<td>Invalid block data (indefinite length blocks are not allowed over the RS-232)</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Execution error</td>
</tr>
<tr>
<td>201</td>
<td>Invalid while in local</td>
</tr>
<tr>
<td>210</td>
<td>Trigger error</td>
</tr>
<tr>
<td>211</td>
<td>Trigger ignored</td>
</tr>
<tr>
<td>212</td>
<td>Arm ignored</td>
</tr>
<tr>
<td>220</td>
<td>Parameter error</td>
</tr>
<tr>
<td>221</td>
<td>Settings conflict</td>
</tr>
<tr>
<td>222</td>
<td>Data out of range</td>
</tr>
<tr>
<td>223</td>
<td>Too much data</td>
</tr>
<tr>
<td>224</td>
<td>Illegal parameter value</td>
</tr>
<tr>
<td>230</td>
<td>Data corrupt or stale</td>
</tr>
<tr>
<td>240</td>
<td>Hardware error</td>
</tr>
<tr>
<td>241</td>
<td>Hardware missing</td>
</tr>
<tr>
<td>242</td>
<td>Hardware configuration error</td>
</tr>
<tr>
<td>243</td>
<td>Hardware I/O device error</td>
</tr>
<tr>
<td>260</td>
<td>Expression error</td>
</tr>
<tr>
<td>261</td>
<td>Math error in expression</td>
</tr>
<tr>
<td>2200</td>
<td>Measurement error, Measurement system error</td>
</tr>
<tr>
<td>2201</td>
<td>Measurement error, Zero period</td>
</tr>
<tr>
<td>2202</td>
<td>Measurement error, No period found</td>
</tr>
<tr>
<td>2203</td>
<td>Measurement error, No period, second waveform</td>
</tr>
<tr>
<td>2204</td>
<td>Measurement error, Low signal amplitude</td>
</tr>
<tr>
<td>2205</td>
<td>Measurement error, Low amplitude, second waveform</td>
</tr>
<tr>
<td>2206</td>
<td>Measurement error, Invalid gate</td>
</tr>
<tr>
<td>2207</td>
<td>Measurement error, Measurement overflow</td>
</tr>
<tr>
<td>2208</td>
<td>Measurement error, Waveform does not cross Mid Ref</td>
</tr>
<tr>
<td>2209</td>
<td>Measurement error, No second Mid Ref crossing</td>
</tr>
</tbody>
</table>
### Table 3-5: Execution Error Messages — EXE Bit 4 (Cont.)

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2210</td>
<td>Measurement error, No Mid Ref crossing, second waveform</td>
</tr>
<tr>
<td>2211</td>
<td>Measurement error, No backwards Mid Ref crossing</td>
</tr>
<tr>
<td>2212</td>
<td>Measurement error, No negative crossing</td>
</tr>
<tr>
<td>2213</td>
<td>Measurement error, No positive crossing</td>
</tr>
<tr>
<td>2214</td>
<td>Measurement error, No crossing</td>
</tr>
<tr>
<td>2215</td>
<td>Measurement error, No crossing, second waveform</td>
</tr>
<tr>
<td>2216</td>
<td>Measurement error, No crossing, target waveform</td>
</tr>
<tr>
<td>2217</td>
<td>Measurement error, Constant waveform</td>
</tr>
<tr>
<td>2218</td>
<td>Measurement error, Unused</td>
</tr>
<tr>
<td>2219</td>
<td>Measurement error, No valid edge – No arm sample</td>
</tr>
<tr>
<td>2220</td>
<td>Measurement error, No valid edge – No arm cross</td>
</tr>
<tr>
<td>2221</td>
<td>Measurement error, No valid edge – No trigger cross</td>
</tr>
<tr>
<td>2222</td>
<td>Measurement error, No valid edge – No second cross</td>
</tr>
<tr>
<td>2223</td>
<td>Measurement error, waveform mismatch</td>
</tr>
<tr>
<td>2224</td>
<td>Measurement error, WAIT calculating</td>
</tr>
<tr>
<td>2225</td>
<td>Measurement error, No waveform to measure</td>
</tr>
<tr>
<td>2226</td>
<td>Null Waveform</td>
</tr>
<tr>
<td>2227</td>
<td>Positive and Negative Clipping</td>
</tr>
<tr>
<td>2228</td>
<td>Measurement error, Positive Clipping</td>
</tr>
<tr>
<td>2229</td>
<td>Measurement error, Negative Clipping</td>
</tr>
<tr>
<td>2230</td>
<td>Measurement error, High Ref &lt; Low Ref</td>
</tr>
<tr>
<td>2235</td>
<td>Math error, Invalid math description</td>
</tr>
<tr>
<td>2240</td>
<td>Invalid password</td>
</tr>
<tr>
<td>2241</td>
<td>Waveform request is invalid</td>
</tr>
<tr>
<td>2242</td>
<td>Data start and stop &gt; record length</td>
</tr>
<tr>
<td>2243</td>
<td>Waveform requested is not a data source</td>
</tr>
<tr>
<td>2244</td>
<td>Waveform requested is not turned on</td>
</tr>
<tr>
<td>2245</td>
<td>Saveref error, Selected channel is turned off</td>
</tr>
<tr>
<td>2246</td>
<td>Saveref error, Selected channel data invalid</td>
</tr>
</tbody>
</table>
### Table 3-5: Execution Error Messages — EXE Bit 4 (Cont.)

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>2248</td>
<td>Saveref error, Source reference data invalid</td>
</tr>
<tr>
<td>2260</td>
<td>Calibration error</td>
</tr>
<tr>
<td>2270</td>
<td>Alias error</td>
</tr>
<tr>
<td>2271</td>
<td>Alias syntax error</td>
</tr>
<tr>
<td>2272</td>
<td>Alias execution error</td>
</tr>
<tr>
<td>2273</td>
<td>Illegal alias label</td>
</tr>
<tr>
<td>2274</td>
<td>Alias parameter error</td>
</tr>
<tr>
<td>2275</td>
<td>Alias definition too long</td>
</tr>
<tr>
<td>2276</td>
<td>Alias expansion error</td>
</tr>
<tr>
<td>2277</td>
<td>Alias redefinition not allowed</td>
</tr>
<tr>
<td>2278</td>
<td>Alias header not found</td>
</tr>
<tr>
<td>2279</td>
<td>Alias label too long</td>
</tr>
<tr>
<td>2280</td>
<td>Alias table full</td>
</tr>
<tr>
<td>2285</td>
<td>Tek Secure® Pass</td>
</tr>
<tr>
<td>2286</td>
<td>Tek Secure® Fail</td>
</tr>
<tr>
<td>2301</td>
<td>Cursor error, Off screen</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>Device-specific error</td>
</tr>
<tr>
<td>310</td>
<td>System error</td>
</tr>
<tr>
<td>311</td>
<td>Memory error</td>
</tr>
<tr>
<td>312</td>
<td>PUD memory lost</td>
</tr>
<tr>
<td>313</td>
<td>Calibration memory lost</td>
</tr>
<tr>
<td>314</td>
<td>Save/recall memory lost</td>
</tr>
<tr>
<td>315</td>
<td>Configuration memory lost</td>
</tr>
<tr>
<td>350</td>
<td>Queue overflow (does not set DDE bit)</td>
</tr>
</tbody>
</table>
Table 3-6: Device Error Messages — DDE Bit 3 (Cont.)

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>361</td>
<td>Parity error in program message (check parity)</td>
</tr>
<tr>
<td>362</td>
<td>Framing error in program message (check baud rate)</td>
</tr>
<tr>
<td>363</td>
<td>Input buffer overrun (check flagging)</td>
</tr>
</tbody>
</table>

Table 3-7 lists the system event messages. These messages are generated whenever certain system conditions occur.

Table 3-7: System Event Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Query event</td>
</tr>
<tr>
<td>401</td>
<td>Power on (PON bit 7 set)</td>
</tr>
<tr>
<td>402</td>
<td>Operation complete (OPC bit 0 set)</td>
</tr>
<tr>
<td>403</td>
<td>User request (URQ bit 6 set)</td>
</tr>
<tr>
<td>404</td>
<td>Power fail (DDE bit 3 set)</td>
</tr>
<tr>
<td>405</td>
<td>Request control</td>
</tr>
<tr>
<td>410</td>
<td>Query INTERRUPTED (QYE bit 2 set)</td>
</tr>
<tr>
<td>420</td>
<td>Query UTERMINATED (QYE bit 2 set)</td>
</tr>
<tr>
<td>430</td>
<td>Query DEADLOCKED (QYE bit 2 set)</td>
</tr>
<tr>
<td>440</td>
<td>Query UTERMINATED after indefinite response (QYE bit 2 set)</td>
</tr>
</tbody>
</table>

Table 3-8 lists warning messages that do not interrupt the flow of command execution. These notify you that you may get unexpected results.

Table 3-8: Execution Warning Messages — EXE Bit 4

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>Execution warning</td>
</tr>
<tr>
<td>510</td>
<td>String data too long, truncated</td>
</tr>
<tr>
<td>525</td>
<td>Parameter underrange</td>
</tr>
<tr>
<td>526</td>
<td>Parameter overflow</td>
</tr>
<tr>
<td>527</td>
<td>Parameter rounded</td>
</tr>
<tr>
<td>528</td>
<td>Parameter out of range</td>
</tr>
<tr>
<td>530</td>
<td>Data stop &gt; stop, Values swapped internally</td>
</tr>
</tbody>
</table>
### Table 3-8: Execution Warning Messages — EXE Bit 4 (Cont.)

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>531</td>
<td>Data stop &gt; record length, Curve truncated</td>
</tr>
<tr>
<td>532</td>
<td>Curve data too long, Curve truncated</td>
</tr>
<tr>
<td>540</td>
<td>Measurement warning</td>
</tr>
<tr>
<td>541</td>
<td>Measurement warning, Low signal amplitude</td>
</tr>
<tr>
<td>542</td>
<td>Measurement warning, Unstable histogram</td>
</tr>
<tr>
<td>543</td>
<td>Measurement warning, Low resolution</td>
</tr>
<tr>
<td>544</td>
<td>Measurement warning, Uncertain edge</td>
</tr>
<tr>
<td>545</td>
<td>Measurement warning, Invalid in minmax</td>
</tr>
<tr>
<td>546</td>
<td>Measurement warning, Need 3 edges</td>
</tr>
<tr>
<td>547</td>
<td>Measurement warning, Clipping positive/negative</td>
</tr>
<tr>
<td>548</td>
<td>Measurement warning, Clipping positive</td>
</tr>
<tr>
<td>549</td>
<td>Measurement warning, Clipping negative</td>
</tr>
</tbody>
</table>

Table 3-9 shows internal errors that indicate an internal fault in the oscilloscope.

### Table 3-9: Internal Warning Messages

<table>
<thead>
<tr>
<th>Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>600</td>
<td>Internal warning</td>
</tr>
</tbody>
</table>
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 GPIBO. 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.
Programming Examples

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-PCI/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:

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-PCI/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:

```bash
cd examples
mkdir examples
```
4. For this installation, you also want to copy `DECL.H` and `MCIB.OBJ` from your Tektronix S3FG210 (National Instruments GPIB-PCI/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:

1. Install QuickBASIC.

2. Install the Tektronix S3FG210 (National Instruments GPIB-PCI/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-PCI/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 .
```
Programming Examples

5. Perform the following two steps for example programs:

1) 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;
   ```

2) 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:

1) 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 GETWF.M.BAS, type:
```
bc /o getwfm.bas;
```

To compile TL.BAS, type:
```
bc /o tl.bas;
```

2) 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 GETWF.M.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: Character Charts

These characters are available for the oscilloscope. Numbers in the lower left corners are character widths in pixels.

#### Table A-1: The TDS 300 Series Character Set

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>∏</td>
<td>space</td>
<td>0</td>
<td>@</td>
<td>P</td>
<td>'</td>
<td>p</td>
</tr>
<tr>
<td>1</td>
<td>⌂</td>
<td>Ω</td>
<td>!</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
</tr>
<tr>
<td>2</td>
<td>⚈</td>
<td>△</td>
<td>&quot;</td>
<td>2</td>
<td>B</td>
<td>R</td>
<td>b</td>
<td>r</td>
</tr>
<tr>
<td>3</td>
<td>Ç</td>
<td>Bw</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
</tr>
<tr>
<td>4</td>
<td>..</td>
<td>∫</td>
<td>$</td>
<td>4</td>
<td>D</td>
<td>T</td>
<td>d</td>
<td>t</td>
</tr>
<tr>
<td>5</td>
<td>\</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>£</td>
<td>µ</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>V</td>
<td>f</td>
<td>v</td>
</tr>
<tr>
<td>7</td>
<td>\</td>
<td>~</td>
<td>7</td>
<td>G</td>
<td>W</td>
<td>g</td>
<td>w</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>i</td>
<td>–</td>
<td>(</td>
<td>8</td>
<td>H</td>
<td>X</td>
<td>h</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>HT</td>
<td>–</td>
<td>)</td>
<td>9</td>
<td>I</td>
<td>Y</td>
<td>i</td>
<td>y</td>
</tr>
<tr>
<td>A</td>
<td>LF</td>
<td>∞</td>
<td>*</td>
<td>J</td>
<td>Z</td>
<td>j</td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>`</td>
<td>ESC</td>
<td>+</td>
<td>;</td>
<td>K</td>
<td>[</td>
<td>k</td>
<td>{</td>
</tr>
<tr>
<td>C</td>
<td>±</td>
<td>±</td>
<td>,</td>
<td>&lt;</td>
<td>L</td>
<td>\</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>D</td>
<td>CR</td>
<td>≠</td>
<td>–</td>
<td>=</td>
<td>M</td>
<td>]</td>
<td>m</td>
<td>}</td>
</tr>
<tr>
<td>E</td>
<td>–</td>
<td>~</td>
<td>.</td>
<td>&gt;</td>
<td>N</td>
<td>^</td>
<td>n</td>
<td>~</td>
</tr>
<tr>
<td>F</td>
<td>•</td>
<td>•</td>
<td>/</td>
<td>?</td>
<td>O</td>
<td>–</td>
<td>o</td>
<td></td>
</tr>
</tbody>
</table>
**Table A-2: The ASCII & GPIB Code Chart**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NUL</td>
<td>DLE</td>
<td>SP</td>
<td>0</td>
<td>60</td>
<td>16</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>120</td>
<td>16</td>
<td>140</td>
<td>0</td>
<td>160</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>GTL</td>
<td>LLD</td>
<td>!</td>
<td>1</td>
<td>A</td>
<td>Q</td>
<td>a</td>
<td>q</td>
</tr>
<tr>
<td></td>
<td>SOH</td>
<td>DC1</td>
<td>1</td>
<td>121</td>
<td>16</td>
<td>141</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>STX</td>
<td>DC2</td>
<td>&quot;</td>
<td>2</td>
<td>R</td>
<td>B</td>
<td>b</td>
<td>r</td>
</tr>
<tr>
<td></td>
<td>ETX</td>
<td>DC3</td>
<td>#</td>
<td>3</td>
<td>C</td>
<td>S</td>
<td>c</td>
<td>s</td>
</tr>
<tr>
<td>3</td>
<td>EOT</td>
<td>SDC</td>
<td>D</td>
<td>4</td>
<td>T</td>
<td>d</td>
<td>t</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENQ</td>
<td>NAK</td>
<td>%</td>
<td>5</td>
<td>E</td>
<td>U</td>
<td>e</td>
<td>u</td>
</tr>
<tr>
<td>4</td>
<td>BEL</td>
<td>ETB</td>
<td>,</td>
<td>6</td>
<td>V</td>
<td>f</td>
<td>f</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>ACK</td>
<td>SYN</td>
<td>&amp;</td>
<td>6</td>
<td>F</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>BS</td>
<td>CAN</td>
<td>(</td>
<td>6</td>
<td>W</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>)</td>
<td>6</td>
<td>g</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>HT</td>
<td>SUB</td>
<td>*</td>
<td>6</td>
<td>X</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>J</td>
<td>6</td>
<td>h</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>LF</td>
<td>ESC</td>
<td>+</td>
<td>6</td>
<td>Y</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Z</td>
<td>6</td>
<td>i</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>VT</td>
<td>FSC</td>
<td>;</td>
<td>6</td>
<td>I</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>J</td>
<td>6</td>
<td>y</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>CR</td>
<td>FS</td>
<td>=</td>
<td>6</td>
<td>M</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td>6</td>
<td>m</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>LF</td>
<td>SUB</td>
<td>*</td>
<td>6</td>
<td>U</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>O</td>
<td>6</td>
<td>^</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>VT</td>
<td>ESC</td>
<td>+</td>
<td>6</td>
<td>M</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td>6</td>
<td>m</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>FF</td>
<td>FS</td>
<td>&lt;</td>
<td>6</td>
<td>L</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>I</td>
<td>6</td>
<td>~</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>CR</td>
<td>GS</td>
<td>=</td>
<td>6</td>
<td>M</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td>6</td>
<td>m</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>SO</td>
<td>RS</td>
<td>&gt;</td>
<td>6</td>
<td>N</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>N</td>
<td>6</td>
<td>~</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>SI</td>
<td>US</td>
<td>/</td>
<td>6</td>
<td>O</td>
<td>102</td>
<td>76</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>o</td>
<td>6</td>
<td></td>
<td>76</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**KEY**
- **octal**: 25
- **hex**: 15
- **PPU**: NAK
- **GPIB code**: ASCII character
- **decimal**: 15

**ADDRESSED COMMANDS**

- **UNIVERSAL COMMANDS**
- **LISTEN ADDRESSES**
- **TALK ADDRESSES**
- **SECONDARY ADDRESSES OR COMMANDS**

**A-2**

**Appendices**
Appendix B: Reserved Words

The following is a list of the reserved words of the digitizing oscilloscope. Do not use these words for aliases.

*CAL COUPLing  HEADer  PERSistence  TARget
*CLS CURSor  HIGH  PORT  TEKSecure
*DDT CURSOR1  HOLdoff  POSITION  TERMinator
*ESE CURSOR2  HORizontal  POSITION1  TEXt
*ESR CURVe  HPOS1  POSITION2  TIME
*IDN DATa  HPOS2  PRESet  TRANSmit
*IST DCD  ID  PREss  TRIGger
*LRN DEFINE  IMMEd  PRObe  TRIGT
*OPC DELay  INTENSIty  PT_FMT  TYPe
*PSC DELTe  INVert  PT_OFF  UNIts
*PUD DELTa  LAYOUT  RECALL  UNLOCK
*RCL DESE  LEVEL  RECORDlength  VALUE
*RST DESTination  LOCK  REF1  VBAr
*SRE DEViCE  LOW  REF2  VDElt
*STB DIAg  LOW  REFLevel  VERBo
*TRG DISplay  MAIn  REM  VERTical
*TST ENCdg  MATH  RESULT  VIDeo
*WAI EVEnt  MATH1  RFR  VOLts
ABSolute  EVMsg  MEAS1  RS232  WAVEFORM
ACQuire  EVQty  MEAS2  RTS  WAFVrm
Allas  EXT  MEAS3  RUNSAfter  WIFd
ALl  EXT10  MEAS4  SAVE  WFMPre
ALL  EXT100  MEASUREment  SBITS  WIDth
ALLEV  FACTory  METHOD  SCALE  XINcr
ALWAYS  FALL  MID  SECdiv  XMUlt
AUTOSet  FIELD1  MODE  SELECT  XOFI
BANwidth  FIELD2  NEWpass  SETUP  XON
BAUD  FLAg  NR_Pt  SLOpe  XUNit
BIT_Nr  FORMat  NUMAcq  SOFTFlagging  XZEr
BN_Fmt  FPAnel  NUMAvg  SOURCE  YMUlt
BUSY  FUNCTION  NUMEnv  SOURCE1  YOFF
BYT_NR  GAting  OFFSet  START  YUNit
BYT_Or  GRAticule  OVERAll  STATE  YZEr
CALibrate  HARDCopy  PACE  STATUS  ZMUlt
CATalog  HARDFlagging  PAIred  STOP  ZOFI
CH1  HBArs  PArity  STOPAfter  ZOOn
CH2  HDELTa  PASSWord  STOPBits  ZUNit
CONTRast  HDR  PERCent  STYle  ZZEro
CONTROI
Appendix B: Reserved Words
Appendix C: Interface Specifications

This appendix describes details of the GPIB remote interface of the oscilloscope. Normally, you will not need this information to use the oscilloscope, but the information is useful when connecting to controllers of unusual configuration.

GPIB Function Subsets

The oscilloscope supports many GPIB function subsets, as listed 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 A-3 shows the standard interface messages that the oscilloscope supports.

<table>
<thead>
<tr>
<th>Message</th>
<th>GPIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCL</td>
<td>Yes</td>
</tr>
<tr>
<td>GET</td>
<td>Yes</td>
</tr>
<tr>
<td>GTL</td>
<td>Yes</td>
</tr>
<tr>
<td>LLO</td>
<td>Yes</td>
</tr>
<tr>
<td>PPC</td>
<td>No</td>
</tr>
<tr>
<td>PPD</td>
<td>No</td>
</tr>
<tr>
<td>PPE</td>
<td>No</td>
</tr>
<tr>
<td>PPU</td>
<td>No</td>
</tr>
<tr>
<td>SDC</td>
<td>Yes</td>
</tr>
<tr>
<td>SPD</td>
<td>Yes</td>
</tr>
<tr>
<td>SPE</td>
<td>Yes</td>
</tr>
<tr>
<td>TCT</td>
<td>No</td>
</tr>
<tr>
<td>UNL</td>
<td>Yes</td>
</tr>
<tr>
<td>UNT</td>
<td>Yes</td>
</tr>
<tr>
<td>Listen Addresses</td>
<td>Yes</td>
</tr>
<tr>
<td>Talk Addresses</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Appendix D: Factory Initialization Settings

The factory initialization settings provide a known state for the oscilloscope. Factory initialization sets values as shown in Table A-4.

<table>
<thead>
<tr>
<th>Control</th>
<th>Changed by Factory Init to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquire mode</td>
<td>Sample</td>
</tr>
<tr>
<td>Acquire stop after</td>
<td>RUN/STOP button only</td>
</tr>
<tr>
<td>Acquire # of averages</td>
<td>16</td>
</tr>
<tr>
<td>Acquire # of envelopes</td>
<td>8</td>
</tr>
<tr>
<td>Channel selection</td>
<td>Channel 1 on, all others off</td>
</tr>
<tr>
<td>Cursor H Bar 1 position</td>
<td>+3.2 divisions from the center</td>
</tr>
<tr>
<td>Cursor H Bar 2 position</td>
<td>– 4 divisions from the center</td>
</tr>
<tr>
<td>Cursor V Bar 1 position</td>
<td>10% of the graticule</td>
</tr>
<tr>
<td>Cursor V Bar 2 position</td>
<td>90% of the graticule</td>
</tr>
<tr>
<td>Cursor function</td>
<td>Off</td>
</tr>
<tr>
<td>Cursor time units</td>
<td>Seconds</td>
</tr>
<tr>
<td>Delay time, delayed runs after main</td>
<td>50 µs</td>
</tr>
<tr>
<td>Delayed, time base mode</td>
<td>Delayed Runs After Main</td>
</tr>
<tr>
<td>Display format</td>
<td>YT</td>
</tr>
<tr>
<td>Display graticule type</td>
<td>Full</td>
</tr>
<tr>
<td>Display intensity – contrast</td>
<td>150%</td>
</tr>
<tr>
<td>Display intensity – text</td>
<td>DIM</td>
</tr>
<tr>
<td>Display intensity – waveform</td>
<td>BRIGHT</td>
</tr>
<tr>
<td>Display intensity – overall</td>
<td>85%</td>
</tr>
<tr>
<td>Display interpolation filter</td>
<td>Sin(x)/x</td>
</tr>
<tr>
<td>Display style</td>
<td>Vectors</td>
</tr>
<tr>
<td>Display trigger “T”</td>
<td>On</td>
</tr>
<tr>
<td>Display variable persistence</td>
<td>500 ms</td>
</tr>
</tbody>
</table>
### Table A-4: Factory Initialization Defaults (Cont.)

<table>
<thead>
<tr>
<th>Control</th>
<th>Changed by Factory Init to</th>
</tr>
</thead>
<tbody>
<tr>
<td>Edge trigger coupling</td>
<td>DC</td>
</tr>
<tr>
<td>Edge trigger level</td>
<td>0.0 V</td>
</tr>
<tr>
<td>Edge trigger slope</td>
<td>Rising</td>
</tr>
<tr>
<td>Edge trigger source</td>
<td>Channel 1</td>
</tr>
<tr>
<td>Horizontal – main trigger position</td>
<td>50%</td>
</tr>
<tr>
<td>Horizontal – main trigger time/div.</td>
<td>500 µs</td>
</tr>
<tr>
<td>Horizontal – time base</td>
<td>Main only</td>
</tr>
<tr>
<td>Main trigger holdoff</td>
<td>20%</td>
</tr>
<tr>
<td>Main trigger mode</td>
<td>Auto</td>
</tr>
<tr>
<td>Main trigger type</td>
<td>Edge</td>
</tr>
<tr>
<td>Math waveform function</td>
<td>CH1 + CH2</td>
</tr>
<tr>
<td>Measure High-Low Setup</td>
<td>Histogram</td>
</tr>
<tr>
<td>Measure High Ref</td>
<td>90% and 0 V (units)</td>
</tr>
<tr>
<td>Measure Low Ref</td>
<td>10% and 0 V (units)</td>
</tr>
<tr>
<td>Measure Mid Ref</td>
<td>50% and 0 V (units)</td>
</tr>
<tr>
<td>Saved setups</td>
<td>No change</td>
</tr>
<tr>
<td>Saved waveforms</td>
<td>No change</td>
</tr>
<tr>
<td>Vertical bandwidth (all channels)</td>
<td>Full</td>
</tr>
<tr>
<td>Vertical coupling (all channels)</td>
<td>DC</td>
</tr>
<tr>
<td>Vertical offset (all channels)</td>
<td>0 V</td>
</tr>
<tr>
<td>Vertical position (all channels)</td>
<td>0 div</td>
</tr>
<tr>
<td>Vertical volts/div. (all channels)</td>
<td>100 mV/div</td>
</tr>
<tr>
<td>Zoom vertical (all channels)</td>
<td>1.0X</td>
</tr>
<tr>
<td>Zoom vertical position (all channels)</td>
<td>0 divs.</td>
</tr>
</tbody>
</table>
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

TEKSecure
A Tektronix custom command that initializes both waveform and setup memories. This overwrites any previously stored data.
Index

A

Abbreviating, command, 2-4
ACQUIRE?, 2-27
ACQUIRE:MODE, 2-28
ACQUIRE:NUMACQ?, 2-29
ACQUIRE:NUMAVG, 2-30
ACQUIRE:NUMENV, 2-31
ACQUIRE:STATE, 2-32
ACQUIRE:STOPAFTER, 2-33
Acquisition command group, 2-11
Acquisition commands
ACQUIRE?, 2-27
ACQUIRE:MODE, 2-28
ACQUIRE:NUMACQ?, 2-29
ACQUIRE:NUMAVG, 2-30
ACQUIRE:NUMENV, 2-31
ACQUIRE:STATE, 2-32
ACQUIRE:STOPAFTER, 2-33
Address, Definition of, G-1
ALIAS, 2-34
Alias commands
ALIAS, 2-34
ALIAS:CATALOG?, 2-35
ALIAS:DEFINE, 2-36
ALIAS:DELETE, 2-37
ALIAS:DELETE:ALL, 2-37
ALIAS:DELETE:NAME, 2-38
ALIAS:STATE, 2-38
ALIAS:CATALOG?, 2-35
ALIAS:DEFINE, 2-36
ALIAS:DELETE, 2-37
ALIAS:DELETE:ALL, 2-37
ALIAS:DELETE:NAME, 2-38
ALIAS:STATE, 2-38
ALLEV?, 2-39
Argument, command, 2-2
ASCII, 2-1, G-1
AUTOSET, 2-40

B

Backus-Naur Form, Definition of, G-1
Block, command argument, 2-9
BNF, G-1
BNF (Backus-Naur form), 2-1
Break, 2-4
BUSY?, 2-41

C

*CAL?, 2-42
CALIBRATE, 2-43
Calibration and diagnostic command group, 2-12
Calibration and diagnostic commands
*CAL?, 2-42
CALIBRATE, 2-43
DIAG:RESULT:FLAG?, 2-73
DIAG:RESULT:LOG?, 2-73
DIAG:SELECT:ALL, 2-74
DIAG:STATE, 2-74
Caution statements, vii
CH<x>?, 2-43
CH<x>:BANDWIDTH, 2-44
CH<x>:COUPLING, 2-45
CH<x>:OFFSET, 2-46
CH<x>:POSITION, 2-47
CH<x>:PROBE?, 2-47
CH<x>:SCALE, 2-48
CH<x>:VOLTS, 2-49
Channel, command mnemonic, 2-6
CH<x>, command mnemonic, 2-6
Clear Status, 2-50
CLEARMENU, 2-49
*CLS, 2-50

Command
Abbreviating, 2-4
Argument, 2-2
Block argument, 2-9
Common, 2-17, 2-19
Concatenating, 2-4
Header, 2-2
Message, 2-2
Mnemonic, 2-2
Query, 2-1
Rules for forming, 2-1
Separator, 2-2
Set, 2-1
Syntax, 2-1
BNF (Backus-Naur form), 2-1
Command argument
Numeric, 2-7
Quoted string, 2-7
Command group
Acquisition, 2-11
Calibration and diagnostic, 2-12
Cursor, 2-12
Display, 2-13
Hardcopy, 2-14
Horizontal, 2-14
Measurement, 2-15
Miscellaneous, 2-17
Save and recall, 2-18
Status and error, 2-19
Trigger, 2-19
Vertical, 2-20
Waveform, 2-22
Command mnemonic
Channel, 2-6
CH<x>, 2-6
Cursor position, 2-6
Math waveform, 2-6
MATH<x>., 2-6
Measurement specifier, 2-6
MEAS<x>., 2-6
POSITION<x>., 2-6
Reference waveform, 2-7
REF<x>., 2-7
Waveform, 2-7
<wfm>., 2-7
Command syntax, 1-1, 2-1
BNF (Backus-Naur form), 2-1
Common command, 2-17, 2-19

TDS 310, TDS 320 & TDS 350 Programmer Manual
I-1
Index

Common GPIB commands
*CAL?, 2-42
*CLS, 2-50

Concatenating, command, 2-4
Configuration, Command query, 2-106
Connecting to a GPIB Device, 1-4
Connecting to an RS-232 Device, 1-7
Controller, Definition of, G-1
Cursor command group, 2-12

Cursor commands
CURSOR?, 2-50
CURSOR:FUNCTION, 2-51
CURSOR:HBARS?, 2-51
CURSOR:HBARS:DELTA?, 2-52
CURSOR:HBARS:POSITION<x>, 2-52
CURSOR:HBARS:SELECT, 2-53
CURSOR:PAIRED:HDELTA, 2-54
CURSOR:PAIRED:HPOS1, 2-54
CURSOR:PAIRED:HPOS2, 2-55
CURSOR:PAIRED:POSITION1, 2-55
CURSOR:PAIRED:POSITION2, 2-56
CURSOR:PAIRED:SELECT, 2-57
CURSOR:PAIR:VDELTA, 2-57
CURSOR:VBARS, 2-58
CURSOR:VBARS:DELTA?, 2-59
CURSOR:VBARS:POSITION<x>, 2-59
CURSOR:VBARS:SELECT, 2-60
CURSOR:VBARS:UNITS, 2-61
CURVE, 2-62

D

DATA, 2-63
DATA:DESTINATION, 2-64
DATA:ENCODG, 2-64
DATA:SOURCE, 2-66
DATA:START, 2-67
DATA:STOP, 2-68
DATA:TARGET, 2-69
DATA:WIDTH, 2-70
DCL, A-6
*DDT, 2-71
DESE command, 2-72, 3-3
DESER register, 2-72, 2-131, 3-3
Device Clear, 2-4, A-6
DIAG:RESULT:FLAG?, 2-73
DIAG:RESULT:LOG?, 2-73
DIAG:SELECT:ALL, 2-74
DIAG:STATE, 2-74
Diagram, syntax, 2-10
Display command group, 2-13
Display commands
CLEARMENU, 2-49
DISPLAY?, 2-75
DISPLAY:FORMAT, 2-76
DISPLAY:GRATICULE, 2-77
DISPLAY:INTENSITY?, 2-77
DISPLAY:INTENSITY:CONTRAST, 2-78
DISPLAY:INTENSITY:OVERALL, 2-78
DISPLAY:INTENSITY:TEXT, 2-79
DISPLAY:INTENSITY:WAVEFORM, 2-80
DISPLAY:PERSISTENCE, 2-81
DISPLAY:STYLE, 2-82
DISPLAY:TRIGT, 2-83

DISPLAY?, 2-75
DISPLAY:FORMAT, 2-76
DISPLAY:GRATICULE, 2-77
DISPLAY:INTENSITY?, 2-77
DISPLAY:INTENSITY:CONTRAST, 2-78
DISPLAY:INTENSITY:OVERALL, 2-78
DISPLAY:INTENSITY:TEXT, 2-79
DISPLAY:INTENSITY:WAVEFORM, 2-80
DISPLAY:PERSISTENCE, 2-81
DISPLAY:STYLE, 2-82
DISPLAY:TRIGT, 2-83

E

Electric overload, viii
End or Identify, G-1
EOI, G-1
EOM, G-1
EOM (end of message), 2-5
Error message, programming interface, 3-13
*ESE, 2-84, 3-4
ESER register, 2-84, 2-131, 3-4
*ESR?, 2-85
*ESR? query, 3-1
Event handling, 3-1, 3-6
Event query, 2-85, 2-86
Event queue, 2-85, 2-86, 3-5
EVENT?, 2-85
EVMSG?, 2-86
EVQTY?, 2-87
Example Programs, 4-1

F

FACTORY, 2-88
Factory initialization settings, A-7–A-8
Fuse, viii
G

GET, A-6
Go to local, A-6

GPIB, 1-4, G-1
  Compared to the RS-232, 1-3
  Configurations, 1-4
  Connecting to, 1-4
  Connection rules, 1-4
  EOM (end of message), 2-5
  Function subsets, A-5

Grounding, viii
Group execute trigger, A-6
GTL, A-6

H

Hardcopies, 1-12
HARDCOPY, 2-88
Hardcopy command group, 2-14
Hardcopy commands
  HARDCOPY, 2-88
  HARDCOPY:FORMAT, 2-90
  HARDCOPY: LAYOUT, 2-91
  HARDCOPY:PORT, 2-92
  HARDCOPY:FORMAT, 2-90
  HARDCOPY: LAYOUT, 2-91
  HARDCOPY:PORT, 2-92
HARDCOPY:PORT, 2-92
HARDCOPY:PORT, 2-92
HEADER, 2-93
Header
  Command, 2-2, 2-93
  Included in query response, 2-93, 2-163
Horizontal command group, 2-14
Horizontal commands
  HORIZONTAL?, 2-94
  HORIZONTAL:DELAY?, 2-94
  HORIZONTAL:DELAY:SCALE, 2-96
  HORIZONTAL:DELAY:SECDIV, 2-97
  HORIZONTAL:DELAY:TIME, 2-98
  HORIZONTAL:DELAY:TIME?, 2-99
  HORIZONTAL:DELAY:TIME:RUNS--AFTER, 2-99
  HORIZONTAL:MAIN?, 2-100
  HORIZONTAL:MAIN:SCALE, 2-100
  HORIZONTAL:MAIN:SECDIV, 2-101
  HORIZONTAL:MODE, 2-102
  HORIZONTAL:POSITION, 2-103
  HORIZONTAL:SCALE, 2-104
  HORIZONTAL:SECDIV, 2-104
  HORIZONTAL:TRIGGER?, 2-104
  HORIZONTAL:TRIGGER:POSITION, 2-105

I

ID?, 2-106
*IDN?, 2-106
IEEE, G-1
IEEE Std. 488.2-1987, 1-4, 2-1, 2-17, 2-19
Interface message, A-6

L

LLO, A-6
Local lock out, A-6
LOCK, 2-107
*LRN?, 2-108

M

Manual trigger, Simulation with command, 2-150
Math waveform, command mnemonic, 2-6
MATH<xx>, 2-108
MATH<xx>:DEFINE, 2-109
MATH<xx>, command mnemonic, 2-6
Measurement command group, 2-15
Measurement commands
  MEASUREMENT?, 2-110
  MEASUREMENT:GATING, 2-111
  MEASUREMENT:IMMED?, 2-111
  MEASUREMENT:IMMED: SOURCE1, 2-112
  MEASUREMENT:IMMED:TYPE, 2-112
  MEASUREMENT:IMMED:UNITS?, 2-115
  MEASUREMENT:IMMED:VALUE?, 2-115
  MEASUREMENT:MEAS<xx>?, 2-116
  DELAY:SOURCE1, 2-116
  MEASUREMENT:MEAS<xx>: STATE, 2-117
  MEASUREMENT:MEAS<xx>: TYPE, 2-117
  MEASUREMENT:MEAS<xx>: UNITS?, 2-120
  MEASUREMENT:MEAS<xx>: VALUES?, 2-120
  MEASUREMENT:REFLEVEL?, 2-121
  MEASUREMENT:REFLEVEL: ABSOLUTE:HIGH, 2-122
  MEASUREMENT:REFLEVEL: ABSOLUTE:LOW, 2-123
  MEASUREMENT:REFLEVEL: METHOD, 2-125
  MEASUREMENT:REFLEVEL: PERCENT:HIGH, 2-126
  MEASUREMENT:REFLEVEL: PERCENT:LOW, 2-127
  MEASUREMENT:REFLEVEL: PERCENT:经开, 2-128
  MEASUREMENT:REFLEVEL: ABSOLUTE:LOW, 2-123
Measurement specifier, command mnemonic, 2-6
MEASUREMENT?, 2-110
MEASUREMENT:GATING, 2-111
MEASUREMENT:IMMED?, 2-111
**Index**

| MEASUREMENT:IMMED: SOURCE1, 2-112 |
| MEASUREMENT:IMMED:TYPE, 2-112 |
| MEASUREMENT:IMMED:UNITS?, 2-115 |
| MEASUREMENT:IMMED:VALUE?, 2-115 |
| MEASUREMENT:MEAS<x>?, 2-116 |
| MEASUREMENT:MEAS<x>: DELAY:SOURCE1, 2-116 |
| MEASUREMENT:MEAS<x>: STATE, 2-117 |
| MEASUREMENT:MEAS<x>: TYPE, 2-117 |
| MEASUREMENT:MEAS<x>: UNITS?, 2-120 |
| MEASUREMENT:MEAS<x>: VALUES?, 2-120 |
| MEASUREMENT:METHOD, 2-121 |
| MEASUREMENT:REFLEVEL?, 2-121 |
| MEASUREMENT:REFLEVEL:ABSOLUTE:HIGH, 2-122 |
| MEASUREMENT:REFLEVEL:ABSOLUTE:LOW, 2-123 |
| MEASUREMENT:REFLEVEL:ABSOLUTE:MID, 2-124 |
| MEASUREMENT:REFLEVEL:METHOD, 2-125 |
| MEASUREMENT:REFLEVEL:PERCENT:HIGH, 2-126 |
| MEASUREMENT:REFLEVEL:PERCENT:LOW, 2-127 |
| MEASUREMENT:REFLEVEL:PERCENT:MID, 2-128 |
| MEAS<x>, command mnemonic, 2-6 |
| Message Command, 2-2 |
| Command terminator, 2-5 |
| Handling, 3-1 |
| Table of program messages, 3-13 |
| Miscellaneous, LOCK, 2-107 |
| Miscellaneous command group, 2-17 |
| Miscellaneous commands AUTOSET, 2-40 |
| *DDT, 2-71 |
| FACTORY, 2-88 |
| HDR, 2-92 |
| HEADER, 2-93 |
| *IDN?, 2-106 |

**P**

*LRN?, 2-108 |
NEWPASS, 2-128 |
PASSWORD, 2-130 |
PUD, 2-132 |
REM, 2-133 |
RS232?, 2-134 |
RS232:BAUD, 2-135 |
RS232:CONTROL:RTS, 2-135 |
RS232:DCD, 2-136 |
RS232:HARDFlagging, 2-136 |
RS232:MODE, 2-137 |
RS232:PACE, 2-138 |
RS232:PARITY, 2-138 |
RS232:PRESet, 2-139 |
RS232:SBITS, 2-140 |
RS232:SOFTFlagging, 2-140 |
RS232:STOPBITS, 2-141 |
RS232:TRANsmit:DELay, 2-142 |
RS232:TRANsmit:TERMinator, 2-142 |
SET, 2-147 |
TEKSECURE, 2-149 |
*TRG, 2-150 |
UNLOCK, 2-162 |
VERBOSE, 2-163 |

Mnemonic, command, 2-2

**Q**

Query, Header in query response, 2-93, 2-163 |
Query command, 2-1 |
Queue Event, 3-5 |
Output, 3-5 |
QuickBASIC, G-1 |
QuickC, G-1 |
Quoted string, command argument, 2-7

**R**

*OFC, 2-131 |
*PSC, 2-131 |
PSC command, 3-4 |
PUD, 2-132 |

**N**

NEWPASS, 2-128 |

Numeric, command argument, 2-7

**O**

*OPC, 2-129 |

Operation, in explosive atmospheres, viii |
Operation complete command, 2-129 |
Operation complete wait, 2-163 |
Oscilloscope, grounding, viii |
Output queue, 3-5 |
Overload, electric, viii

**R**

*RCL, 2-132 |

Recall setting command, 2-132 |
RECALL:SETUP, 2-133 |
Reference waveform, command mnemonic, 2-7 |
REF<x>, command mnemonic, 2-7
Safe

specific precautions, viii
summary, vii
symbols and terms, vii

*S, 2-143

Save and recall command group, 2-18

Save and recall commands
*RCL, 2-132
*SAV, 2-143

SAVE: SETUP, 2-144
SAVE: WAVEFORM, 2-144

Save setting command, 2-143

SAVE: SETUP, 2-144
SAVE: WAVEFORM, 2-144

SBR register, 2-149, 3-2

SDC, A-6

SELECT?, 2-145
SELECT: <wfm>, 2-145
SELECT: CONTROL?, 2-146

Selected device clear, A-6

Self test, 2-162

Separator, command, 2-2

Serial poll, 3-2

Disable, A-6

Enable, A-6

Service request enable command, 2-148

Service request enable register, 2-148

Service Requests, 1-12

SESR register, 2-50, 2-85, 2-129, 3-1

Set command, 2-1

SET?, 2-147

Setting

Command query, 2-108
Query, 2-108
Recall command, 2-132
Save command, 2-143

Shock hazards, viii

SPD, A-6

SPE, A-6

*SRE command, 2-148, 3-4

SRER register, 2-131, 2-148, 3-4

Status, 3-1

Status and error command group, 2-19

Status and Error commands

EVENT?, 2-85

*WAI, 2-163

Status and error commands

ALLEV?, 2-39

BUSY?, 2-41

*CLS, 2-50

DESE, 2-72, 3-3

*ESE, 2-84, 3-4

*ESR?, 2-85, 3-1

EVMSG?, 2-86

EVQTY?, 2-87

ID?, 2-106

*OPC, 2-129

*PSC, 2-131, 3-4

*RST, 2-134

*SRE, 2-148, 3-4

*STB?, 2-149, 3-2

*TST?, 2-162

Status and Events, 1-2

*STB?, 2-149

*STB? query, 3-2

Symbols and terms, vii

Syntax

BNF (Backus-Naur form), 2-1

Command, 2-1

Diagram, 2-10

T

Table, programming message, 3-13

TCT, A-6

Tek Std. Codes and Formats 1989, 2-17, 2-19

TEKSECURE, 2-149

TEKSecure, G-1

Terminator, command message, 2-5

Time base, Manual trigger simulation, 2-150

*TRG, 2-150

TRIGGER, 2-150

Trigger command group, 2-19

Trigger commands

TRIGGER, 2-150

TRIGGER:MAIN, 2-151

TRIGGER:MAIN:EDGE?, 2-151
Index

TRIGGER:MAIN:EDGE:COUPLING, 2-152
TRIGGER:MAIN:EDGE:SLOPE, 2-153
TRIGGER:MAIN:EDGE:SOURCE, 2-154
TRIGGER:MAIN:HOLDOFF?, 2-154
TRIGGER:MAIN:HOLDOFF:VALUE, 2-155
TRIGGER:MAIN:LEVEL, 2-155
TRIGGER:MAIN:MODE, 2-156
TRIGGER:MAIN:TYPE, 2-157
TRIGGER:MAIN:VIDEO:FIELD, 2-158
TRIGGER:MAIN:VIDEO:HOLDOFF?, 2-158
TRIGGER:MAIN:VIDEO:HOLDOFF:VALUE, 2-159
TRIGGER:MAIN:VIDEO:SCAN, 2-160
TRIGGER:MAIN:VIDEO:SOURCE, 2-160
TRIGGER:STATE?, 2-161
TRIGGER:MAIN, 2-151
TRIGGER:MAIN:EDGE?, 2-151
TRIGGER:MAIN:EDGE:COUPLING, 2-152
TRIGGER:MAIN:EDGE:SLOPE, 2-153
TRIGGER:MAIN:EDGE:SOURCE, 2-154
TRIGGER:MAIN:HOLDOFF?, 2-154
TRIGGER:MAIN:HOLDOFF:VALUE, 2-155
TRIGGER:MAIN:LEVEL, 2-155
TRIGGER:MAIN:MODE, 2-156
TRIGGER:MAIN:TYPE, 2-157
TRIGGER:MAIN:VIDEO:FIELD, 2-158
TRIGGER:MAIN:VIDEO:HOLDOFF?, 2-158
TRIGGER:MAIN:VIDEO:HOLDOFF:VALUE, 2-159
TRIGGER:MAIN:VIDEO:SCAN, 2-160
TRIGGER:MAIN:VIDEO:SOURCE, 2-160
TRIGGER:STATE?, 2-161
*TST? query, 2-162

U

UNL, A-6
Unlisten, A-6
UNLOCK, 2-162
UNT, A-6
Untalk, A-6

V

VERBOSE, 2-163
Vertical
MATH<x>?, 2-108
MATH<x>:DEFINE, 2-109
Vertical bar cursors, 2-58
Vertical command group, 2-20
Vertical commands
CH<x>?, 2-43
CH<x>:BANDWIDTH, 2-44
CH<x>:COUPLING, 2-45
CH<x>:OFFSET, 2-46
CH<x>:POSITION, 2-47
CH<x>:PROBE?, 2-47
CH<x>:SCALE, 2-48
CH<x>:VOLTS, 2-49
SELECT?, 2-145
SELECT:<wfm>, 2-145
SELECT:CONTROL?, 2-146

W

*WAI, 2-163
Wait for operation complete, 2-163
Warning statements, vii
Waveform, command mnemonic, 2-7
Waveform command group, 2-22
Waveform commands
CURVE, 2-62
DATA, 2-63
DATA:DESTINATION, 2-64
DATA:ENCODG, 2-64
DATA:SOURCE, 2-66
DATA:START, 2-67
DATA:STOP, 2-68
DATA:TARGET, 2-69
DATA:WIDTH, 2-70
WAVFRM?, 2-164
WFMPRE?, 2-164
WFMPRE:<wfm>?, 2-173
WFMPRE:<wfm>:NR_PT, 2-174
WFMPRE:<wfm>:PT_FMT, 2-174
WFMPRE:<wfm>:PT_OFF, 2-176
WFMPRE:<wfm>:WFID, 2-176
WFMPRE:<wfm>:XINC, 2-177
WFMPRE:<wfm>:XUNIT, 2-177
WFMPRE:<wfm>:YMULT, 2-178
WFMPRE:<wfm>:YOFF, 2-178
WFMPRE:<wfm>:YZERO, 2-179
WFMPRE:BIT_NR, 2-165
WFMPRE:BN_FMT, 2-166
WFMPRE:BYT_NR, 2-167
WFMPRE:BYT_OR, 2-168
WFMPRE:ENCODG, 2-169
WFMPRE:NR_PT, 2-173
WFMPRE:PT_FMT, 2-170
WFMPRE:PT_OFF, 2-171
WFMPRE:WFID, 2-173
WFMPRE:XINC, 2-171
WFMPRE:XMULT, 2-173
WFMPRE:YOFF, 2-173
WFMPRE:XUNIT, 2-173
WFMPRE:ZERO, 2-173
WFMPRE:YUNIT, 2-173
WFMPRE:ZMULT, 2-173
WFMPRE:ZOFF, 2-173
WFMPRE:ZUNIT, 2-173
WFMPRE:ZZERO, 2-173

Glossary and Index
WFMPRE:BYT_NR, 2-167
WFMPRE:BYT_OR, 2-168
WFMPRE:ENCDBG, 2-169
WFMPRE:NR_PT, 2-173
WFMPRE:PT_FMT, 2-170
WFMPRE:PT_OFF, 2-171
WFMPRE:WFID, 2-173
WFMPRE:XINC, 2-171
WFMPRE:XMULT, 2-173
WFMPRE:XOFF, 2-173
WFMPRE:XUNIT, 2-173
WFMPRE:XZERO, 2-173
WFMPRE:YMULT, 2-172
WFMPRE:YOFF, 2-172
WFMPRE:YUNIT, 2-173
WFMPRE:YZERO, 2-172
WFMPRE:ZMULT, 2-173
WFMPRE:ZOFF, 2-173
WFMPRE:ZUNIT, 2-173
WFMPRE:ZZERO, 2-173

Z

Zoom commands

ZOOM:STATE, 2-180
ZOOM:VERTICAL:POSITION, 2-180
ZOOM:VERTICAL:SCALE, 2-181

ZOOM:STATE, 2-180
ZOOM:VERTICAL:POSITION, 2-180
ZOOM:VERTICAL:SCALE, 2-181