Programmer Manual

Tektronix

TDS Family
Includes TDS 420, 460, 520, 540, 620, 640
Digitizing Oscilloscopes

070-8318-05

This manual supports the following firmware:
TDS 420 & 460, Version 2
TDS 520 & 540, Version 2
TDS 620 & 640, Version 2

Please check for change information at the rear of this manual.

First Printing: December 1992
Revised Printing: January 1993

Instrument Serial Numbers

Each instrument manufactured by Tektronix has a serial number on a panel insert or tag, or stamped on the chassis. The first letter in the serial number designates the country of manufacture. The last five digits of the serial number are assigned sequentially and are unique to each instrument. Those manufactured in the United States have six unique digits. The country of manufacture is identified as follows:

B010000	Tektronix, Inc., Beaverton, Oregon, USA
E200000	Tektronix United Kingdom, Ltd., London
J300000	Sony/Tektronix, Japan
H700000	Tektronix Holland, NV, Heerenveen, The Netherlands

Instruments manufactured for Tektronix by external vendors outside the United States are assigned a two digit alpha code to identify the country of manufacture (e.g., JP for Japan, HK for Hong Kong, IL for Israel, etc.).

Tektronix, Inc., P.O. Box 500, Beaverton, OR 97077

Printed in U.S.A.

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.

Welcome

This programmer manual covers the TDS 420/460/520/540/620/640 version 2 and later. It also covers the TDS Option 2F Advanced DSP Math and the TDS Option 13 RS-232/Centronics Interface. This manual provides information on operating your oscilloscope using the General Purpose Interface Bus (GPIB) interface.

Related Manuals

Table i lists other documentation for the TDS 400/500/600 digitizing oscilloscopes.

Table i: Other TDS 400/500/600 Documentation

Manual	Tek Part Number
TDS User Manual	
TDS 420 & 460	070-8034-01
TDS 520 & 540	070-8317-01
TDS 620 & 640	070-8506-01
Option 13 – RS-232/Centronics I/F	070-8567-00
Option 2F – Adv. DSP Math	070-8582-00
TDS Reference	
TDS 420 & 460	070-8035-01
TDS 520 & 540	070-8316-01
TDS 620 & 640	070-8505-01
TDS Service Manual	
TDS 420 & 460	070-8036-01
TDS 520	070-8312-01
TDS 540	070-8314-01
TDS 620	070-8507-00
TDS 640	070-8508-00

Contents

Getting Started

Syntax and Commands

Status & Events

Programming Examples

Appendices

Glossary and Index

Change Information

List of Figures

List of Tables

Getting Started

You can write computer programs that remotely set the oscilloscope's front panel controls or take measurements and read those measurements for further analysis or storage.

This section covers the following:

- *This Manual* describes the major sections in this manual.
- *Setting Up Remote Communications* describes remote control. This includes connecting the oscilloscope and setting the appropriate front panel controls.

This Manual

This manual includes the following sections.

Syntax and Commands

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

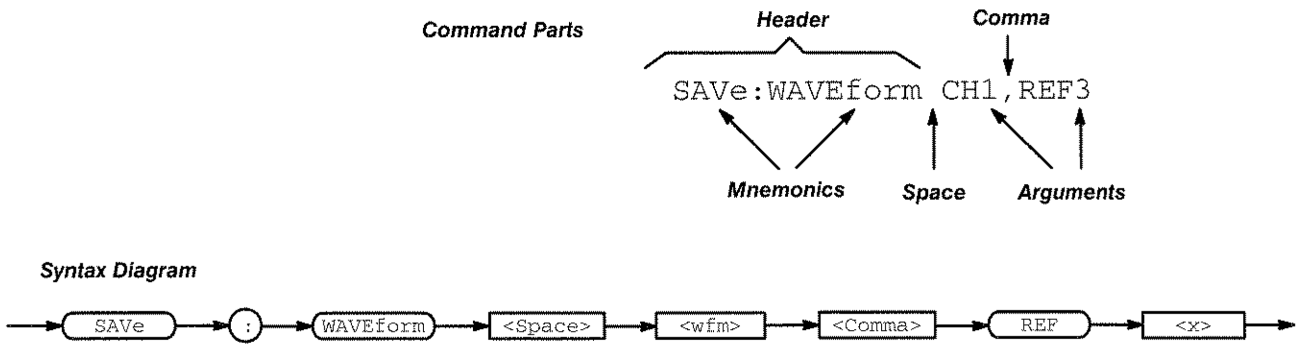
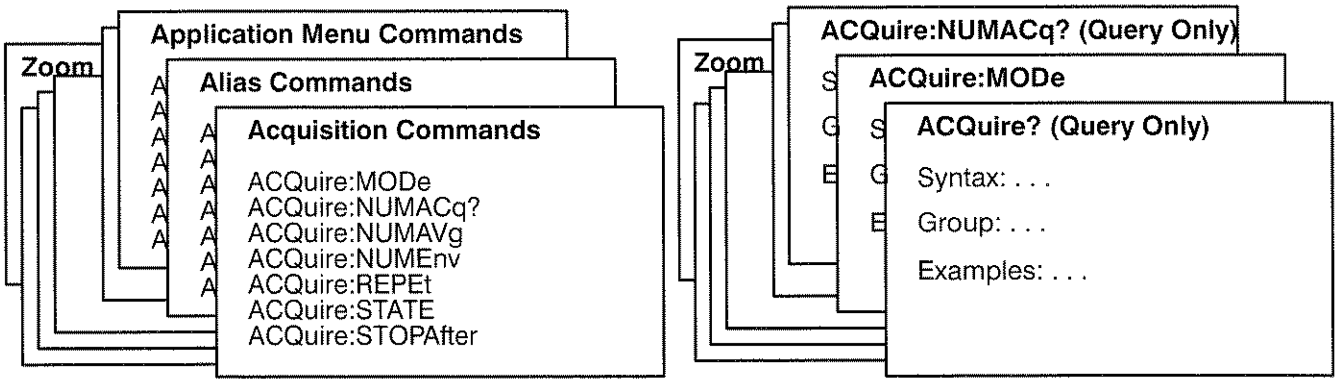


Figure 1-1: Common Message Elements

Section 2 also describes each command's effect and provides examples of how you might use it. The *Command Groups* subsection provides a list by functional area. The *Command Descriptions* subsection arranges commands alphabetically (Figure 1-2).



Commands Grouped in 17 Functional Area *and* *Commands Listed Alphabetically*

Figure 1-2: Functional Groupings and an Alphabetical List of Commands

Status and Events

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.

The *Status and Events* section (Section 3) starting on page NO TAG describes how to use service requests (SRQ's) and various event messages in your programs.

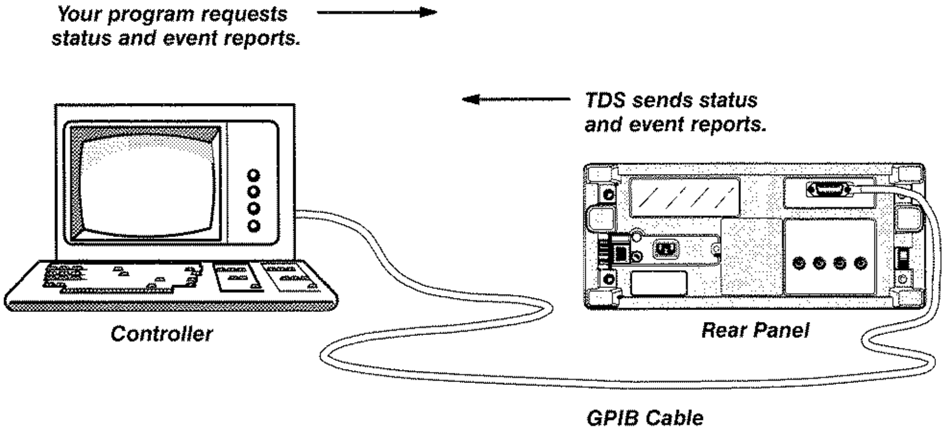

Your program requests status and event reports.
TDS sends status and event reports.
Controller
Rear Panel
GPIB Cable

Figure 1-3: Service Requests (SRQ's) Provide for Event (Interrupt) Driven Programs

Programming Examples

The *Programming Examples* section (Section 4) starting on page 4-1 describes some example digitizing oscilloscope programs and how to compile them. The disks that come with this manual (Figure 1-4) have an executable and a Microsoft QuickBASIC 4.5 and a Microsoft QuickC 2.5 source code version of each program.

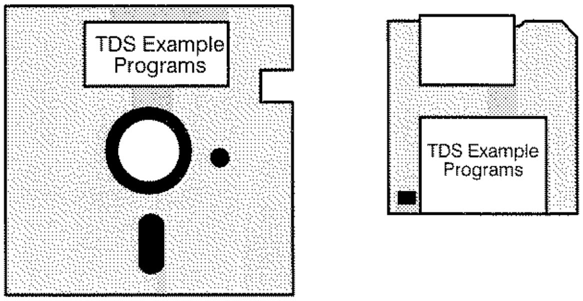


Figure 1-4: The Disks That Accompany This Manual

Setting Up Remote Communications

Even the best instrument control program will not do much if the instrument is not connected to the controller.

The digitizing oscilloscope has a 24-pin **GPIB** connector on its rear panel, as shown in Figure 1-5. This connector has a D-type shell and conforms to IEEE Std 488.1-1987.

Attach an IEEE Std 488.1-1987 GPIB cable (available from Tektronix as part number 012-0991-00) to this connector.

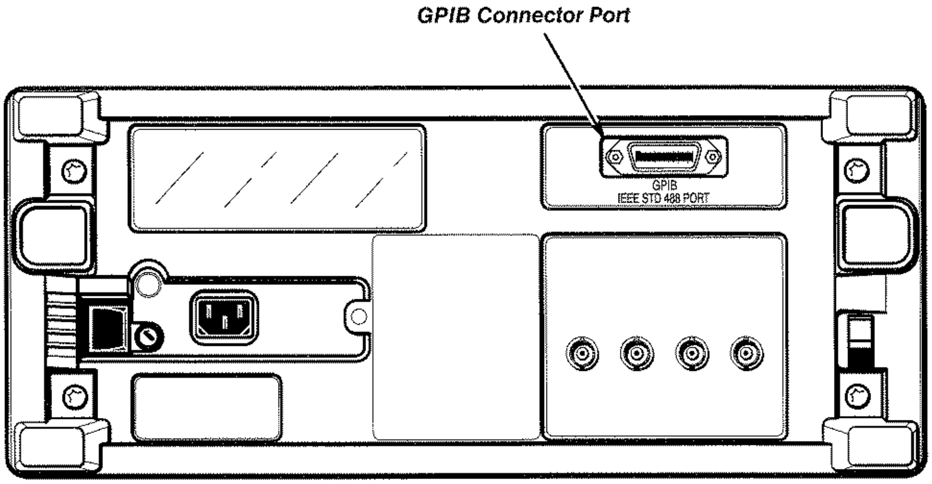


Figure 1-5: GPIB Connector Location

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

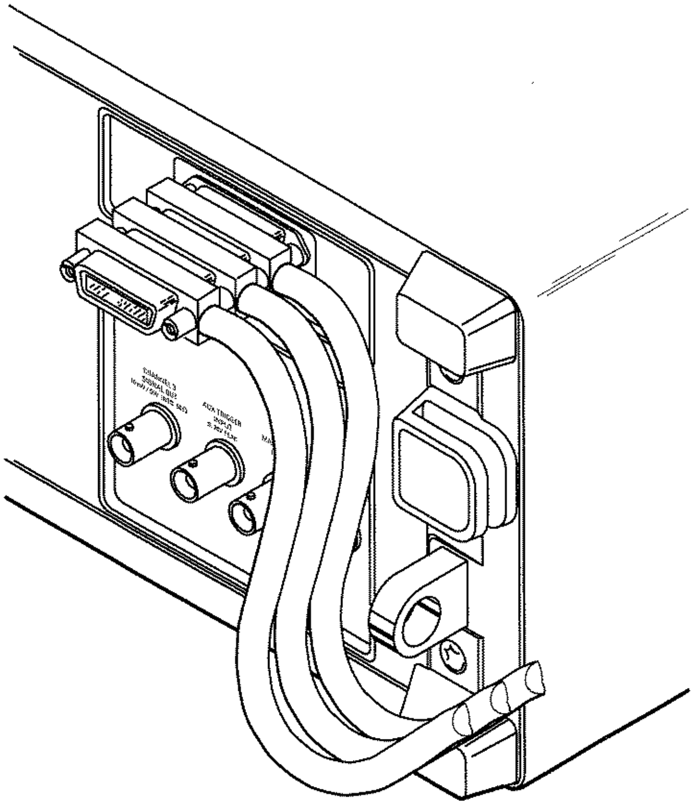


Figure 1-6: How to Stack GPIB Connectors

GPIB Requirements

Observe these rules when you use your digitizing oscilloscope with a GPIB network:

- Assign a unique device address to each device on the bus. No two devices can share the same device address.
- Do not connect more than 15 devices to any one bus.
- Connect one device for every 2 meters (6 feet) of cable used.
- Do not use more than 20 meters (65 feet) of cable to connect devices to a bus.
- Turn on at least two-thirds of the devices on the network while using the network.
- Connect the devices on the network in a star or linear configuration as shown in Figure 1-7. Do not use loop or parallel configurations.

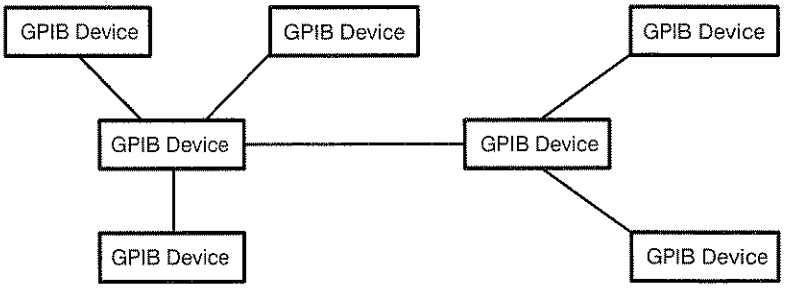


Figure 1-7: Typical GPIB Network Configurations

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

Setting the GPIB Parameters

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

□ **Step 1:** Press the **UTILITY** (**SHIFT DISPLAY**) button to display the Utility menu.

□ **Step 2:** Press the **System** button in the main menu until it highlights the **I/O** selection in the pop-up menu.

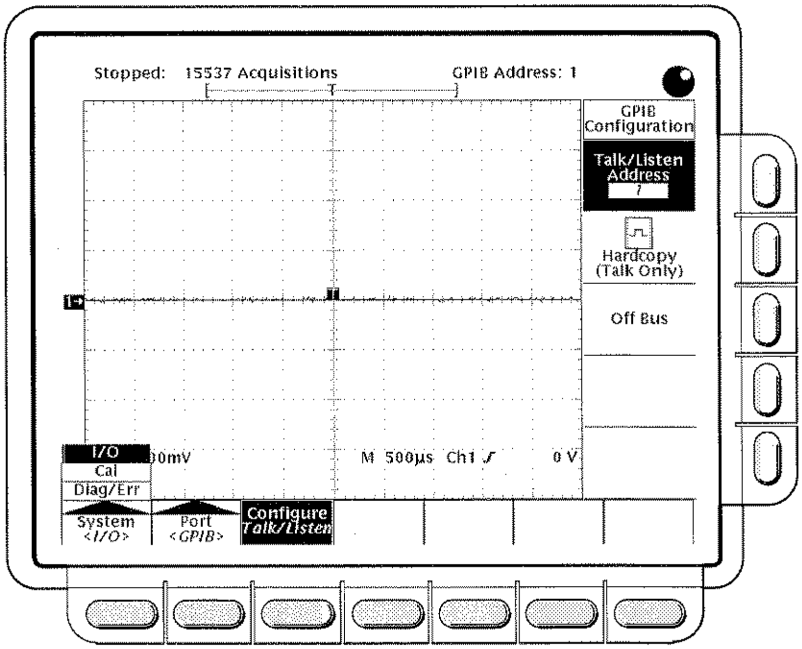


Figure 1-8: Selecting the I/O System in the Main Menu

□ **Step 3:** Press the **Port** button in the main menu until it highlights the **GPIB** selection in the pop-up menu.

□ **Step 4:** Press the **Configure** button in the main menu to display the GPIB Configuration side menu.

□ **Step 5:** Press the **Talk/Listen Address** side menu button, and set the GPIB address using either the general purpose knob or, if available, the keypad.

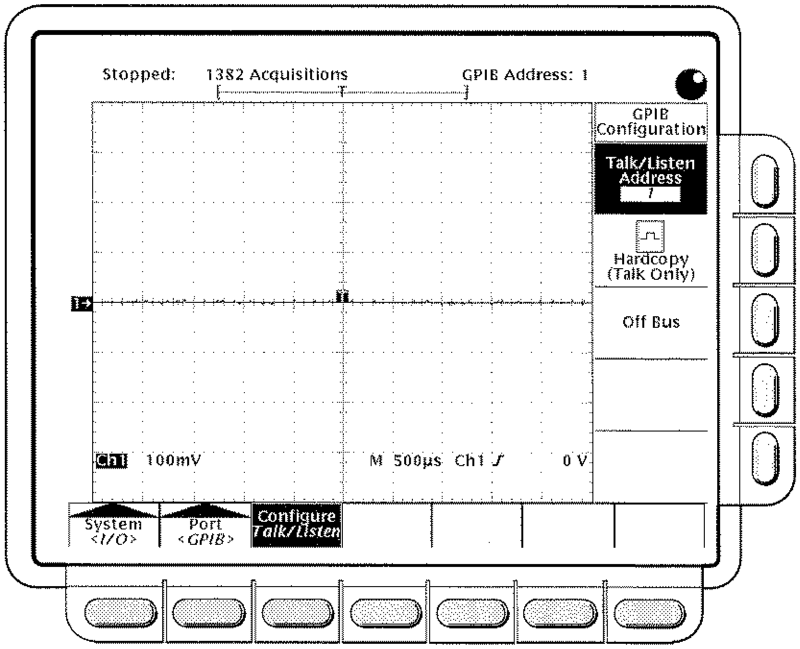


Figure 1-9: Selecting the GPIB Address in the GPIB Configuration Side Menu

The digitizing oscilloscope is set up for bidirectional communication with your controller. If you wish to isolate the digitizing oscilloscope from the bus:

- Press the **Off Bus** side menu button. This disables all communication with the controller.

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 to have the digitizing oscilloscope send hard copy information only when you press the **HARDCOPY** button (and accept a HARDCOPY ABORT command).

Command Syntax

You can control the digitizing oscilloscope through the GPIB and RS-232-C interfaces using commands and queries. This section describes the syntax these commands and queries use. It also describes the conventions the digitizing oscilloscope uses to process them. The next section, entitled *Commands,* lists the commands and queries themselves.

You transmit commands to the digitizing 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 describes commands and queries using Backus-Naur Form (BNF) notation and syntax diagrams.

This manual uses the following BNF symbols:

Table 2-1: BNF Symbols and Meanings

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

Command and Query Structure

Commands consist of set commands and query commands (usually simply called commands and queries). Commands modify instrument settings or tell the digitizing oscilloscope to perform a specific action. Queries cause the digitizing 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 differs from the set form by its 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 have set only and some have query only.

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

Table 2-2: Command Message Elements

Symbol	Meaning
`<Header>`	The basic command name. If the header ends with a question mark, the command is a query. The header may begin with a colon (:) character. If the command is concatenated with other commands, the beginning colon is required. Never use the beginning colon with command headers beginning with a star (*).
`<Mnemonic>`	A header sub-function. Some command headers have only one mnemonic. If a command header has multiple mnemonics, a colon (:) character always separates them from each other.
`<Argument>`	A quantity, quality, restriction, or limit associated with the header. Some commands have no argument while others have multiple arguments. A `<Space>` separates arguments from the header. A `<Comma>` separates arguments from each other.
`<Comma>`	A single comma between arguments of multiple-argument commands. It may optionally have white space characters before and after the comma.
`<Space>`	A white space character between command header and argument. It may optionally consist of multiple white space characters.

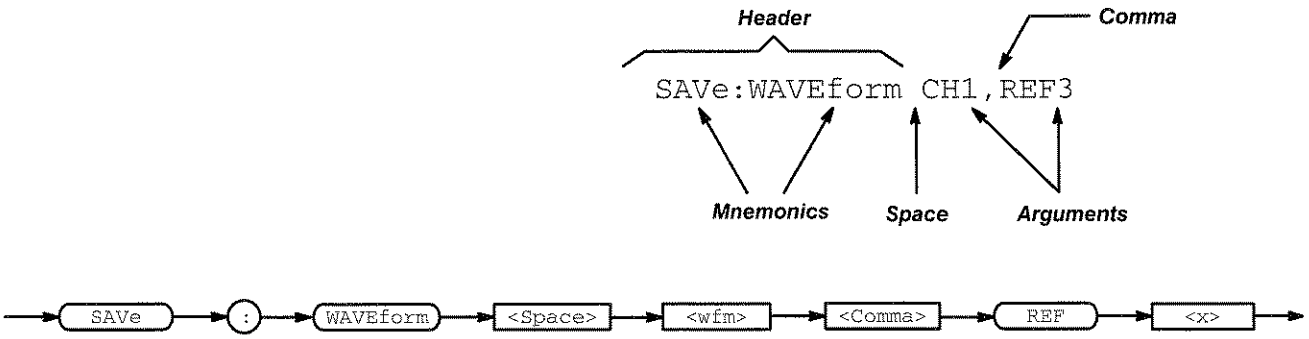


Figure 2-1: Command Message Elements

Commands

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

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

A command header consists 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 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 digitizing oscilloscope to return information about its status or settings. Queries have the structure:

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

You can specify a query command at any level within the command tree unless otherwise noted. These branch queries return information about all the mnemonics below the specified branch or level. For example, `MEASUrement:MEAS<x>:DELay:DIRection?` returns the starting point and direction of the edge of a delayed measurement, while `MEASUrement:MEAS<x>:DELay?` returns the current settings of all delayed measurement parameters, and `MEASUrement:MEAS<x>?` returns all the measurement parameters for the specified measurement.

Headers in Query Responses

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

Table 2-3: Comparison of Header Off and On Responses

Query	Header Off Response	Header On Response
APPMenu:TITLe?	"Test Setup"	:APPMENU:TITLE "Test Setup"
ACQuire:NUMAVg?	100	:ACQUIRE:NUMAVG 100

Clearing the Digitizing Oscilloscope

You can clear the Output Queue and reset the digitizing oscilloscope to accept a new command or query by using the Device Clear (DCL) GPIB command.

Command Entry

- 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 digitizing oscilloscope ignores commands consisting of any combination of white space characters and line feeds.

Abbreviating Commands

You can abbreviate many digitizing oscilloscope commands. Each command's listing in the *Commands* section shows the abbreviations in capitals. For example, you can enter the command `ACQuire:NUMAvg` 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 digitizing oscilloscope executes concatenated commands in the order received.

When concatenating commands and queries, you must follow these rules:

1. Separate completely different headers by a semicolon and by the beginning colon on all commands but the first. For example, the commands `TRIGger:MODe NORMal` and `ACQuire:NUMAVg 10` would be concatenated into a single command:

   ```
   TRIGger:MODe NORMal;:ACQuire:NUMAVg 10
   ```

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, you can concatenate the commands `ACQuire:MODe ENVelope` and `ACQuire:NUMAVg 10` into a single command:

   ```
   ACQuire:MODe ENVelope; NUMAVg 10
   ```

The longer version works equally well:

```
ACQuire:MODe ENVelope;:ACQuire:NUMAVg 10
```

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

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

   Any commands that follow will be processed as if the star command was not there so

   ```
   ACQuire:MODe ENVelope;*TRG;NUMAVg 10
   ```

   will set the acquisition mode to envelope and set the number of acquisitions for averaging to 10.

4. When you concatenate queries, the responses to all the queries are concatenated into a single response message. For example, if the display intensity for text is 80% and for the waveform it is 90%, the concatenated query

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

   will return either `:DISPLAY:INTENSITY:TEXT 80;:DISPLAY:INTENSITY:WAVEFORM 90` if header is on or `80;90` if header is off.

5. Set commands and queries may be concatenated 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. Concatenated commands and queries are executed in the order received.

Here are some invalid concatenations:

- `DISPlay:INTENsity:TEXt 80;ACQuire:NUMAVg 10`
  (no colon before `ACQuire`)
- `DISPlay:INTENsity:TEXt 80;:WAVEform 90`
  (extra colon before `WAVEform`—could use `DISPlay:INTENsity:WAVEform` instead)
- `DISPlay:INTENsity:TEXt 80;:*TRG`
  (colon before a star (*) command)
- `APPMenu:LABel:BOTTOM1 "foo";LABel:BOTTOM2 "fee"`
  (levels of the mnemonics are different—either remove the second use of `LABel:` or place `:APPMenu:` in front of `LABel:BOTTOM2`)

Message Terminators

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

Symbol	Meaning
`<EOM>`	Message terminator

The end-of-message terminator 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 digitizing oscilloscope always terminates messages with LF and EOI. It allows white space before the terminator. For example, it allows CR LF.

Constructed Mnemonics

Some header mnemonics specify one of a range of mnemonics. For example, a channel mnemonic can be either `CH1`, `CH2`, `CH3`, or `CH4`. You use these mnemonics in the command just as you do any other mnemonic. For example, there is a `CH1:VOLts` command, and there is also a `CH2:VOLts` command. In the command descriptions, this list of choices is abbreviated as `CH<x>`.

Application Menu Mnemonics

When the application menu is displayed, commands may specify which menu button to use.

Symbol	Meaning
`BOTTOM<x>`	A main menu button selector; `<x>` is `1`, `2`, `3`, `4`, `5`, `6`, or `7`. Main menu buttons are located along the bottom of the display and are numbered left to right, starting with 1.
`RIGHT<x>`	A side menu button selector; `<x>` is `1`, `2`, `3`, `4`, or `5`. Side menu buttons are located along the right side of the display and are numbered top to bottom, starting with 1.

Cursor Position Mnemonics

When cursors are displayed, commands may specify which cursor of the pair to use.

Symbol	Meaning
`POSITION<x>`	A cursor selector; `<x>` is either `1` or `2`.

Measurement Specifier Mnemonics

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

Symbol	Meaning
MEAS<x>	A measurement specifier; <x> is either 1 [top], 2, 3, or 4[bottom].

Channel Mnemonics

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

Symbol	Meaning

Math Waveform Mnemonics

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

Symbol	Meaning
MATH<x>	A math waveform specifier; <x> is 1, 2, or 3.

Reference Waveform Mnemonics

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

Symbol	Meaning
REF<x>	A reference waveform specifier; <x> is either 1, 2, 3, or 4.

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 such a waveform as follows:

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

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

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

Most numeric arguments will be automatically forced to a valid setting, either by rounding or truncating, when an invalid number is input unless otherwise noted in the command description.

Quoted String Arguments

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

```
"this is a quoted string"
```

Symbol	Meaning
`<QString>`	Quoted string of ASCII text

Follow these rules when you use quoted strings:

1. A quoted string can include any character defined in the 7-bit ASCII character set. (See Appendix A on page A-1).
2. Use the same type of quote character to open and close the string:

   ```
   "this is a valid string"
   ```

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

   ```
   "this is an 'acceptable' string"
   ```

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

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

5. Strings can have upper or lower case characters.
6. If you use a GPIB network, you cannot terminate a quoted string with the END message before the closing delimiter.
7. A carriage return or line feed imbedded in a quoted string does not terminate the string, but is treated as just another character in the string.
8. 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)

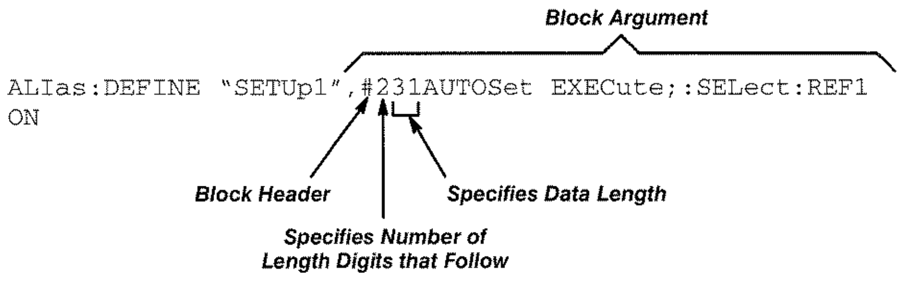
Block Arguments

Several digitizing oscilloscope commands use a block argument form:

Symbol	Meaning
`<NZDig>`	A non-zero digit character, in the range 1–9
`<Dig>`	A digit character, in the range 0–9
`<DChar>`	A character with the hex equivalent of 00 through FF hexadecimal (0 through 255 decimal)

`<Block>`	A block of data bytes, defined as:
	`<Block> ::=`
	`{ #<NZDig><Dig>[<Dig>...][<DChar>...]`
	`| #0[<DChar>...]<terminator> }`

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



```
ALIas:DEFINE "SETUp1",#231AUTOSet EXECute;:SELect:REF1
ON
```

Figure 2-2: Block Argument Example

Syntax Diagrams

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 command mnemonics are shown in both upper and lower case to distinguish between complete and abbreviated spellings. These elements are not case sensitive. 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.

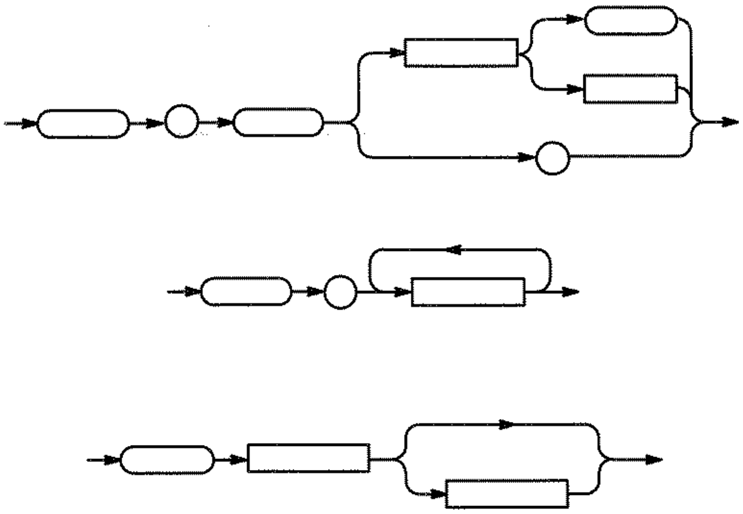


Figure 2-3: Typical Syntax Diagrams

Command Groups

This section lists Digitizing Oscilloscope commands in two ways. It first presents them by functional groups. It then lists them alphabetically. The functional group list starts below. The alphabetical list provides more detail on each command and starts on page 2-33.

The TDS Family's GPIB interface conforms to Tektronix standard codes and formats and IEEE Std 488.2-1987 except where noted.

Acquisition Commands

Acquisition commands affect waveform acquisition. These commands control mode, averaging, enveloping, and single-waveform acquisition. (Persistence controls are in the Display Commands section on page 2-14.)

Table 2-4: Acquisition Commands

Header	Description
`ACQuire?`	Return acquisition parameters
`ACQuire:MODe`	Acquisition mode
`ACQuire:NUMACq?`	Return # of acquisitions obtained
`ACQuire:NUMAVg`	Number of acquisitions for average
`ACQuire:NUMEnv`	Number of acquisitions for envelope
`ACQuire:REPEt` (TDS420/460/520/540)	Repetitive acquisition mode
`ACQuire:STATE`	Start or stop acquisition system
`ACQuire:STOPAfter`	Acquisition control

Alias Commands

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

Table 2-5: Alias Commands

Header	Description
`ALIas`	Turn the alias state on and off
`ALIas:CATalog?`	Return a list of aliases
`ALIas:DEFINE`	Create a new alias
`ALIas:DELEte`	Remove an alias
`ALIas:DELEte:ALL`	Remove all aliases
`ALIas:DELEte:NAMe`	Remove a named alias
`ALIas:STATE`	Turn the alias state on and off

Application Menu Commands

Application menu commands let you define special-purpose menus. You can define labels for the main and side menus as well as a side menu title. You can display an Application menu by either pressing the front-panel **APPLICATION** button or sending the APPMenu ACTivate command.

When the digitizing oscilloscope displays an Application menu and a user presses a front-panel button, the oscilloscope generates an event that tells the controller which button the user pressed. You can also set up the event reporting system so that it generates a Service Request when a user presses a menu button.

Table 2-6: Application Menu Commands

Header	Description
`APPMenu`	Display the application menu
`APPMenu:LABel`	Return or remove all application menu button labels
`APPMenu:LABel:BOTTOM<x>`	Label for a bottom menu button
`APPMenu:LABel:RIGHT<x>`	Label for a side menu button
`APPMenu:TITLe`	Create a title for the application menu

Calibration and Diagnostic Commands

Calibration and Diagnostic commands let you start the oscilloscope's built-in self-calibration and diagnostic routines. The diagnostic test operation includes selecting the test sequence, executing the sequence, and viewing the results.

Table 2-7: Calibration and Diagnostic Commands

Header	Description
`*CAL?`	Perform an internal self-calibration
`DIAg:RESULT:FLAG?`	Return diagnostic tests status
`DIAg:RESULT:LOG?`	Return diagnostic test sequence results
`DIAg:SELect:ACQUISition`	Acquisition system diagnostic test sequence
`DIAg:SELect:ALL`	Diagnostic test sequence for Acquisition, Processor, Display, and Front panel
`DIAg:SELect:CPU`	Processor diagnostic test sequence
`DIAg:SELect:DISplay`	Display system diagnostic test sequence
`DIAg:SELect:FPAnel`	Front panel diagnostic test sequence
`DIAg:STATE`	Control of diagnostic tests

Cursor Commands

Cursor commands provide control over cursor (caliper) display and readout.

Table 2-8: Cursor Commands

Header	Description
`CURSor?`	Returns cursor settings
`CURSor:FUNCtion`	Cursors on or off; select cursor type
`CURSor:HBArs?`	Return H bar settings
`CURSor:HBArs:DELTa?`	Return distance between H bars
`CURSor:HBArs:POSITION<x>`	Position a horizontal cursor
`CURSor:HBArs:SELect`	Set which cursor the knob controls
`CURSor:MODe`	Set cursor tracking mode

Table 2-8: Cursor Commands (Cont.)

Header	Description
`CURSor:PAIred:HDELTA?`	Query horizontal distance between 1st and 2nd paired cursors
`CURSor:PAIred:HPOS1?`	Query horizontal position of 1st paired cursor
`CURSor:PAIred:HPOS2?`	Query horizontal position of 2nd paired cursor
`CURSor:PAIred:POSITION1?`	Return vbar position of the1st paired cursor
`CURSor:PAIred:POSITION2?`	Return vbar position of the 2nd paired cursor
`CURSor:PAIred:SELect?`	Return active paired cursor
`CURSor:PAIred:VDELTA?`	Query vertical distance between 1st and 2nd paired cursors
`CURSor:VBArs`	Position vertical bar cursors
`CURSor:VBArs:DELTa?`	Horizontal distance between cursors
`CURSor:VBArs:POSITION<x>`	Position a vertical cursor
`CURSor:VBArs:SELect`	Set which cursor the knob controls
`CURSor:VBArs:UNIts`	Set vertical cursors to period or frequency

Display Commands

Display commands let you change the graticule style, change the displayed intensities, turn off waveform display, display messages, and clear the menu. When you turn off waveform display, waveforms are acquired and transmitted but not displayed. The update rate is much faster when waveform display is off.

Table 2-9: Display Commands

Header	Description
`CLEARMenu`	Clear menus from display
`DISplay?`	Returns display settings
`DISplay:CLOCk`	Controls the display of the date/time stamp
`DISplay:FILTer`	Displayed data interpolation
`DISplay:FORMat`	YT or XY display

Table 2-9: Display Commands (Cont.)

Header	Description
`DISplay:GRAticule`	Graticule style
`DISplay:INTENSITy?`	Returns intensity settings
`DISplay:INTENSITy:CONTRast`	Waveform intensified zone brightness
`DISplay:INTENSITy:OVERALL`	Main brightness
`DISplay:INTENSITy:TEXt`	Text brightness
`DISplay:INTENSITy:WAVEform`	Waveform brightness
`DISplay:PERSistence`	Variable persistence decay time
`DISplay:STYle`	Waveform dots, vector, infinite or variable persistence
`DISplay:TRIGT`	Controls the display of the trigger indicator on screen
`DISplay:TRIGBar`	Controls the display of the trigger bar/s on screen
`MESSage`	Remove text from the message window
`MESSage:BOX`	Set size and location of message window
`MESSage:SHOw`	Remove and display text in the message window
`MESSage:STATE`	Control display of message window

Hardcopy Commands

Hardcopy commands let you control the format of hardcopy output and the initiation and termination of hardcopies.

Table 2-10: Hardcopy Commands

Header	Description
`HARDCopy`	Start or terminate hardcopy
`HARDCopy:FORMat`	Hardcopy output format
`HARDCopy:LAYout`	Hardcopy orientation
`HARDCopy:PORT`	Hardcopy port for output

Horizontal Commands

Horizontal commands control the time bases of the digitizing 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.

You may substitute SECdiv for SCAle in the horizontal commands. This provides program compatibility with earlier models of Tektronix digitizing oscilloscopes.

Table 2-11: Horizontal Commands

Header	Description
`HORizontal?`	Return horizontal settings
`HORizontal:DELay?`	Return delay time base settings
`HORizontal:DELay:MODe`	Delay time base mode
`HORizontal:DELay:SCAle`	Delay time base time/division
`HORizontal:DELay:SECdiv`	Same as HORizontal:DELay:SCAle
`HORizontal:DELay:TIMe`	Delay time
`HORizontal:DELay:TIMe?`	Return delay time parameters
`HORizontal:DELay:TIMe:RUNSAfter`	Time to wait in delay-runs-after-main mode
`HORizontal:DELay:TIMe:TRIGAfter`	Time to wait in delay-runs-after-trigger mode
`HORizontal:MAIn?`	Returns main time/division
`HORizontal:MAIn:SCAle`	Main time base time/division
`HORizontal:MAIn:SECdiv`	Same as HORizontal:MAIn:SCAle
`HORizontal:MODe`	Turn delay time base on or off
`HORizontal:POSition`	Portion of waveform to display
`HORizontal:RECOrdlength`	Number of points in waveform record
`HORizontal:SCAle`	Same as HORizontal:MAIn:SCAle
`HORizontal:SECdiv`	Same as HORizontal:MAIn:SCAle
`HORizontal:TRIGger?`	Return trigger position
`HORizontal:TRIGger:POSition`	Main time base trigger position

Limit Test Commands

The Limit Test commands let you automatically compare each incoming waveform against a template waveform. You set an envelope of limits around a waveform and let the digitizing oscilloscope find the waveforms that fall outside those limits. When it finds such a waveform, it can generate a hardcopy, ring a bell, stop and wait for your input, or any combination of these actions.

Table 2-12: Limit Test Commands

Header	Description
LIMit:BELl	Ring bell when limit exceeded
LIMit:COMpare:CH<x>	Template to compare waveform to
LIMit:HARDCopy	Make hardcopy when limit exceeded
LIMit:STATE	Limit testing on or off
LIMit:TEMPLate	Template to compare waveform to
LIMit:TEMPLate:DESTination	Ref. storage for template waveform
LIMit:TEMPLate:SOUrce	Template waveform source
LIMit:TEMPLate:TOLerance:HORizontal	Tested waveform horizontal tolerance
LIMit:TEMPLate:TOLerance:VERTical	Tested waveform vertical tolerance

Measurement Commands

Measurement commands control the automated measurement system. Up to four automated measurements can be displayed on the screen. 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 measurements, the measurement commands let you specify a fifth measurement, IMMed. The immediate measurement has no front-panel equivalent. Immediate measurements are never displayed. Because they are computed only when needed, immediate measurements slow the waveform update rate less than displayed measurements.

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

Measurement commands can 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 2-13: Measurement Commands

Header	Description
MEASUrement?	Returns all measurement parameters
MEASUrement:CLEARSNapshot	Take down measurement snapshot
MEASUrement:GATING	Set or query measurement gating
MEASUrement:IMMed?	Return immediate measurement parameters.
MEASUrement:IMMed:DELay?	Return info on immediate delay measurement
MEASUrement:IMMed:DELay:DIRection	Search direction to use for delay measurements
MEASUrement:IMMed:DELay:EDGE1	Which waveform edge to use for delay measurements
MEASUrement:IMMed:DELay:EDGE2	Which waveform edge to use for delay measurements
MEASUrement:IMMed:SOURCE[1]	Channel to take measurement from
MEASUrement:IMMed:SOURCE2	Second channel to take measurement from (delay or "to" channel)
MEASUrement:IMMed:TYPe	The measurement to be taken
MEASUrement:IMMed:UNIts?	Return measurement units
MEASUrement:IMMed:VALue?	Return measurement result
MEASUrement:MEAS<x>?	Return parameters on measurement
MEASUrement:MEAS<x>:DELay?	Return delay measurement parameters.
MEASUrement:MEAS<x>:DELay:DIRection	Search direction to use for delay measurements
MEASUrement:MEAS<x>:DELay:EDGE1	Which waveform edge to use for delay measurements
MEASUrement:MEAS<x>:DELay:EDGE2	Which waveform edge to use for delay measurements
MEASUrement:MEAS<x>:SOURCE[1]	Channel to take measurement from
MEASUrement:MEAS<x>:SOURCE2	Second channel to take measurement from (delay or "to" channel)
MEASUrement:MEAS<x>:STATE	Turn measurement display on or off
MEASUrement:MEAS<x>:TYPe	The measurement to be taken

Table 2-13: Measurement Commands (Cont.)

Header	Description
MEASUrement:MEAS<x>:UNIts?	Units to use for measurement
MEASUrement:MEAS<x>:VALue?	Measurement result query
MEASUrement:METHod	Method for calculating reference levels
MEASUrement:REFLevel?	Returns reference levels
MEASUrement:REFLevel:ABSolute:HIGH	The top level for risetime (90% level)
MEASUrement:REFLevel:ABSolute:LOW	The low level for risetime (10% level)
MEASUrement:REFLevel:ABSolute:MID	Mid level for measurements
MEASUrement:REFLevel:ABSolute:MID2	Mid level for delay measurements
MEASUrement:REFLevel:METHod	Method to assign HIGH and LOW levels: either % or absolute volts
MEASUrement:REFLevel:PERCent:HIGH	The top level for risetime (90% level)
MEASUrement:REFLevel:PERCent:LOW	The low level for risetime (10% level)
MEASUrement:REFLevel:PERCent:MID	Mid level for measurements
MEASUrement:REFLevel:PERCent:MID2	Mid level for delay measurements
MEASUrement:SNAPShot	Displays measurement snapshot

Miscellaneous Commands

Miscellaneous commands do not fit into other categories.

Several commands and queries are common to all 488.2-1987 devices on the GPIB bus. The 488.2-1987 standard defines them. They begin with a star (*) character.

Table 2-14: Miscellaneous Commands

Header	Description
`AUTOSet`	Automatic instrument setup
`BELl`	Audio alert
`*DATE`	Set date
`*DDT`	Define group execute trigger (GET)
`FACtory`	Reset to factory default
`HDR`	Same as HEADer
`HEADer`	Return command header with query
`*IDN?`	Identification
`*LRN?`	Learn device setting
`LOCk`	Lock front panel (local lockout)
`NEWpass`	Change password for User Protected Data
`PASSWord`	Access to change User Protected Data
`REM`	No action; remark only
`RS232:BAUd` (Option 13 only)	Set RS232 baud rate
`RS232:HARDFlagging` (Option 13 only)	Set RS232 hard flagging
`RS232:PARity` (Option 13 only)	Set RS232 parity
`RS232:SOFTFlagging` (Option 13 only)	Set RS232 soft flagging
`RS232:STOPBits` (Option 13 only)	Set # of stop bits for RS232
`RS232?` (Option 13 only)	Query RS232 parameters
`SET?`	Same as *LRN?
`TEKSecure`	Initialize waveforms and setups
`*TIMe`	Set time
`*TRG`	Perform Group Execute Trigger (GET)

Table 2-14: Miscellaneous Commands (Cont.)

Header	Description
*TST?	Self-test
UNLock	Unlock front panel (local lockout)
VERBose	Return full command name or minimum spellings with query

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 digitizing oscilloscope. When you then "recall a setting," the digitizing oscilloscope restores itself to the state it was in when you originally saved that setting.

Table 2-15: Save and Recall Commands

Header	Description
ALLOcate?	Return number of allocated and unallocated data points
ALLOcate:WAVEFORM?	Return number of allocated data points
ALLOcate:WAVEFORM:FREE?	Return number of unallocated data points
ALLOcate:WAVEFORM:REF<x>?	Specify the number of allocated data points
DELEte:SETUp	Delete stored setup
DELEte:WAVEFORM	Delete stored waveform
*RCL	Recall setting
RECAll:SETUp	Recall saved instrument setting
*SAV	Save setting
SAVe:SETUp	Save instrument setting
SAVe:WAVEFORM	Save waveform

Status and Error Commands

Table 2-16 lists the status and error commands the digitizing oscilloscope supports. These commands let you determine the status of the digitizing oscilloscope, and control events.

Several commands and queries used with the digitizing oscilloscope are common to all devices on the GPIB bus. IEEE Std 488.2-1987 defines these commands and queries. They begin with an asterisk (*).

Table 2-16: Status and Error Commands

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

Trigger Commands

Trigger commands control all aspects of digitizing oscilloscope triggering. There are two triggers, main and delayed. Where appropriate, the command set has parallel constructions for each trigger.

You can set the main or delayed triggers to edge mode. Edge triggering lets you display a waveform at or near the point where the signal passes through a voltage level of your choosing.

You can also set TDS 500 and 600 main triggers to pulse and logic modes. Pulse triggering lets the oscilloscope trigger whenever it detects a pulse of a certain width or height. Logic triggering lets you logically combine the signals on one or more channels. The digitizing oscilloscope then triggers when it detects a certain combination of signal levels.

Table 2-17: Trigger Commands

Header	Description
`TRIGger`	Force trigger event; Return parameters.
`TRIGger:DELay`	Delay trigger level to 50%
`TRIGger:DELay:BY`	Delay by time or events
`TRIGger:DELay:EDGE?`	Return delay trigger parameters
`TRIGger:DELay:EDGE:COUPling`	Delay trigger coupling
`TRIGger:DELay:EDGE:SLOpe`	Delay trigger slope
`TRIGger:DELay:EDGE:SOUrce`	Delay trigger source
`TRIGger:DELay:EVENTS?`	Return delay trigger event parameters
`TRIGger:DELay:EVENTS:COUNt`	Delay by events count
`TRIGger:DELay:LEVel`	Delay trigger level
`TRIGger:DELay:TIMe`	Time for delay by time
`TRIGger:DELay:TYPe`	Delay trigger, edge or video
`TRIGger:MAIn`	Main trigger level to 50%
`TRIGger:MAIn:EDGE?`	Return main edge trigger parameters
`TRIGger:MAIn:EDGE:COUPling`	Main trigger coupling
`TRIGger:MAIn:EDGE:SLOpe`	Main trigger slope
`TRIGger:MAIn:EDGE:SOUrce`	Main trigger source
`TRIGger:MAIn:HOLDoff?`	Return main trigger holdoff value
`TRIGger:MAIn:HOLdoff:VALue`	Main trigger holdoff value

Table 2-17: Trigger Commands (Cont.)

Header	Description
TRIGger:MAIn:LEVel	Main trigger level
TRIGger:MAIn:LOGIc? (TDS 520/540/620/640)	Returns main logic trigger parameters
TRIGger:MAIn:LOGIc:CLAss (TDS 520/540/620/640)	Logic trigger input usage
TRIGger:MAIn:LOGIc:FUNCtion (TDS520/540/620/640)	Logic trigger input combining
TRIGger:MAIn:LOGIc:INPut? (TDS520/540/620/640)	Return main logic trigger input settings
TRIGger:MAIn:LOGIc:INPut :CH<x> (TDS520/540/620/640)	Logic trigger expected channel state
TRIGger:MAIn:LOGIc:PATtern :INPut:CH4 (TDS 520/540/620/640)	Logic trigger expected for channel 4 pattern class
TRIGger:MAIn:LOGIc:PATtern :WHEn (TDS 520/540/620/640)	Main logic pattern trigger condition
TRIGger:MAIn:LOGIc:PATtern :WHEn:LESSLimit (TDS520/540/620/640)	Maximum time the selected pattern may be true and still generate main logic pattern trigger
TRIGger:MAIn:LOGIc:PATtern :WHEn:MORELimit (TDS520/540/620/640)	Minimum time the selected pattern may be true and still generate main logic pattern trigger
TRIGger:MAIn:LOGIc:STATE: INPut:CH4 (TDS 520/540/620/640)	Logic trigger expected for channel 4 state class
TRIGger:MAIn:LOGIc:STATE:WHEn (TDS 520/540/620/640)	When the logic trigger occurs (on true or false)
TRIGger:MAIn:LOGIc: THReshold? (TDS 520/540/620/640)	Return main logic thresholds
TRIGger:MAIn:LOGIc: THReshold:CH<x> (TDS 520/540/620/640)	Logic trigger thresholds
TRIGger:MAIn:LOGIc:WHEn (TDS 520/540/620/640)	Logic trigger on combination true or false
TRIGger:MAIn:MODe	Main trigger mode
TRIGger:MAIn:PULse? (TDS 520/540/620/640)	Returns pulse trigger parameters

Table 2-17: Trigger Commands (Cont.)

Header	Description
TRIGger:MAIn:PULse:CLAss (TDS 520/540/620/640)	Pulse trigger class
TRIGger:MAIn:PULse:GLItch? (TDS 520/540/620/640)	Returns glitch trigger parameters
TRIGger:MAIn:PULse:GLItch:FILTer (TDS 520/540/620/640)	Glitch filter on and off
TRIGger:MAIn:PULse:GLItch:POLarity (TDS 520/540/620/640)	Glitch filter positive, negative, or both
TRIGger:MAIn:PULse:GLItch:WIDth (TDS 520/540/620/640)	Glitch trigger with differentiation between glitch and valid pulse
TRIGger:MAIn:PULse:RUNT? (TDS 520/540/620/640)	Return runt trigger parameters
TRIGger:MAIn:PULse:RUNT:POLarity (TDS 520/540/620/640)	Runt trigger positive, negative, or both
TRIGger:MAIn:PULse:RUNT:THReshold? (TDS 520/540/620/640)	Return runt trigger thresholds
TRIGger:MAIn:PULse:RUNT:THReshold:HIGH (TDS 520/540/620/640)	Upper limit for runt pulse
TRIGger:MAIn:PULse:RUNT:THReshold:LOW (TDS 520/540/620/640)	Lower limit for runt pulse
TRIGger:MAIn:PULse:SOUrce (TDS 520/540/620/640)	Pulse trigger channel
TRIGger:MAIn:PULse:WIDth:HIGHLimit (TDS 520/540/620/640)	Pulse trigger maximum pulse width
TRIGger:MAIn:PULse:WIDth:LOWLimit (TDS 520/540/620/640)	Pulse trigger minimum pulse width
TRIGger:MAIn:PULse:WIDth:POLarity (TDS 520/540/620/640)	Pulse trigger positive, negative, or both
TRIGger:MAIn:PULse:WIDth:WHEn (TDS 520/540/620/640)	Pulse trigger when pulse detected or when not detected
TRIGger:MAIn:TYPe	Main trigger edge, logic, pulse, video
TRIGger:MAIn:VIDeo:BY (TDS 420/460 Option 5)	Video trigger delay mode
TRIGger:MAIn:VIDeo:FIELD (TDS 420/460 Option 5)	Video trigger field

Table 2-17: Trigger Commands (Cont.)

Header	Description
TRIGger:MAIn:VIDeo:HOLdoff? (TDS 420/460 Option 5)	Return video trigger holdoff
TRIGger:MAIn:VIDeo:HOLdoff:VALue (TDS 420/460 Option 5)	Video trigger holdoff value
TRIGger:MAIn:VIDeo:INTERLAce (TDS 420/460 Option 5)	Video trigger interlace format
TRIGger:MAIn:VIDeo:LINES (TDS 420/460 Option 5)	Video trigger delay in terms of a number of lines
TRIGger:MAIn:VIDeo:SCAN (TDS 420/460 Option 5)	Video trigger scan rate
TRIGger:MAIn:VIDeo:SOUrce (TDS 420/460 Option 5)	Video trigger source
TRIGger:MAIn:VIDeo:SYNc (TDS 420/460 Option 5)	Video trigger sync polarity
TRIGger:MAIn:VIDeo:SYStem (TDS 420/460 Option 5)	Video trigger class
TRIGger:MAIn:VIDeo:TIMe (TDS 420/460 Option 5)	Video trigger delay time
TRIGger:STATE?	Trigger system status

Vertical Commands

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

You may replace VOLts for SCAle in the vertical commands. This provides program compatibility with earlier models of Tektronix digitizing oscilloscopes.

Table 2-18: Vertical Commands

Header	Description
CH<x>?	Return vertical parameters
CH<x>:BANdwidth	Channel bandwidth
CH<x>:COUPling	Channel coupling
CH<x>:IMPedance	Channel impedance
CH<x>:OFFSet	Channel offset

Table 2-18: Vertical Commands (Cont.)

Header	Description
`CH<x>:POSition`	Channel position
`CH<x>:PRObe?`	Return channel probe attenuation
`CH<x>:SCAle`	Channel volts/div
`CH<x>:VOLts`	Same as CH<x>:SCAle
`MATH<x>?`	Return math waveform definition
`MATH<x>:DEFINE`	Math waveform
`SELect?`	Return selected waveform
`SELect:<wfm>`	Set selected waveform
`SELect:CONTROl`	Front-panel channel selector

Waveform Commands

Waveform commands let you transfer waveform data points to and from the digitizing 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 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 mode uses one 8-bit byte per waveform data point. Data acquired in HIRes or AVErage mode uses two 8-bit bytes per point. For more information on the acquisition modes see the ACQuire:MODe command on page 2-33.

The DATa:WIDth command lets you specify the number of bytes per data point when transferring data to and from the digitizing 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 digitizing 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 signed integer values. The range of the values depends on the byte width specified. One byte wide data ranges from −128 to 127. Two byte wide data ranges from −32768 to 32767.

Each data value requires two to seven characters. This includes one to five characters to represent the value, another character, if the value is negative, to represent a minus sign, and a comma to separate the data points.

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
```

Use ASCII to obtain more human readable and easier to format output than binary. On the other side, it may require more bytes to send the same values with ASCII as binary. This may reduce transmission speeds.

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. The four binary formats are RIBinary, RPBinary, SRIbinary, and SRPbinary.

RIBinary 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 RIBinary 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 digitizing oscilloscope, you must specify the location of the first data point within the waveform record. For example, when you set DATa:STARt to 1, data points will be stored starting with the first point in the record, and when you set DATa:STARt to 500, data will be stored starting at the 500th point in the record. The digitizing oscilloscope will ignore DATa:STOP when reading in data as it will stop reading data when it has no more data to read or when it has reached the specified record length.

When transferring data from the digitizing 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 will always return the entire waveform. You can also use the vertical bar cursors to delimit the portion of the waveform that you want to transfer. DATa:STARt and DATa:STOP can then be set to the current cursor positions by sending the command DATa SNAp.

Waveform Data Locations and Memory Allocation

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

You can transfer in to the digitizing oscilloscope only one waveform at a time. Waveforms sent to the oscilloscope are always stored in one of the four reference memory locations. You can specify the reference memory location with the DATa:DESTination command. You must define the memory size for the specified location before you store the data. The ALLOcate:WAVEFORM:REF<x> command lets you specify the memory size for each reference location.

Waveform Preamble

Each waveform that you transfer has an associated waveform preamble that contains information such as the horizontal scale, the vertical scale, and other settings in place when the waveform was created. Refer to the WFMPre command starting on page 2-207 for more information about the waveform preamble.

Scaling Waveform Data

Once you transfer the waveform data to the controller, you can convert the data points into voltage values for analysis using information from the waveform preamble. The `GETWFM` program on the diskettes that come with this manual shows how you can scale data.

Transferring Waveform Data from the Digitizing Oscilloscope

You can transfer waveforms from the digitizing oscilloscope to an external controller using the following sequence:

□ **Step 1:** Select the waveform source(s) using the DATa:SOUrce command. If you want to transfer multiple waveforms, select more than one source.

□ **Step 2:** Specify the waveform data format using DATa:ENCdg.

□ **Step 3:** Specify the number of bytes per data point using DATa:WIDth.

□ **Step 4:** Specify the portion of the waveform that you want to transfer using DATa:STARt and DATa:STOP.

□ **Step 5:** Transfer waveform preamble information using WFMPRe? query.

□ **Step 6:** Transfer waveform data from the digitizing oscilloscope using the CURVe? query.

Transferring Waveform Data to the Digitizing Oscilloscope

You can transfer waveform data to one of the four reference memory locations in the digitizing oscilloscope using the following sequence:

□ **Step 1:** Specify waveform reference memory using DATa:DESTination.

□ **Step 2:** Specify the memory size for the reference location specified in Step 1 using the ALLOcate:WAVEFORM:REF<x> command.

□ **Step 3:** Specify the waveform data format using DATa:ENCdg.

□ **Step 4:** Specify the number of bytes per data point using DATa:WIDth.

□ **Step 5:** Specify first data point in the waveform record using DATa:STARt.

□ **Step 6:** Transfer waveform preamble information using WFMPRe:<wfm>.

□ **Step 7:** Transfer waveform data to the digitizing oscilloscope using CURVe.

Table 2-19: Waveform Commands

Header	Description
CURVe	Transfer waveform data
DATa	Waveform data format and location
DATa:DESTination	Destination for waveforms sent to digitizing oscilloscope
DATa:ENCdg	Waveform data encoding method
DATa:SOUrce	Source of CURVe? data
DATa:STARt	Starting point in waveform transfer
DATa:STOP	Ending point in waveform transfer
DATa:TARget	Same as DATa:DESTination
DATa:WIDth	Byte width of waveform points
WAVFrm?	Returns waveform preamble and data
WAVPre?	Returns waveform format data
WFMPre:BIT_Nr	Preamble bit width of waveform points.
WFMPre:BN_Fmt	Preamble binary encoding type
WFMPre:BYT_Nr	Preamble byte width of waveform points

Table 2-19: Waveform Commands (Cont.)

Header	Description
WFMPre:BYT_Or	Preamble byte order of waveform points
WFMPre:CRVchk	Preamble checksum of waveform points
WFMPre:ENCdg	Preamble encoding method
WFMPre:NR_Pt	Number of points in the curve
WFMPre:PT_Fmt	Format of curve points
WFMPre:PT_Off	Trigger position
WFMPre:WFId	Curve identifier
WFMPre:XINcr	Horizontal sampling interval
WFMPre:XMUlt	Horizontal scale factor
WFMPre:XOFf	Horizontal offset
WFMPre:XUNit	Horizontal units
WFMPre:XZEro	Horizontal origin offset
WFMPre:YMUlt	Vertical scale factor
WFMPre:YOFf	Vertical offset
WFMPre:YUNit	Vertical units
WFMPre:YZEro	Offset voltage
WFMPre:ZMUlt	Z-axis scale factor
WFMPre:ZOFf	Z-axis offset
WFMPre:ZUNit	Z-axis units
WFMPre:ZZEro	Z-axis origin offset
WFMPre:<wfm>:NR_Pt	Number of points in the curve
WFMPre:<wfm>:PT_Fmt	Format of curve points
WFMPre:<wfm>:PT_Off	Trigger position
WFMPre:<wfm>:WFId	Curve identifier
WFMPre:<wfm>:XINcr	Horizontal sampling interval
WFMPre:<wfm>:XUNit	Horizontal units
WFMPre:<wfm>:YMUlt	Vertical scale factor
WFMPre:<wfm>:YOFf	Vertical offset

Table 2-19: Waveform Commands (Cont.)

Header	Description
`WFMPre:<wfm>:YUNit`	Vertical units
`WFMPre:<wfm>:YZEro`	Offset voltage

Zoom Commands

Zoom commands let you expand and position the waveform display horizontally and vertically without changing the time base or vertical settings.

Table 2-20: Zoom Commands

Header	Description
`ZOOm`	Resets zoom parameters to defaults
`ZOOm:HORizontal:LOCk`	Horizontal zoom lock
`ZOOm:HORizontal:POSition`	Horizontal zoom position
`ZOOm:HORizontal:SCAle`	Horizontal zoom scale
`ZOOm:STATE`	Turn zoom mode on or off
`ZOOm:VERTical:POSition`	Vertical zoom position
`ZOOm:VERTical:SCAle`	Vertical zoom scale

Command Descriptions

You can use commands to either set instrument features or query instrument 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 spells out fully headers, mnemonics, and arguments with the minimal spelling shown in upper case. For example, to use the abbreviated form of the ACQuire:MODe command just type ACQ:MOD.

ACQuire? (Query Only)

Returns all the current acquisition parameters.

Group: Acquisition

Syntax: `ACQuire?`

Examples: `ACQUIRE?`

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

ACQuire:MODe

Sets or queries the acquisition mode of the digitizing 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 set by the horizontal scale (time per division). The digitizing 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: For the TDS 420/460/520/540:

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

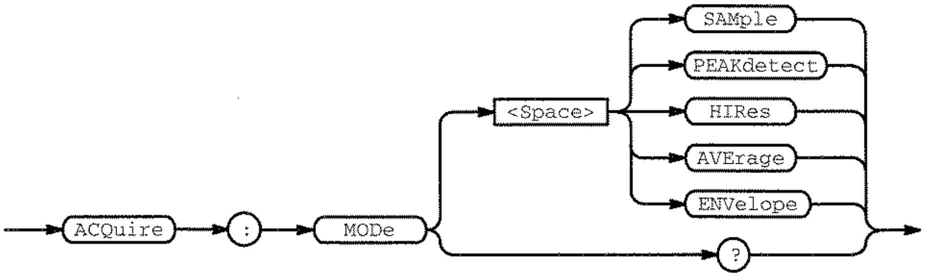
For the TDS 620/640:

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

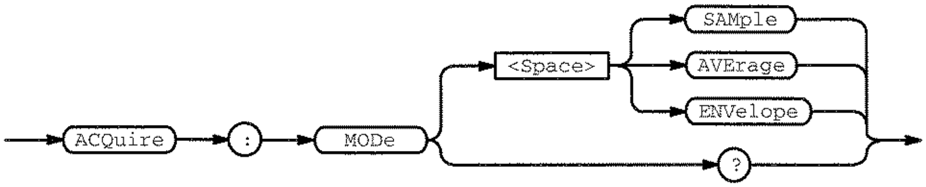
For all TDS:

```
ACQuire:MODe?
```

For the TDS 420/460/520/540:



For the TDS 620/640:



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` (for the TDS 420/460/520/540) specifies the display of the high-low range of the samples taken from a single waveform acquisition. The high-low range is displayed 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.

`HIRes` (for the TDS 420/460/520/540) specifies Hi Res mode, where the displayed data point value is the average of all the samples taken during the acquisition interval. This is a form of averaging, where the average comes from a single waveform acquisition. The number of samples taken during the acquisition interval determines the number of data values that compose the average.

`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`.

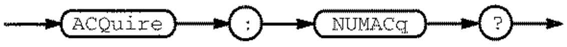
ACQuire:NUMACq? (Query Only)

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 $2^{30}-1$. Counting stops when this number is reached. This is the same value that is displayed 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.

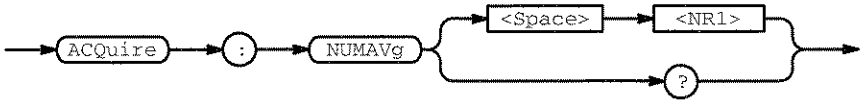
Group: Acquisition

Related Commands: ACQuire:MODe

Syntax:

```
ACQuire:NUMAVg <NR1>
```

```
ACQuire:NUMAVg?
```



Arguments: `<NR1>` is the number of waveform acquisitions, from 2 to 10,000.

Examples:

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

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

ACQuire:NUMENv

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

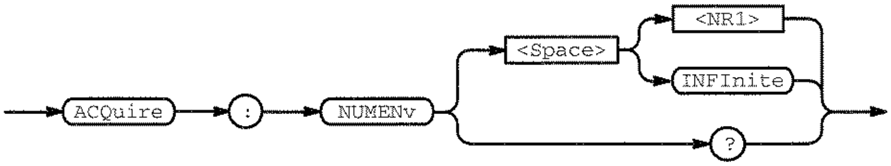
Group: Acquisition

Related Commands: ACQuire:MODe

Syntax:

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

```
ACQuire:NUMENv?
```



Arguments: `<NR1>` ≠ 0 is the number of waveform acquisitions, from 1 to 2000. 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 digitizing oscilloscope will envelope a maximum of 2001 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:REPEt

TDS 420/460/520/540 only

Controls repetitive signal acquisition. This is equivalent to setting **Repetitive Signal** in the Acquire menu. When the digitizing oscilloscope is in real-time operation, this setting has no effect.

The ACQuire:REPEt command specifies the behavior of the acquisition system during equivalent-time (ET) operation. When repetitive mode is on, the acquisition system will continue to acquire waveform data until the waveform record is filled with acquired data. When repetitive mode is off and you specify single acquisition operation, only some of the waveform data points will be set with acquired data, and the displayed waveform shows interpolated values for the unsampled data points.

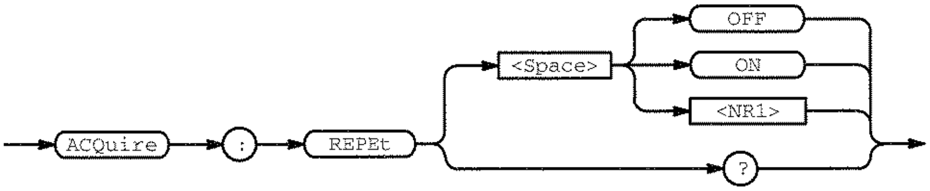
Group: Acquisition

Related Commands: ACQuire:STATE, ACQuire:STOPAfter

Syntax:

```
ACQuire:REPEt { OFF | ON | <NR1> }

ACQuire:REPEt?
```



Arguments: `OFF` or `<NR1>` = 0 turns repetitive mode off.

`ON` or `<NR1>` ≠ 0 turns repetitive mode on.

Examples: `ACQUIRE:REPET 1`
turns repetitive mode on.

`ACQUIRE:REPET OFF`
turns repetitive mode off.

`ACQUIRE:REPET?`
might return `1`, indicating that repetitive signal acquisition mode is on.

ACQuire:STATE

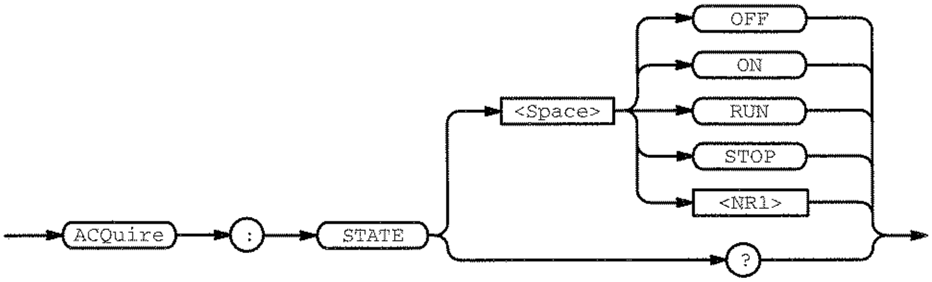
Starts or stops acquisitions. This is the equivalent of pressing the front-panel **RUN/STOP** button. If ACQuire:STOPAfter is set to SEQuence, other signal events may also stop acquisition.

Group: Acquisition

Related Commands: ACQuire:NUMACq?, ACQuire:REPEt, 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 prior to 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 digitizing oscilloscope when to stop taking acquisitions. This is equivalent to setting **Stop After** in the Acquire menu.

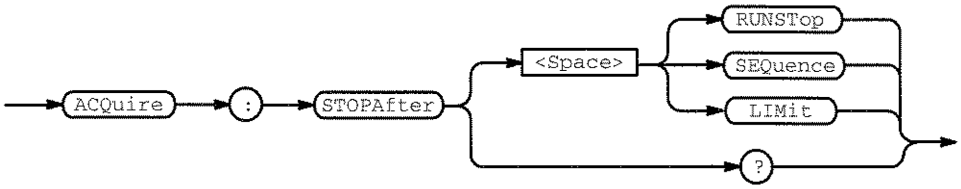
Group: Acquisition

Related Commands: ACQuire:MODe, ACQuire:STATE, ACQuire:REPEt

Syntax:

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

```
ACQuire:STOPAfter?
```



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

SEQuence specifies "single sequence" operation, where the digitizing 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 digitizing oscilloscope will stop after digitizing a waveform from a single trigger event. However, if the acquisition mode is set to average 100 waveforms, then the digitizing oscilloscope will stop only after all 100 waveforms have been acquired. The ACQuire: STATE command and the front-panel **RUN/STOP** button will also stop acquisition when the digitizing oscilloscope is in single sequence mode.

LIMit specifies the digitizing oscilloscope stops after the limit test condition is met.

NOTE

If you set the acquisition system to single sequence, envelope mode, and set the number of envelopes to infinity, the digitizing oscilloscope will envelope a maximum of 2001 acquisitions.

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

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

ALIas

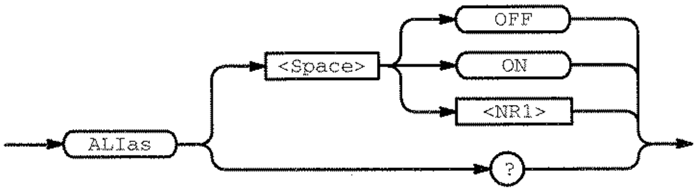
Turns command aliases on or off. This command is identical to the ALIas:STATE command.

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 (110, "Command header error") will be generated.

`ON` or `<NR1>` $\neq$ 0 turns alias expansion on. When a defined alias is received, the specified command sequence is substituted for the alias and executed.

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

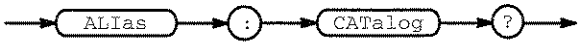
`ALIAS?`
returns `1` when the aliases are on.

ALIas:CATalog? (Query Only)

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

Group: Alias

Syntax: `ALIas:CATalog?`



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

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

ALIas:DEFINE

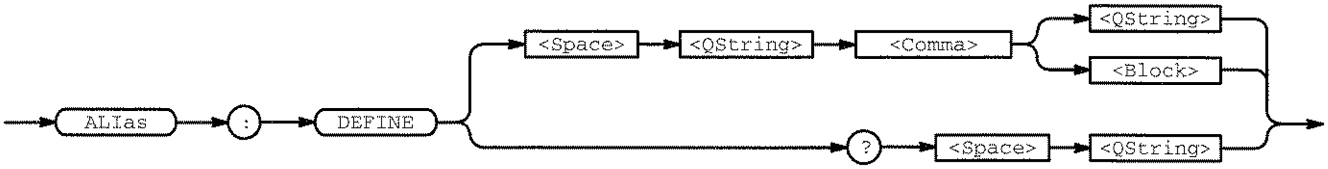
Assigns a sequence of program messages to an alias label. These messages are then substituted for the alias whenever it is received as a command or query provided ALIas:STATE has been turned ON. The ALIas:DEFINE? query returns the definition of a selected alias.

Up to 10 aliases can be defined 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 that must be separated by semicolons and must follow 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 digitizing oscilloscope.

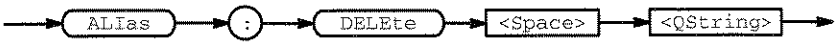
`ALIAS:DEFINE? "ST1"`
might return `:ALIAS:DEFINE "ST1",#239:RECALL:SETUP 5;:AUTOSET EXECUTE;:SELECT:CH1 ON`

ALIas:DELEte (No Query Form)

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

Group: Alias

Syntax: `ALIas:DELEte <QString>`



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

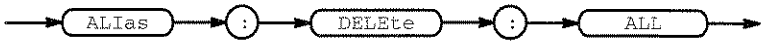
Examples: `ALIAS:DELETE "SETUP1"`
deletes the alias named SETUP1.

ALIas:DELEte:ALL (No Query Form)

Deletes all existing aliases.

Group: Alias

Syntax: `ALIas:DELEte:ALL`



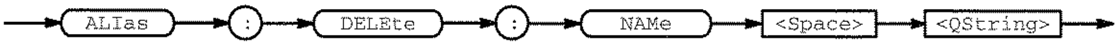
Examples: `ALIAS:DELETE:ALL`
deletes all aliases.

ALIas:DELEte:NAMe (No Query Form)

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

Group: Alias

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



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

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

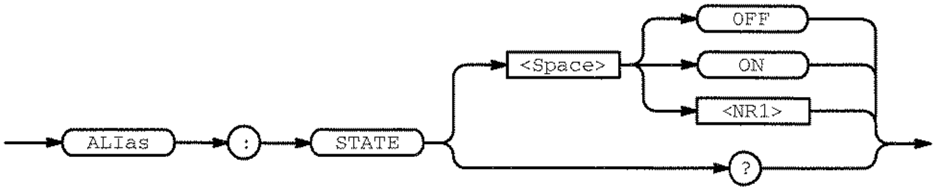
ALIas:STATE

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

Group: Alias

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

`ALIas:STATE?`



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

`ON` or `<NR1>` ≠ 0 turns alias expansion on. When a defined alias is received, the specified command sequence is substituted for the alias and executed.

Examples:

```
ALIAS:STATE OFF
```

turns the command alias feature off.

```
ALIAS:STATE?
```

returns `0` when alias mode is off.

ALLEv? (Query Only)

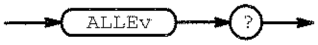
Causes the digitizing oscilloscope to return all events and their messages, and removes the returned events from the Event Queue. The messages are separated by commas. Use the *ESR? query to enable the events to be returned. For a complete discussion of the use of these registers, see page NO TAG. This command is similar to repeatedly sending *EVMsg? queries to the instrument.

Group: Status and error

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

Syntax:

```
ALLEv?
```



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

```
<Event Code><Comma><QString>[<Comma><Event Code><Com-
ma><QString>...]

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

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

Examples:

```
ALLEV?
```

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

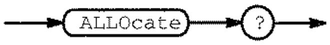
ALLOcate? (Query Only)

Returns the number of data points allocated for all four reference memory locations.

Group: Save and Recall

Syntax:

```
ALLOcate?
```



Examples: `ALLOCATE?`

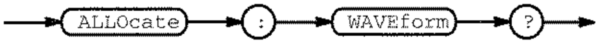
might return `:ALLOCATE:WAVEFORM:REF1 50000;REF2 0;REF3 0; REF4 0;`, indicating that all 50000 data points are allocated to reference memory location 1.

ALLOcate:WAVEform? (Query Only)

Returns the number of data points allocated for all four reference memory locations.

Group: Save and Recall

Syntax: `ALLOcate:WAVEform?`



Examples: `ALLOCATE?`

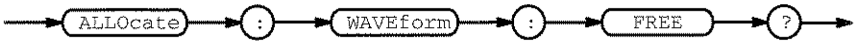
might return `:ALLOCATE:WAVEFORM:REF1 500;REF2 500;REF3 500; REF4 0;`, indicating that 500 data points are allocated to each of the first three reference memory locations.

ALLOcate:WAVEform:FREE? (Query Only)

Returns the approximate number of data points that have not been allocated.

Group: Save and Recall

Syntax: `ALLOcate:WAVEform:FREE?`



Returns: `<NR1>` is the approximate number of data points available.

Examples: `ALLOCATE:WAVEFORM:FREE?`

might return `520` indicating that there are approximately 500 data points available for allocation. The extra 20 are used for administration purposes.

ALLOcate:WAVEform:REF<x>

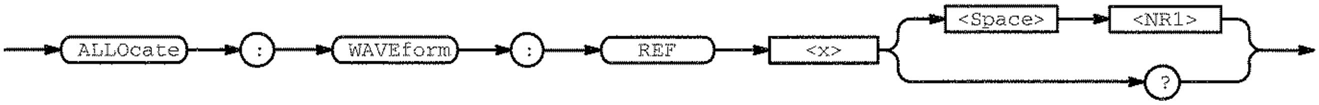
Sets or queries the number of waveform data points for the specified reference location. If an attempt is made to allocate memory when it is not available, an execution error is generated and the memory is not allocated.

Group: Save and Recall

Syntax:

```
ALLOcate:WAVEform:REF<x> <NR1>
```

```
ALLOcate:WAVEform:REF<x>?
```



Arguments: <NR1> = 0 is returned when the reference location is empty.

<NR1> ≠ 0 specifies the number of data points. For the TDS 420/460/520/540 they can be 500, 1000, 2500, 5000, or 15000. The TDS 420/460 Option 1M also allows 30000 or 60000. The TDS 520/540 also allows 50000 with Option 1M. The TDS 620/640 offers 500, 1000, or 2000 samples. All invalid values less than the maximum will be forced to the next highest valid value, and those higher than the maximum will be forced to the maximum. For example, 15002 points on a TDS 540 with option 1M will allocate 50000 points of data for the reference. The memory size of the four TDS 540 reference locations combined cannot exceed 50000 data points.

Examples:

```
ALLOCATE:WAVEFORM:REF2 1000
```

reserves 1,000 data points for REF2.

```
ALLOCATE:WAVEFORM:REF1?
```

might return `500`

APPMenu

Displays the user-definable Application menu and the query returns the current Application menu labels and title. This is equivalent to pressing the front-panel **APPLICATION** button.

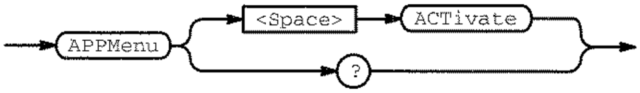
Group: Application Menu

Related Commands: CLEARMenu, *ESR, EVENT?

Syntax:

```
APPMenu ACTivate
```

```
APPMenu?
```



Arguments: ACTivate displays the Application menu. Use the CLEARMenu command to deactivate the Application menu.

Once the Application menu is activated, whenever a front-panel menu button is pressed an event is generated that tells which button was pressed. See page NO TAG for event codes.

Menu button presses will also generate Service Requests when the URQ bit is enabled in DESER and ESER, and the ESB bit is enabled in SRER. See page NO TAG for a complete discussion of the use of these registers.

Examples: `APPMENU ACTIVATE`
displays the application menu.

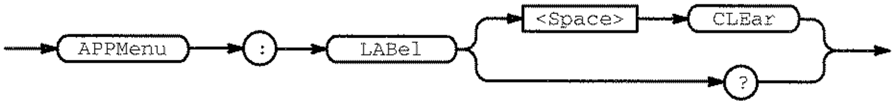
APPMenu:LABel

Removes all user-defined Application menu button labels from the display. The APPMenu:LABel? query returns all the current label settings.

Group: Application Menu

Syntax: `APPMenu:LABel CLEar`

`APPMenu:LABel?`



Arguments: CLEar removes the main and side menu button labels from the display. Front-panel bezel button presses will continue to generate events.

Examples: `APPMENU:LABEL CLEAR`
clears the user-defined menu labels from the display.

APPMenu:LABel:BOTTOM<x>

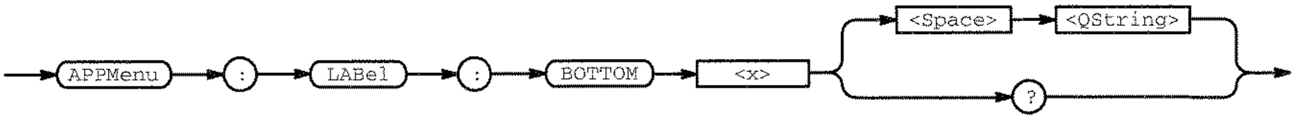
Defines a label for the main menu button that is specified by <x>. Main menu buttons are located along the bottom of the display, and are numbered from 1 to 7 starting with the left-most button.

Group: Application Menu

Syntax:

```
APPMenu:LABel:BOTTOM<x> <QString>

APPMenu:LABel:BOTTOM<x>?
```



Arguments: `<QString>` is the menu button label and can include any of the characters shown in the TDS Character Chart in Appendix A. The maximum length of the label is 1000 characters. The label is displayed in the area above the specified main menu button.

The label is displayed on a single line and is centered, both vertically and horizontally, within the label area. A line feed character can be embedded in the string to position the label on multiple lines. You can also use white space tab characters to position the label within a line.

A tab can be sent by sending a tab character (decimal 9) followed by two numeric characters that specify the pixel column relative to the left margin of the label area.

The ESC @ character turns reverse video on and off, and can be embedded in the label string. The first ESC @ character displays all text following the ESC @ in reverse video until another ESC @ character is found in the string.

NOTE

The use of any undocumented codes may produce unpredictable results.

The label area is 45 pixels high and 90 pixels wide. The length of the label that fits in the label area depends on the contents of the label, because the width of characters varies. The label area is about 10 characters wide and 3 lines high. For a complete list of character widths in pixels, see Table A-1 on page A-1.

If the label exceeds the limits of the label area, either horizontally or vertically, the portion of the label that exceeds the limits will not be displayed. Note: the label itself is not altered. The entire label can be returned as a query response regardless of what is displayed.

Examples:

```
APPMENU:LABEL:BOTTOM3 "SETUP1"
```

assigns the label "SETUP1" to the third main menu button.

APPMenu:LABel:RIGHT<x>

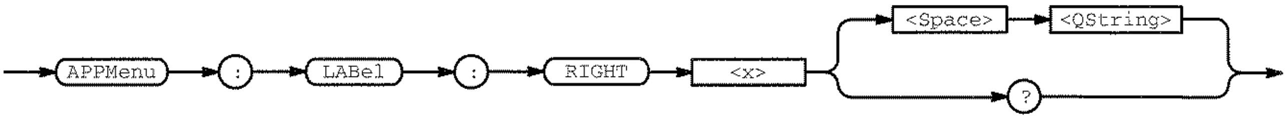
Defines a label for the side menu button that is specified by <x>. Side menu buttons are located on the right side of the display, and are numbered from 1 to 5 starting with the top-most button.

Group: Application Menu

Syntax:

```
APPMenu:LABel:RIGHT<x> <QString>

APPMenu:LABel:RIGHT<x>?
```



Arguments: `<QString>` is the menu button label and can include any of the characters shown in the TDS Character Chart in Appendix A. The maximum length of the label is 1000 characters. The label is displayed in the area to the left of the specified side menu button. Refer to the APPMenu:LABel:BOTTOM<x> command on page 2-48 for more information on defining menu labels.

The label area is 72 pixels high and 112 pixels wide. The length of the label that fits in the label area depends on the contents of the label, because the width of characters varies. The label area is about 12 characters wide and 2 lines high. For a complete list of character widths in pixels, see Table A-1 on page A-1.

Examples:

```
APPMENU:LABEL:RIGHT1 "TEST ON"
```

displays the label "TEST ON" next to the top side menu button.

APPMenu:TITLe

Sets or queries the user-defined application menu title. The title is displayed above the side menu.

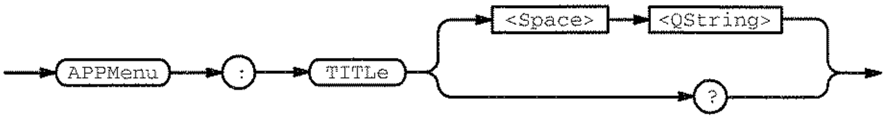
Group: Application Menu

Related Commands: APPMenu, APPMenu:LABel

Syntax:

```
APPMenu:TITLe <QString>

APPMenu:TITLe?
```



Arguments: `<QString>` is the side menu title and can include any of the characters shown in the TDS Character Chart in Appendix A. The maximum length of the title is 1000 characters. The APPMenu:LABel:BOTTOM<x> command on page 2-48 provides information on defining menu labels.

The label area is 40 pixels high and 112 pixels wide. The length of the label that fits in the label area depends on the contents of the label, because the width of characters varies. The label area is about 12 characters wide and 2 lines high. For a complete list of character widths in pixels, see Table A-1 on page A-1.

Examples: `APPMENU:TITLE "Custom Menu"`
displays the title "Custom Menu" on the screen.

`APPMENU:TITLE?`
might return `"Test Setup"` for the current application menu title.

AUTOSet (No Query Form)

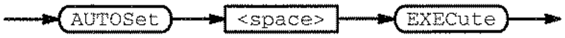
Causes the digitizing oscilloscope to adjust its vertical, horizontal, and trigger controls to provide a stable display of the selected waveform. This is equivalent to pressing the front-panel **AUTOSET** button. For a detailed description of the autoset function, see Autoset in the In Detail section of the Tutorial/User Manual for your instrument.

NOTE

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

Group: Miscellaneous

Syntax: `AUTOSet EXECute`



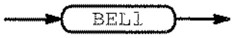
Arguments: `EXECute` autosets the displayed waveform.

BELl (No Query Form)

Beeps the audio indicator of the digitizing oscilloscope.

Group: Miscellaneous

Syntax: `BELl`



Examples: `BELL`

rings the bell.

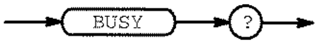
BUSY? (Query Only)

Returns the status of the digitizing oscilloscope. This command allows you to synchronize the operation of the digitizing oscilloscope with your application program. Synchronization methods are described on page NO TAG.

Group: Status and error

Related Commands: *OPC, *WAI

Syntax: `BUSY?`



Returns: `<NR1>` = 0 means that the digitizing oscilloscope is not busy processing a command whose execution time is extensive. These commands are listed in Table 2-21.

`<NR1>` = 1 means that the digitizing oscilloscope is busy processing one of the commands listed in Table 2-21.

Table 2-21: Commands that Affect BUSY? Response

Operation	Command
Single sequence acquisition	ACQuire:STATE ON or ACQuire:STATE RUN (when ACQuire:STOPAfter is set to SEQuence)
Hardcopy output	HARDCopy STARt

Examples: `BUSY?`
might return `1`, indicating that the instrument is busy.

*CAL? (Query Only)

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

NOTE

The self-calibration can take 40 seconds or more on the TDS 520/540 and 60 or more on other TDS to respond. No other commands will be executed until calibration is complete.

Group: Calibration and Diagnostic

Syntax: `*CAL?`

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

`<NR1>` ≠ 0 indicates that the calibration did not complete successfully.

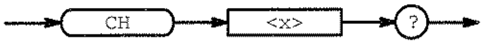
Examples: `*CAL?`
performs an internal self-calibration and might return `0` to indicate that the calibration was successful.

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;IMPEDANCE MEG;BANDWIDTH FULL` for channel 1.

CH<x>:BANdwidth

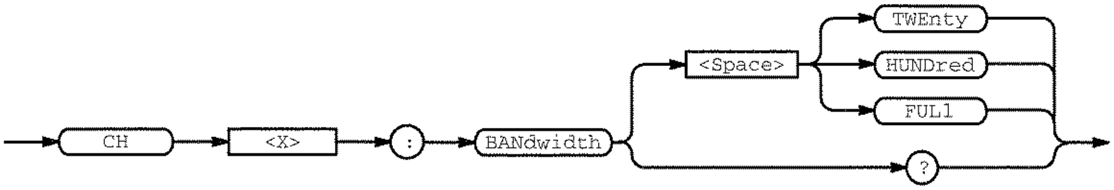
Sets or queries the bandwidth setting of the specified channel. This is equivalent to setting **Bandwidth** in the Vertical menu.

Group: Vertical

Syntax:

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

CH<x>:BANdwidth?
```



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

`HUNDred` sets the channel bandwidth to 100 MHz.

`FULl` sets the channel bandwidth to the full bandwidth of the digitizing 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

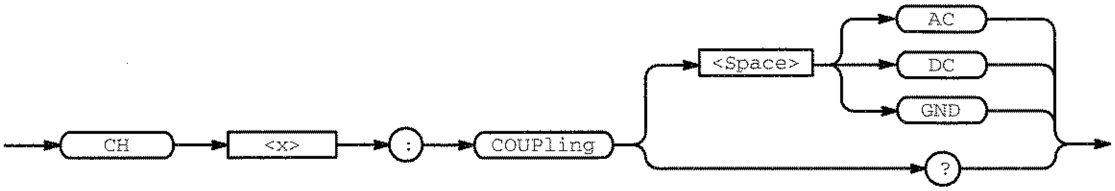
Sets or queries the input attenuator coupling setting of the specified channel. This is equivalent to setting **Coupling** in the Vertical menu.

Group: Vertical

Related Commands: CH<x>:IMPedance

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

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



Arguments: `AC` sets the specified channel to AC coupling.

`DC` sets the specified channel to DC coupling.

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

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

`CH3:COUPLING?`
might return `DC`, indicating that channel 3 is set to DC coupling.

CH<x>:IMPedance

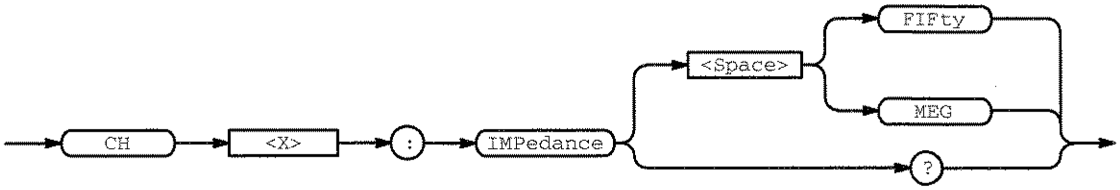
Sets or queries the impedance setting at the specified input channel. This is equivalent to setting the **Impedance** in the Ch<x> Coupling Impedance side menu.

Group: Vertical

Related Commands: CH<x>:COUPling

Syntax: `CH<x>:IMPedance { FIFty | MEG }`

```
CH<x>:IMPedance?
```



Arguments: `FIFty` sets the specified channel to 50 Ω impedance.

`MEG` sets the specified channel to 1 MΩ impedance.

Examples: `CH1:IMPEDANCE FIFty`
establishes 50 Ω impedance on channel 1.

`CH3:IMPEDANCE?`
might return `MEG`, indicating that channel 3 is set to 1 MΩ impedance.

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.

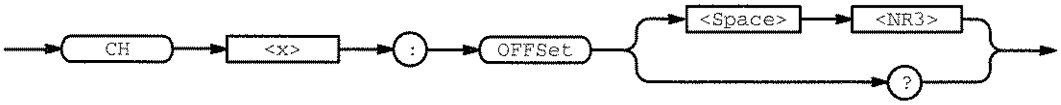
Group: Vertical

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 2-22: Offset Ranges for the TDS 420/460/540/620/640 (All Channels) & TDS 520 (Channel 1 & Channel 2) using a 1x Probe

CH<x>:SCAle	OFFSet Range
1 mV/div – 99.5 mV/div	±1 V
100 mV/div – 995 mV/div	±10 V
1 V/div – 10 V/div	±100 V

Table 2-23: Offset Ranges for the TDS 520 (Aux 1 & Aux 2) using a 1x Probe

CH<x>:SCAle	OFFSet Range
50 mV/div & 100 mV/div	±.5 V
500 mV/div & 1 V/div	±5.0 V
5 V/div & 10 V/div	±50 V

Examples: `CH1:OFFSET 0.5E+00`
lowers the channel 1 displayed waveform by 0.5 volts.

`CH1:OFFSET?`
might return `500.0E-3`, indicating that the current channel 1 offset is 0.5 volts.

CH<x>:POSition

Sets or queries the vertical position of the specified channel. The position 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.

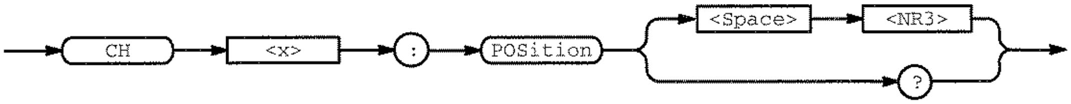
Group: Vertical

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.

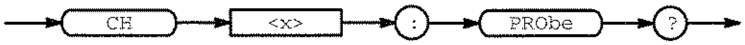
`CH1:POSITION?`
might return `-1.3E+00`, indicating that the current position of channel 1 is at −1.3 divisions.

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: `CH4:PROBE?`
might return `100.0E-3` for a 10x probe.

CH<x>:SCAle

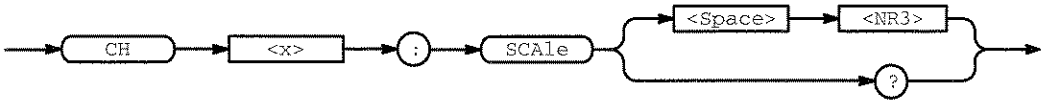
Sets or queries the vertical gain of the specified channel. This is equivalent to setting **Fine Scale** in the Vertical menu or adjusting the front-panel **Vertical SCALE** knob.

Group: Vertical

Related Commands: CH1:VOLts

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

`CH<x>:SCAle?`



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

Examples: `CH4:SCALE 100E-03`
sets the channel 4 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.

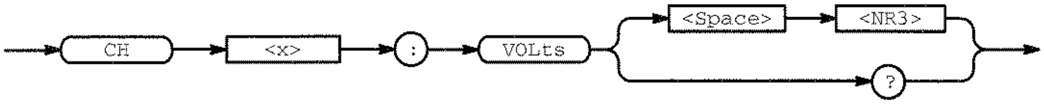
Group: Vertical

Related Commands: CH1:SCAle

Syntax:

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

CH<x>:VOLts?
```



Examples: `CH4:VOLTS 100E-03`
sets the channel 4 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.

*CLS (No Query Form)

Clears the digitizing 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 NO TAG.)

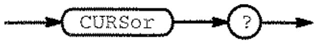
*CLS can suppress a Service Request that is to be generated by an *OPC. This will happen if a hardcopy output or single sequence acquisition operation is still being processed when the *CLS command is executed.

CURSor? (Query Only)

Returns all current cursor settings.

Group: Cursor

Syntax: `CURSor?`



Examples: `CURSOR?`

might return `:CURSOR:FUNCTION OFF;VBARS:UNITS SECONDS; POSITION1 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.

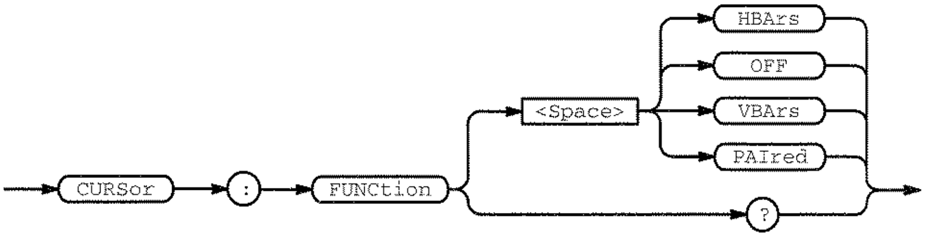
Group: Cursor

Related Commands: SELect:CONTROl

Syntax:

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

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.

PAIred specifies paired cursors that measure 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?

CURSor
:
HBArs
?

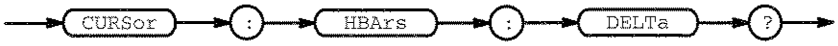
Examples: CURSOR:HBARS?
might return :CURSOR:HBARS:POSITION1 0;POSITION2 0;SELECT CURSOR1.

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>

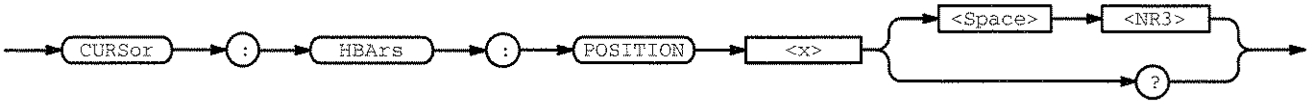
Positions a horizontal bar cursor.

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

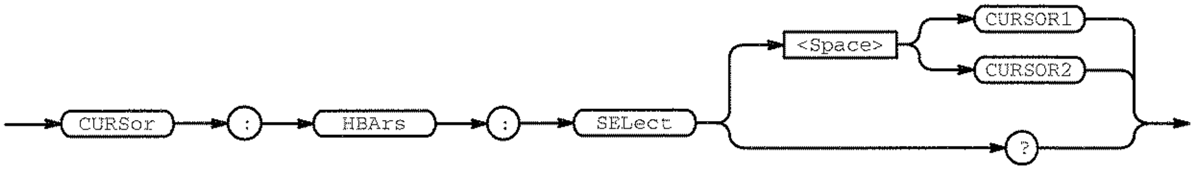
Selects which horizontal bar cursor is active for front-panel control. The active cursor will be displayed as a solid horizontal line and can be moved using the front-panel general purpose knob when the cursor menu is active. The unselected cursor will be displayed as a dashed horizontal line. This command 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.

CURSor:MODe

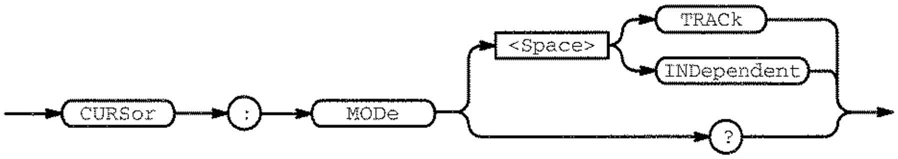
Selects whether the two cursors move together in unison or separately from each other.

Group: Cursor

Related Commands: CURSor:FUNCtion

Syntax: CURSor:MODe{ TRACk | INDependent }

CURSor:MODe?



Arguments: TRACk ties the two cursors together as you move the general purpose knob.

INDependent frees the two cursors to move separately of each other.

Examples: CURSOR:MODE TRACK
specifies that the cursors positions move in unison.

CURSOR:MODE?
might return :TRACK showing the two cursors move in unison.

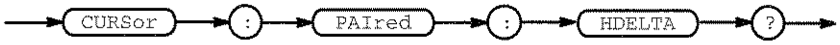
CURSor:PAIred:HDELTA (Query Only)

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?`



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

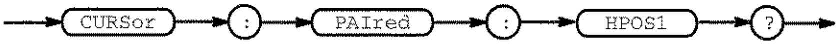
CURSor:PAIred:HPOS1 (Query Only)

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

Group: Cursor

Related Commands: CURSor:FUNCtion

Syntax: `CURSor:PAIred:HPOS1?`



Examples: `CURSOR:PAIRED:HPOS1?`
might return `-64.0E-3`, indicating that the first cursor is at −64.0 mV.

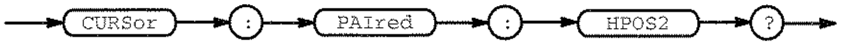
CURSor:PAIred:HPOS2 (Query Only)

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 `-64.0E-3`, indicating the second cursor is at −64.0 mV.

CURSor:PAIred:POSITION1

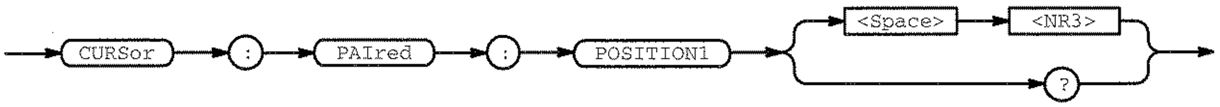
Sets or queries the vertical bar (time) position of the first paired cursor.

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.

CURSor:PAIred:POSITION2

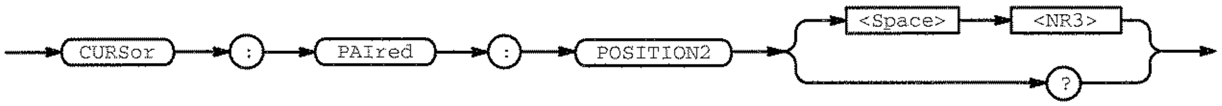
Sets or queries the vertical bar (time) position of the second paired cursor.

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:SELect

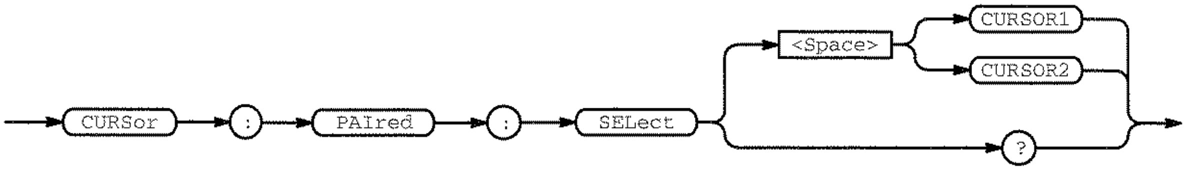
Selects the active paired cursor. The active cursor appears as a solid vertical line. The unselected cursor appears as a dashed vertical line. This command is equivalent to pressing the **TOGGLE** button on the front panel when the Cursor menu is displayed.

Group: Cursor

Syntax:

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

```
CURSor:PAIred:SELect?
```



Arguments: `CURSOR1` specifies the first paired cursor.

`CURSOR2` specifies the second paired cursor.

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

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

CURSor:PAIred:VDELTA (Query Only)

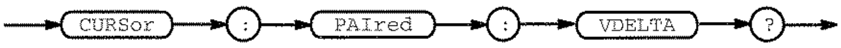
Queries the vbar (time) distance between paired cursors. It returns the absolute value of the first cursor's less the second cursor's horizontal positions.

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.

CURSor:VBArs

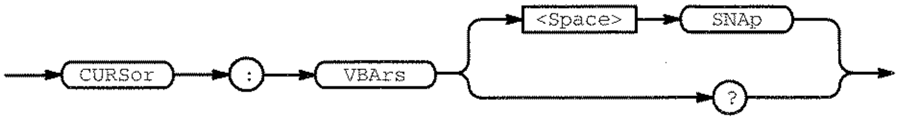
Positions the vertical bar cursors and the CURSor:VBArs? query returns the current vertical bar cursor settings for horizontal position, delta, cursor selection, and units.

Group: Cursor

Related Commands: DATa:STARt, DATa:STOP, MEASUrement:GATing

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.`

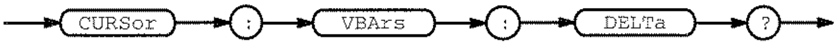
CURSor:VBArs:DELTa? (Query Only)

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?`



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>

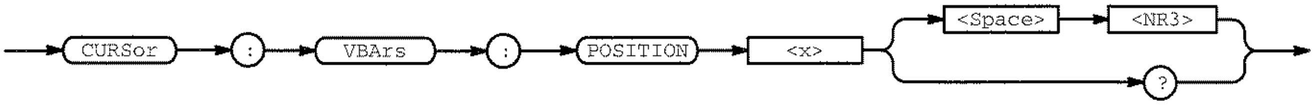
Positions a vertical bar cursor for both vertical bar and paired cursors. The units is specified by the CURSor:VBArs:UNIts command.

Group: Cursor

Related Commands: CURSor:VBArs:UNIts

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

`CURSor:VBArs:POSITION<x>?`



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.

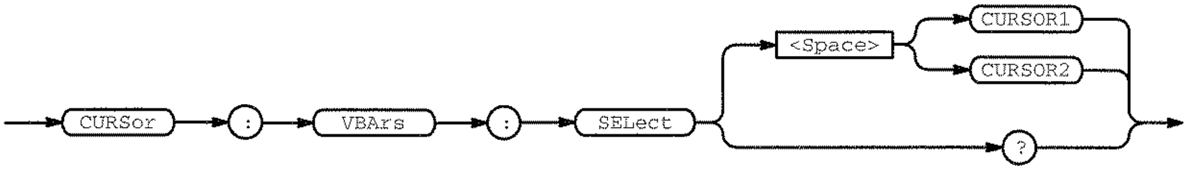
CURSor:VBArs:SELect

Selects which vertical bar cursor is active. The active cursor will be displayed as a solid vertical line and can be moved using the front-panel general purpose knob when the cursor menu is active. The unselected cursor will be 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.

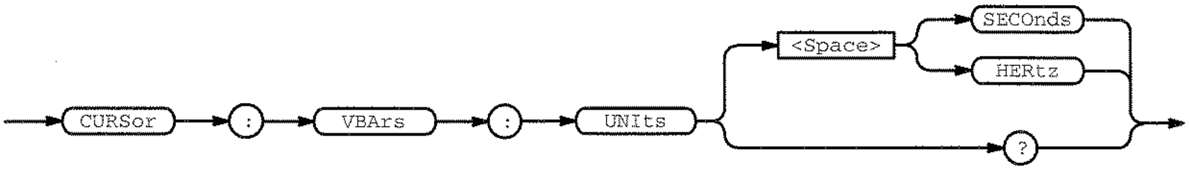
Group: Cursor

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

Syntax:

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

CURSor:VBArs:UNIts?
```



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

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

CURVE

Transfers waveform data to and from the digitizing 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 2-207 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 instrument. 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 instrument. 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 will only be displayed if the reference is displayed.

A description of the waveform transfer process starts on page 2-27.

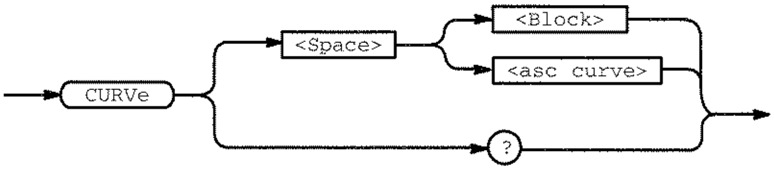
Group: Waveform

Related Commands: DATa, WFMPre

Syntax:

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

CURVe?
```



Arguments: `<Block>` is the waveform data in binary format. The waveform is formatted as: `#<x><yyy><data><newline>` where `<x>` is the number of y bytes. For example, if `<yyy>` = 500, then `<x>` = 3. `<yyy>` is the number of bytes to transfer including checksum. 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.

Examples: `CURVE?`
might return, for ASCII data: `CURVE 0,0,0,0,-1,1,0,-1,0,0,-1,0,0,-1,0,-1, -1,1,0,0,0,-1,0,0,-1,0,1,1,0,-1,0,0,-1,0,0,-1,0,0`

DATa

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

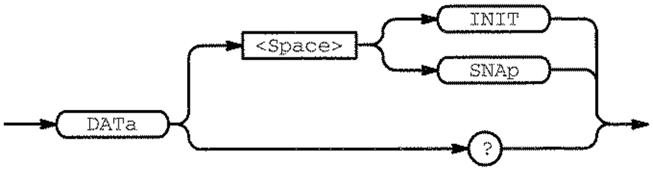
Group: Waveform

Related Commands: CURVE, WAVFrm

Syntax:

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

```
DATa?
```



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

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

Examples: `DATA SNAP`
assigns 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 REF4;START 1;STOP 500;WIDTH 2`

DATa:DESTINATION

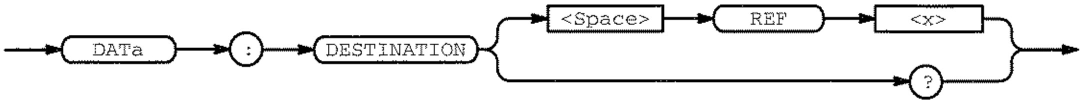
Sets or queries the reference memory location for storing waveform data that is transferred into the digitizing oscilloscope by the CURVe command. This command is identical to the DATa:TARget command.

Group: Waveform

Syntax:

```
DATa:DESTINATION REF<x>

DATa:DESTINATION?
```



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

Examples:

`DATA:DESTINATION REF3`
stores incoming waveform data in reference memory 3.

`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 2-24. Setting the DATa:ENCdg value causes the corresponding WFMPre values to be updated and vice versa.

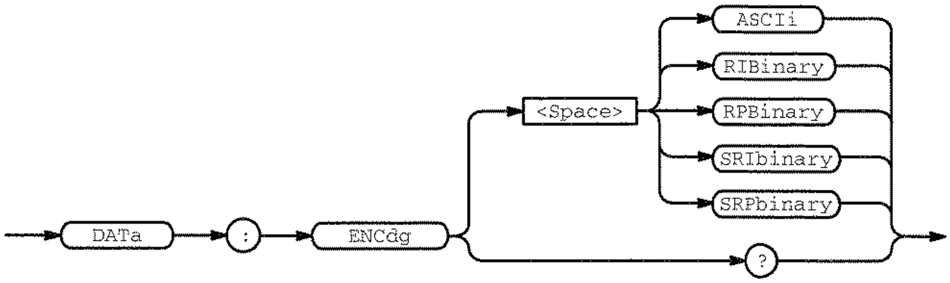
Group: Waveform

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

Syntax:

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

DATa:ENCdg?
```



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

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

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

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

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

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

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 PC's.

Table 2-24: DATa and WFMPre Parameter Settings

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

Examples: `DATA:ENCDG RPBINARY`

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

`DATA:ENCDG?`

might return `SRPBINARY` for the format of the waveform data.

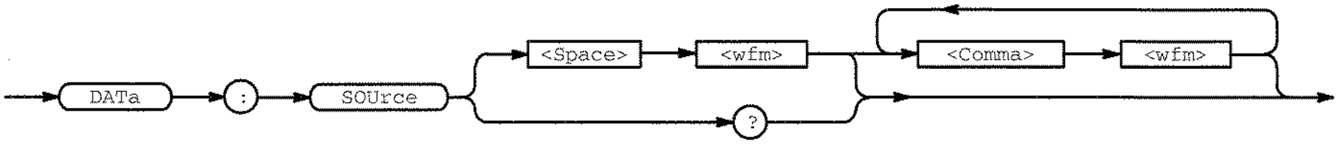
DATa:SOUrce

Sets or queries the location of the waveform data that is transferred from the instrument by the CURVe? query. 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 through CH4, MATH1 through MATH3, then REF1 through REF4.

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 digitizing 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 `REF3`, indicating the source for the waveform data that is transferred using a CURVE? query.

DATa:STARt

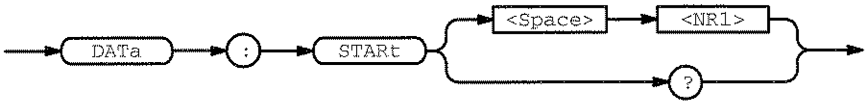
Sets or queries the starting data point for waveform transfer. This command allows for the transfer of partial waveforms to and from the digitizing oscilloscope.

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 will be transferred from `<NR1>` to DATa:STOP or the record length, whichever is less. If `<NR1>` is greater than the record length then no data will be transferred. When DATa:STOP is less than DATa:STARt, the values will be swapped internally for the CURVe? query.

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 the CURVe? query. This allows the transfer of partial waveforms to the controller.

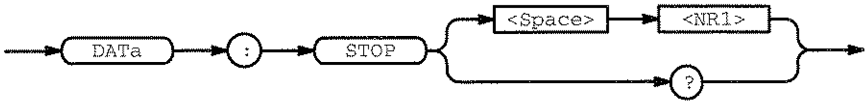
When using the CURVe command the digitizing oscilloscope will stop reading data when there is no more data to read or when the specified record length has been reached so this command will be ignored.

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 be executed. When DATa:STOP is less than DATa:STARt, the values will be swapped internally for the CURVe? query.

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

Examples:

```
DATA:STOP 15000
```

specifies that the waveform transfer will stop at data point 15000.

```
DATA:STOP?
```

might return `14900` as the last data point that will be transferred.

DATa:TARget

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

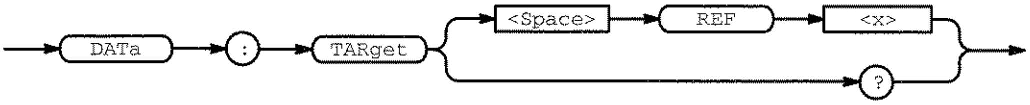
Group: Waveform

Related Commands: CURVe

Syntax:

```
DATa:TARget REF<x>

DATa:TARget?
```



DATa:WIDth

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

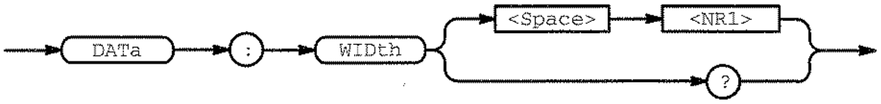
Group: Waveform

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

Syntax:

```
DATa:WIDth <NR1>

DATa:WIDth?
```



Arguments: `<NR1>` = 1 specifies that there is 1 byte (8 bits) per point. This format is useful when the acquisition mode is set to SAMple, ENVelope, or PEAKdetect. If used for AVErage or HIRes, 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 and HIRes waveforms. If used for ENVelope, PEAKdetect, or SAMple, the least significant byte is always zero.

Examples: `DATA:WIDth 1`
sets the data width to 1 byte per data point for CURVe data.

DATE

Sets or queries the date that the digitizing oscilloscope can display.

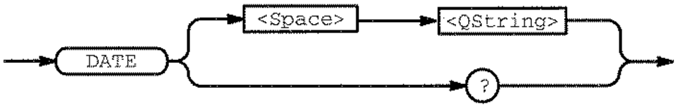
Group: Miscellaneous

Related Commands: DISplay: CLOCk, TIMe

Syntax:

```
DATE <QString>

DATE?
```



Arguments:

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

Examples: `DATE "1993-01-24"`
specifies that the date is set to January 24th, 1993.

*DDT

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

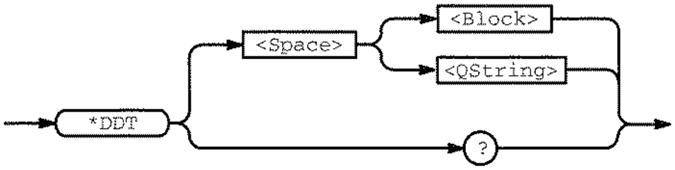
Group: Miscellaneous

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 can 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 #0ACQUIRE:STATE RUN;BELL<EOI>
```

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

DELEte:SETUp (No Query Form)

Removes stored setups from memory and initializes the location with the factory default setup.

NOTE

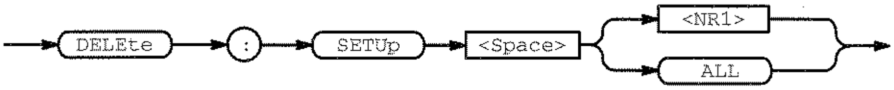
The setup information cannot be recovered once it has been deleted.

Group: Save and Recall

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

Syntax:

```
DELEte:SETUp { <NR1> | ALL }
```



Arguments: `<NR1>` is a value in the range 1 to 10, and specifies a setup storage location. Using an out-of-range value causes an execution error.

`ALL` specifies all the stored setups.

Examples: `DELETE:SETUP ALL`

removes all stored setups. All ten storage locations are initialized to the factory default setup.

DELEte:WAVEform (No Query Form)

Deletes one or all of the stored reference waveforms from memory. The memory allocated for the reference location is then available for reallocation. Memory must be reallocated for the deleted references before any waveform data can be stored in the reference location.

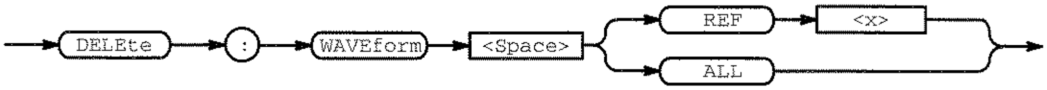
NOTE

The waveform data is not actually cleared from the reference location.

Group: Save and Recall

Related Commands: RECAll:WAVEform, SAVe:WAVEform,TEKSecure

Syntax: `DELEte:WAVEform { REF<x> | ALL }`



Arguments: `<x>` = 1 to 4, and specifies one of the reference memory locations.

`ALL` specifies all the stored waveforms.

Examples: `DELETE:WAVEFORM ALL`

removes all the waveforms stored in reference memory.

`DELETE:WAVEFORM REF2`

removes the waveform stored at REF2.

DESE

Sets and queries the bits in the Device Event Status Enable Register (DESER). The DESER is the mask that determines whether events are reported to the Standard Event Status Register (SESR), and entered into the Event Queue. For a more detailed discussion of the use of these registers, see page NO TAG.

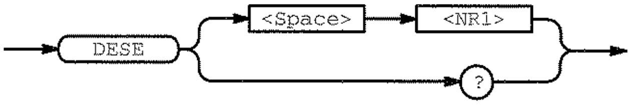
Group: Status and Error

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 the 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, etc.).

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 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 `*ESE` *command to set the ESER. A discussion of event handling begins on page NO TAG.*

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 the DESER contains the binary value 10111010.

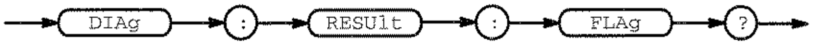
DIAg:RESUlt:FLAg? (Query Only)

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

Group: Calibration and Diagnostic

Related Commands: DIAg:RESUlt:LOG?

Syntax: `DIAg:RESUlt:FLAg?`



Returns: `PASS` indicating that all of the selected diagnostic tests have passed.

`FAIl` indicating that at least one of the selected diagnostic tests have failed.

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

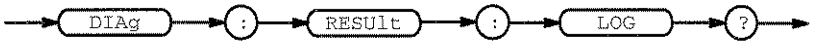
DIAg:RESUlt:LOG? (Query Only)

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

Group: Calibration and Diagnostic

Related Commands: DIAg:RESUlt:FLAg?

Syntax: `DIAg:RESUlt:LOG?`



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

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

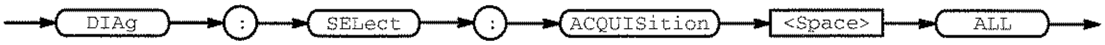
Examples: `DIAG:RESULT:LOG?`
might return `:DIAG:RESULT:LOG "pass--Processor,pass--Display, pass--FP/Proc Interface,FAIL--Front Panel"`

DIAg:SELect:ACQUISition (No Query Form)

Selects the acquisition system test sequence that will be run when the DIAg:STATE EXECUte command is sent. This command is equivalent to setting **Area** in the Utility menu when **System** is set to Diag/Err.

Group: Calibration and Diagnostic

Syntax: `DIAg:SELect:ACQUISition ALL`



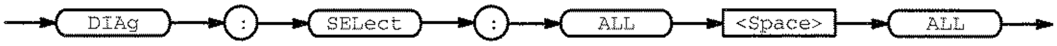
Arguments: `ALL` selects functional, memory and register tests.

DIAg:SELect:ALL (No Query Form)

Specifies that all system test sequences will be run when the DIAg:STATE EXECUte command is sent. This command is equivalent to setting **Area** in the Utility menu when **System** is set to Diag/Err.

Group: Calibration and Diagnostic

Syntax: `DIAg:SELect:ALL ALL`



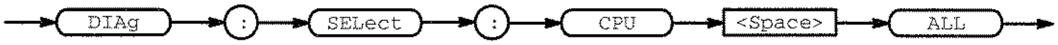
Arguments: `ALL` selects functional, memory, and register tests for the acquisition, processor and display systems, and self diagnostics for the front panel.

DIAg:SELect:CPU (No Query Form)

Selects the processor system test sequence that will be run when the DIAg:STATE EXECUte command is sent. This command is equivalent to setting **Area** in the Utility menu when **System** is set to Diag/Err.

Group: Calibration and Diagnostic

Syntax: `DIAg:SELect:CPU ALL`



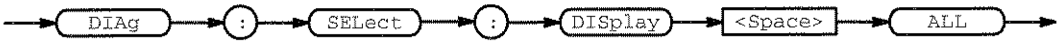
Arguments: `ALL` selects functional, memory and register tests.

DIAg:SELect:DISplay (No Query Form)

Selects the display system test sequence that will be run when the DIAg:STATE EXECUte command is sent. This command is equivalent to setting **Area** in the Utility menu when.**System** is set to Diag/Err.

Group: Calibration and Diagnostic

Syntax: `DIAg:SELect:DISplay ALL`



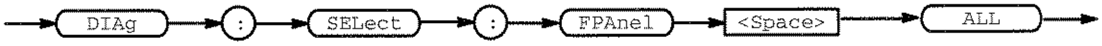
Arguments: `ALL` selects functional, memory and register tests.

DIAg:SELect:FPAnel (No Query Form)

Selects the front-panel test sequence that will be run when the DIAg:STATE EXECUte command is sent. This command is equivalent to setting **Area** in the Utility menu when **System** is set to Diag/Err.

Group: Calibration and Diagnostic

Syntax: `DIAg:SELect:FPAnel ALL`



Arguments: `ALL` selects self diagnostic tests.

DIAg:STATE (No Query Form)

Executes the diagnostic tests that have been specified with the DIAg:SELect commands.

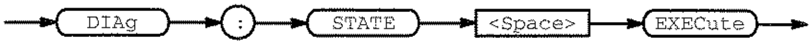
When the test sequence has completed, any of the modules or module interfaces that failed diagnostics are displayed on the screen and stored in an internal log file. The pass/fail status will be returned by the DIAg:RESUlt:FLAg? query and the internal log will be returned by the DIAg:RESUlt:LOG? query.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

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 commands. When complete, the digitizing oscilloscope will return to the state it was in just prior to the test. If the PON event was enabled before running the tests, a Service Request will be generated. When the Service Request has been received, the pass/fail status of the tests can be returned by executing the DIAg:RESUlt:FLAg? query.

The DIAg:STATE EXECute command clears the following:

- the Event Queue
- the Input Queue
- the Status Registers (SESR 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

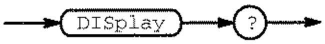
Examples: `DIAg:STATE EXECUTE`
executes all the diagnostic tests that have been selected.

DISplay? (Query Only)

Returns the current display settings.

Group: Display

Syntax: `DISplay?`



Examples: `DISPLAY?`
might return `:DISPLAY:FORMAT YT;STYLE VECTORS;FILTER SINX;PERSISTENCE 500.0E-3;GRATICULE FULL;TRIGT 1;INTENSITY:OVERALL 85;WAVEFORM 70;TEXT 60;CONTRAST 150`

DISplay:CLOCk

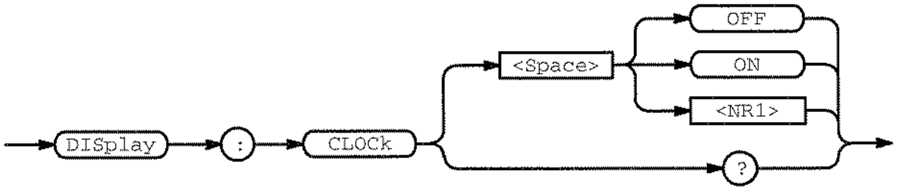
Controls the display of the date and time. This is equivalent to setting the **Display Date/Time** in the Readout Options side menu. The query form returns an ON (1) or an OFF (0).

Group: Display

Syntax:

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

DISplay:CLOCk?
```



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

`<ON>` or `<NR1>` ≠ 0 displays the clock on the display.

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

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

DISplay:FILTer

Sets or queries the type of interpolation to use for the display when the DISplay:STYle is VECtors or DOTs. This command is equivalent to setting **Filter** in the Display menu.

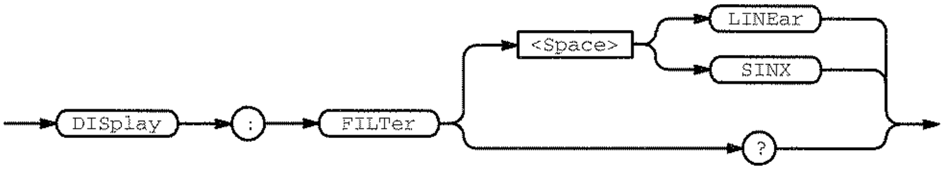
Group: Display

Related Commands: DISplay:STYle

Syntax:

```
DISplay:FILTer { LINEar | SINX }

DISplay:FILTer?
```



Arguments: `LINEar` specifies linear interpolation where acquired points are connected with straight lines.

`SINX` specifies sin(x)/x interpolation where acquired points are fit to a curve.

Examples: `DISPLAY:FILTER LINEAR`
sets the interpolation filter type to linear.

`DISPLAY:FILTER?`
returns either `LINEAR` or `SINX`, indicating the type of interpolation filter.

DISplay:FORMat

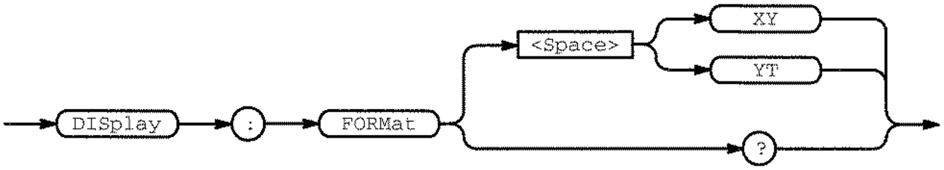
Sets or queries the display format. This command is equivalent to setting **Format** in the Display menu.

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 2-25. Displaying one source causes its corresponding source to be displayed.

Table 2-25: XY Format Pairs

X-Axis Source	Y-Axis Source
Ch 1	Ch 2
Ch 3 (or AUX 1)	Ch 4 (or AUX 2)
Ref 1	Ref 2
Ref 3	Ref 4

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

Examples:

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

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

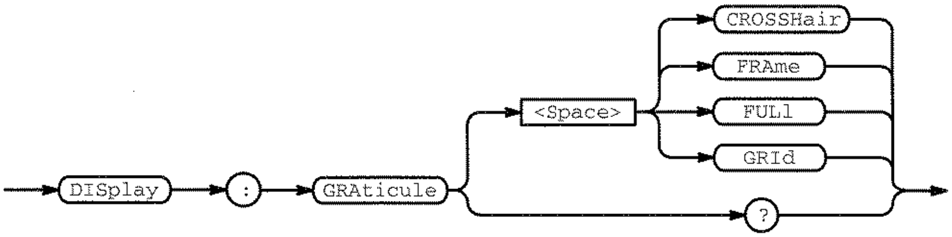
DISplay:GRAticule

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

Group: Display

Syntax:

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



Arguments: `CROSSHair` specifies a frame and cross hairs.

`FRAme` specifies just a frame.

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

`GRId` specifies a frame and a grid.

Examples:

```
DISPLAY:GRATICULE GRID
```
sets the graticule type to display a frame and a grid.

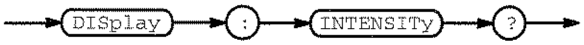
`DISPLAY:GRATICULE?`
returns `FULL` when all graticule elements (grid, frame, and cross hairs) are selected.

DISplay:INTENSITy? (Query Only)

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

Group: Display

Syntax: `DISplay:INTENSITy?`



Examples: `DISPLAY:INTENSITY?`
might return `:DISPLAY:INTENSITY:OVERALL 85;WAVEFORM 70;TEXT 60;CONTRAST 150`

DISplay:INTENSITy:CONTRast

Sets the intensity of the intensified zone on a waveform. This command is equivalent to setting **Contrast** in the Display Intensity side menu.

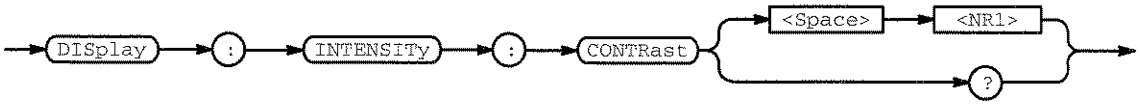
The command has no effect on limit test templates or intensified samples. They are displayed at a fixed contrast ratio.

Group: Display

Related Commands: HORizontal:MODe

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

`DISplay:INTENSITy:CONTRast?`



Arguments: `<NR1>` ranges from 100 to 250 percent.

Examples: `DISPLAY:INTENSITY:CONTRAST 140`
sets the intensity of the intensified portion of a waveform.

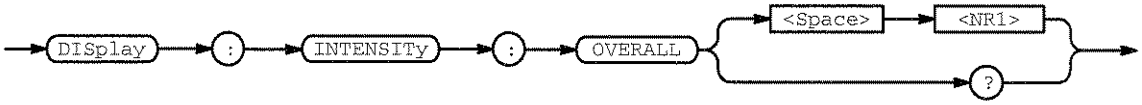
DISplay:INTENSITy:OVERALL

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 percent.

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.

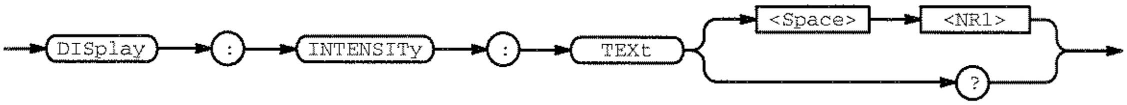
DISplay:INTENSITy:TEXt

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 <NR1>`

`DISplay:INTENSITy:TEXt?`



Arguments: `<NR1>` ranges from 20 to 100 percent.

Examples: `DISPLAY:INTENSITY:TEXT 100`
sets the intensity of the text to the brightest level.

DISplay:INTENSITy:WAVEform

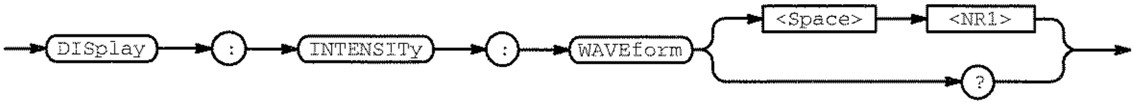
Sets the intensity of the waveforms. This command is equivalent to setting **Waveform** in the Display Intensity side menu.

Group: Display

Syntax:

```
DISplay:INTENSITy:WAVEform <NR1>

DISplay:INTENSITy:WAVEform?
```



Arguments: <NR1> ranges from 20 to 100 percent.

Examples: DISPLAY:INTENSITY:WAVEFORM?
might return 60 as the intensity of the waveform.

DISplay:PERSistence

Sets the length of time that data points are displayed when DISplay:STYle is set to VARpersist. This affects the display only and is equivalent to setting **Variable Persistence** in the Display Style side menu.

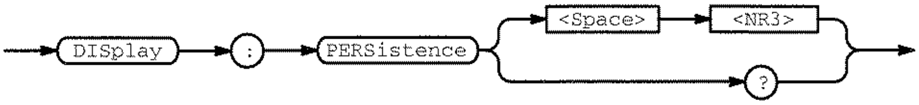
Group: Display

Related Commands: DISplay:STYle

Syntax:

```
DISplay:PERSistence <NR3>

DISplay:PERSistence?
```



Arguments: <NR3> specifies the length, in seconds, that the waveform points are displayed on the screen. The range is 250 ms to 10 s.

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

DISplay:STYle

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

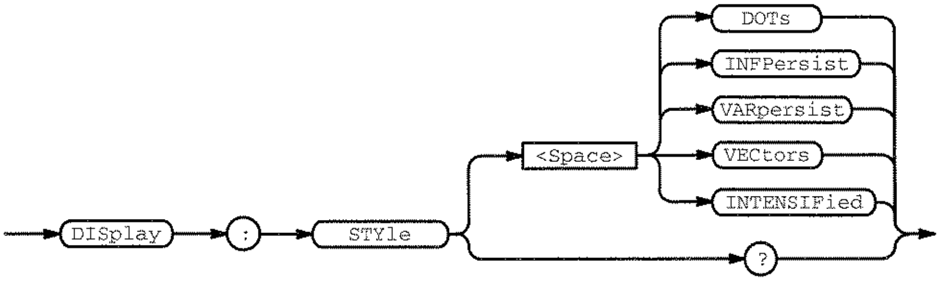
Group: Display

Related Commands: DISplay:PERSistence

Syntax:

```
DISplay:STYle { DOTs | INFPersist | VARpersist | VECtors
    | INTENSIFied }
```

```
DISplay:STYle?
```



Arguments: `DOTs` displays individual data points.

`INFPersist` accumulates data points on the display indefinitely. The display is reset when the style or acquisition is reset.

`VARpersist` leaves acquired data points on the display for a period of time specified by DISplay:PERSistence.

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

`INTENSIFied` causes the display to show acquired (non-interpolated) samples with brighter dots than the rest of the waveform.

Examples:

`DISPLAY:STYLE INFPERSIST`
sets the display to indefinitely accumulate data points on the screen.

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

DISplay:TRIGT

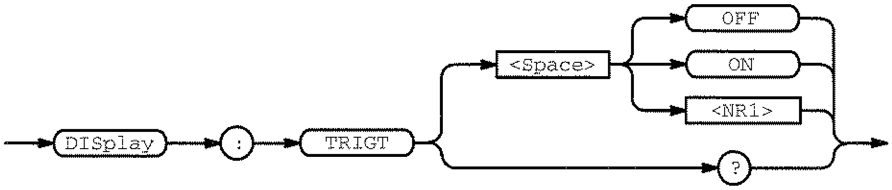
Controls the display of the trigger indicator. This is equivalent to setting the **Display 'T' @ Trigger Point** in the Readout Options side menu. The query form returns an ON (1) or an OFF (0).

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.

DISplay:TRIGBar

Controls the display of the trigger bar indicator/s. The bar indicates where the trigger will occur, in voltage.

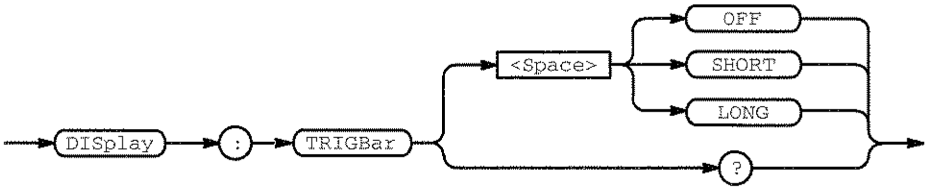
The digitizing oscilloscope will only display the bar if the trigger source is also displayed. If both a main and a delayed trigger are displayed, then two bars will appear. One will accompany each source. If a logic trigger is selected, then multiple bars may appear. If a runt pulse trigger is selected, then two bars may appear. One will show the upper threshold and one the lower threshold.

Group: Display

Syntax:

```
DISplay:TRIGBar { OFF | SHORT | LONG }

DISplay:TRIGBar?
```



Arguments: OFF removes the trigger bar indicator from the display.

SHORT displays a short arrow at the right side of the graticule for each displayed trigger signal.

LONG displays a horizontal line in the center of the graticule for each displayed trigger signal.

Examples: `DISPLAY:TRIGBAR LONG`
sets the display to show long trigger bar indicator (or 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 NO TAG.

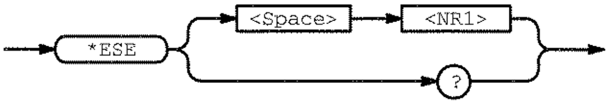
Group: Status and Error

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 NO TAG.*

Examples: *ESE 209

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

*ESE?

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

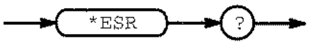
*ESR? (Query Only)

Returns the contents of the Standard Event Status Register (SESR). *ESR? also clears the SESR (since reading the SESR clears it). For a more detailed discussion of the use of these registers, see page NO TAG.

Group: Status and Error

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

Syntax: *ESR?



Examples: *ESR?

might return the value 213, showing that the SESR contains binary 11010101.

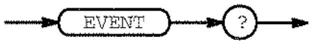
EVENT? (Query Only)

Returns from the Event Queue an event code that provides information about the results of the last *ESR? read. EVENT? also removes the returned value from the Event Queue. A discussion of event handling begins on page NO TAG.

Group: Status and Error

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

Syntax: `EVENT?`



Examples: `EVENT?`

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

EVMsg? (Query Only)

Removes from the Event Queue a single event code associated with the results of the last *ESR? read, and returns the event code along with an explanatory message. A more detailed discussion of event handling begins on page NO TAG.

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><Com-
ma><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 digitizing oscilloscope. As much of the command will be returned as possible without exceeding the 60 character limit of the `<Message>` and `<Command>` strings combined. The command string is right-justified.

Examples: `EVMSG?`

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

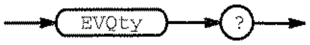
EVQty? (Query Only)

Returns the number of event codes that are in the Event Queue. This is useful when using the ALLEv? query 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.

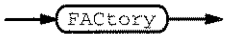
FACtory (No Query Form)

Resets the digitizing oscilloscope to its factory default settings. This command is equivalent to selecting **Recall Factory Setup** in the Waveform Save/Recall menu.

Group: Miscellaneous

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

Syntax: `FACtory`



Setting the digitizing oscilloscope to factory default includes:

- 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).
- Set the macro defined by *DDT to a “zero-length field.”
- Clear the pending operation flag and associated operations.

The FACtory command does not alter the following:

- The state of the GPIB (IEEE Std 488.1-1987) interface.
- The selected GPIB address.
- Calibration data that affects device specifications.

- Protected user data – reference waveforms and setups.
- Stored settings.
- The current password (if implemented).

HARDCopy

Sends a copy of the screen display followed by and 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.

The HARDCopy? query returns format, layout and port information.

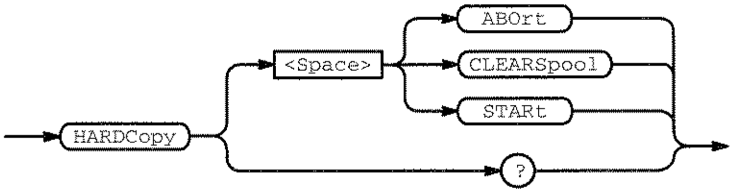
NOTE

This command is NOT IEEE Std 488.2-1987 compatible.

Group: Hardcopy

Syntax:

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



Arguments: `ABOrt` terminates the hardcopy output in process.

NOTE

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

`CLEARSpool` clears the printer output spooler. Specifically, it clears the oscilloscope's spooler. However, the printer may have its own buffer. The printer may continue to print from there even after the oscilloscope acts on this command.

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

NOTE

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

Examples: `HARDCOPY ABORT`
stops any hardcopy output that is in process.

HARDCopy:FORMat

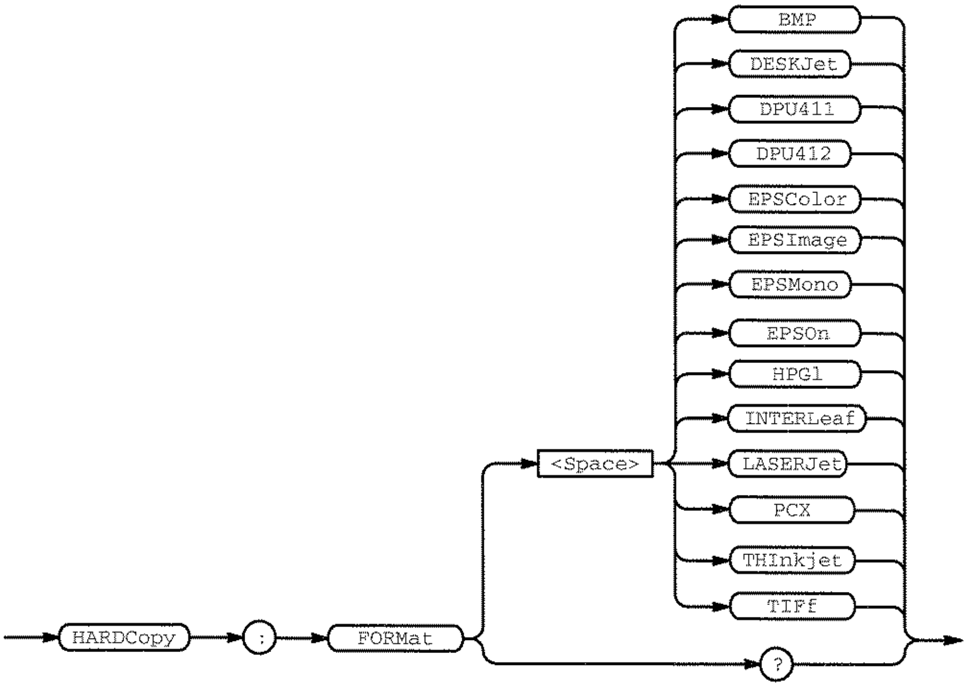
Selects the output data format for hardcopies. This is equivalent to setting **Format** in the Hardcopy menu.

Group: Hardcopy

Syntax:

```
HARDCopy:FORMat { BMP | DESKJet | DPU411 | DPU412 | EPS-
    Color | EPSImage | EPSMono | EPSOn | HPGl | INTERLeaf
    | LASERJet | PCX | THInkjet | TIFf }
```

```
HARDCopy:FORMat?
```



Examples: `HARDCOPY:FORMAT HPGL`
sets the hardcopy output format to HPGL.

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

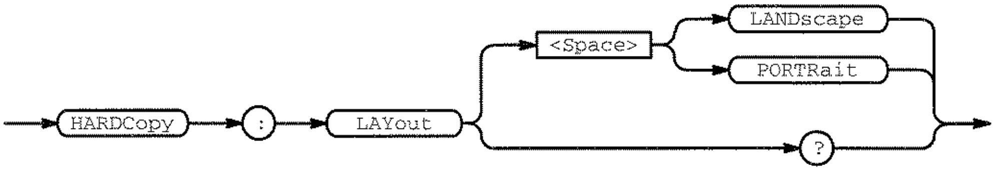
HARDCopy:LAYout

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 longest side of the page.

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

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

HARDCopy:PORT

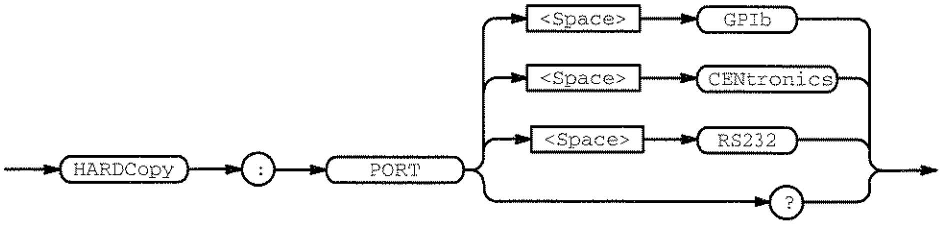
Selects the output port for the printer. This is equivalent to setting **Port** in the Hardcopy menu.

Group: Hardcopy

Related Commands: HARDCopy

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

`HARDCopy:PORT?`



`GPIb` specifies that the hardcopy is sent out the GPIB port.

`CENtronics` specifies that the hardcopy is sent out the Centronics port.

`RS232` specifies that the hardcopy is sent out the RS232 port.

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

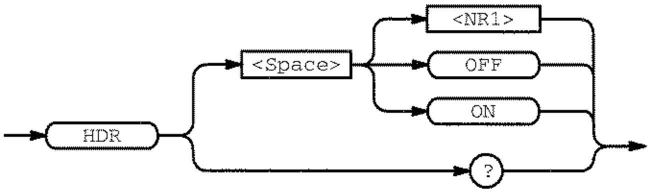
HDR

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?
```



HEADer

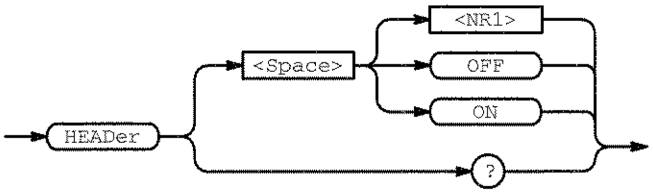
Sets and queries the Response Header Enable State that causes the digitizing oscilloscope to either include or omit headers on query responses. This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk); they never return headers.

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 digitizing 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 digitizing oscilloscope to omit headers on query responses, so that only the argument is returned.

Examples: HEADER OFF

causes the digitizing oscilloscope to omit headers from query responses.

HEADER?

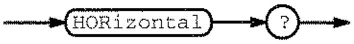
might return the value 1, showing that the Response Header Enable State is true.

HORizontal? (Query Only)

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

Group: Horizontal

Syntax: HORizontal?



Examples: HORIZONTAL?

might return the string :HORIZONTAL:MODE MAIN;RECORDLENGTH 500; POSITION 5.0E+0;TRIGGER:POSITION 50;:HORIZONTAL:MAIN:SCALE 1.0E-6;:HORIZONTAL:DELAY:MODE RUNSAFTER;SCALE 1.0E-6;TIME: 16.0E-9

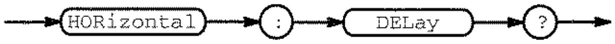
HORizontal:DELay? (Query Only)

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

Group: Horizontal

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

Syntax: `HORizontal:DELay?`



Examples: `HORIZONTAL:DELAY?`

might return the delay parameters `:HORIZONTAL:DELAY:MODE RUNSAFTER;SCALE 1.0E-6;TIME: 16.0E-9`

HORizontal:DELay:MODe

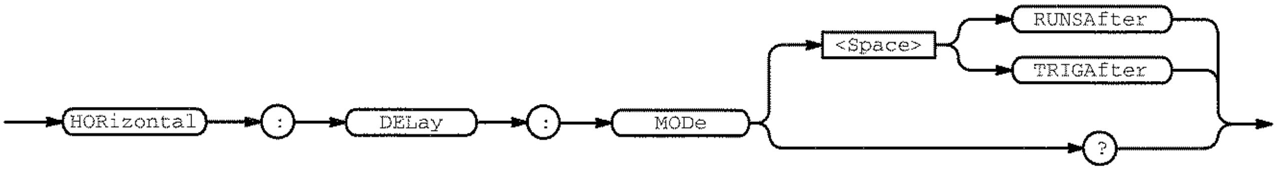
Selects the mode for the delayed time base. This is equivalent to setting **Time Base** in the Horizontal menu.

Group: Horizontal

Related Commands: HORizontal:DELay:TIMe

Syntax: `HORizontal:DELay:MODe { RUNSAfter | TRIGAfter }`

`HORizontal:DELay:MODe?`



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

`TRIGAfter` specifies that the delayed time base is triggerable after the main time base triggers.

Examples: `HORIZONTAL:DELAY:MODE?`

returns either `RUNSAFTER` or `TRIGAFTER`, indicating the delayed time base mode.

HORizontal:DELay:SCAle

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

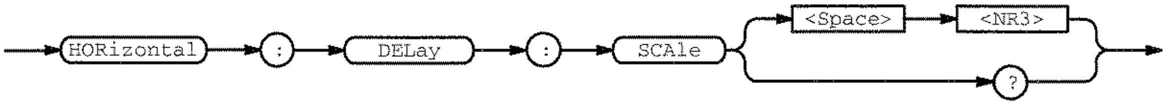
Group: Horizontal

Related Commands: HORizontal:DELay:SECdiv

Syntax:

```
HORizontal:DELay:SCAle <NR3>

HORizontal:DELay:SCAle?
```



Arguments: `<NR3>` is the time per division. The range is 10 s (5 s on the TDS 620/640 and 20 s on the TDS 420/460) to 500 ps (1 ns on the TDS 420/460) in a 1-2-5 sequence. Values that are not in a 1-2-5 sequence (1-2.5-5 on the TDS 620/640) will be 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 will be 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 (1-2.5-5 on the TDS 620/640), 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

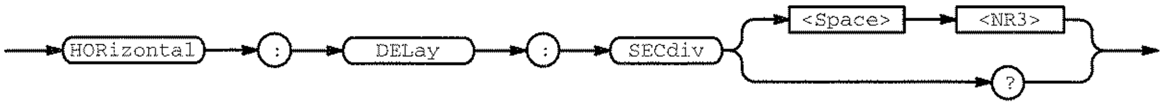
This command is identical to the HORizontal:DELay:SCAle command. It is provided to maintain program compatibility with some older models of Tektronix digitizing oscilloscopes.

Group: Horizontal

Syntax:

```
HORizontal:DELay:SECdiv <NR3>

HORizontal:DELay:SECdiv?
```



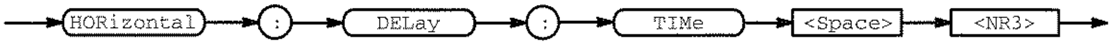
HORizontal:DELay:TIMe

Sets or queries the delay time to wait after the main trigger before the delayed time base begins. This is equivalent to setting **Delayed Runs After Main** in the Horizontal menu's **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 on the TDS 520/540/620/640 is from 16 ns to 250 seconds with a resolution of 4 ns.The range on the TDS 420/460 is from 10 ns to 20 seconds with a resolution of 10 ns.

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

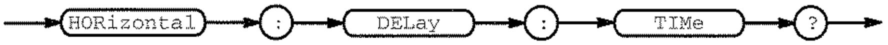
HORizontal:DELay:TIMe? (Query Only)

Returns the delay time parameters.

Group: Horizontal

Related Commands: HORizontal:DELay:TIMe:RUNSAfter?, HORizontal:DELay:TIMe:TRIGAfter?

Syntax: `HORizontal:DELay:TIMe?`



Examples: `HORIZONTAL:DELAY:TIME?`
might return `:HORIZONTAL:DELAY:TIME:16.0E-9` for the delay time.

HORizontal:DELay:TIMe:RUNSAfter

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

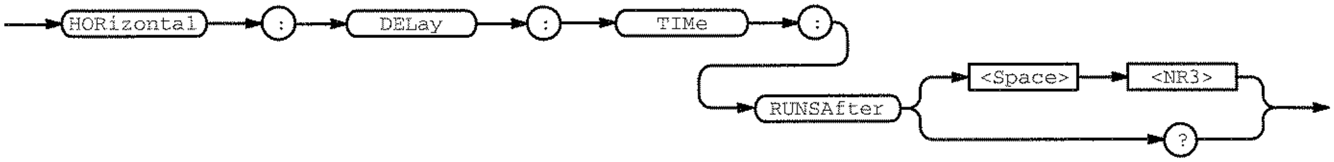
Group: Horizontal

Related Commands: HORizontal:DELay:MODe

Syntax:

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

HORizontal:DELay:TIMe:RUNSAfter?
```



Arguments: <NR3> is the time, in seconds, between the main trigger and the delayed trigger. The range is from 10 ns on the TDS 420/460 or 16 ns on the TDS 520/540/620/640 to 250 seconds (20 s on the TDS420/460) with a resolution of 4 ns.

Examples:

```
HORIZONTAL:DELAY:TIME:RUNSAFTER 2.0E-3
```

sets the delay time between the main and delayed time base to 2 ms.

HORizontal:DELay:TIMe:TRIGAfter

Sets the delay time to wait in the trigger after delay mode. This is the amount of time that must pass before a delayed trigger is accepted. This command is equivalent to setting **Delay by Time** time in the Delayed Trigger menu.

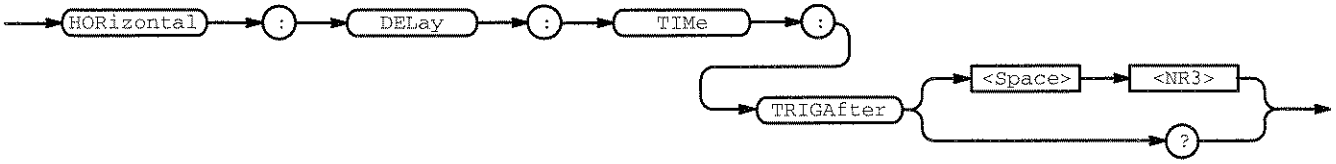
Group: Horizontal

Related Commands: HORizontal:DELay:MODe

Syntax:

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

HORizontal:DELay:TIMe:TRIGAfter?
```



Arguments: <NR3> is the delay time, in seconds. The range on the TDS 520/540/620/640 is from 16 ns to 250 seconds with a resolution of 4 ns.The range on the TDS 420/460 is from 60 ns to 20 seconds with a resolution of 10 ns down to 110 ns.

Examples: `HORIZONTAL:DELAY:TIME:TRIGAFTER 4.0E-6` sets the delay time to 4 µs.

`HORIZONTAL:DELAY:TIME:TRIGAFTER?` might return `1.000E-3`, indicating that the delay time is 1 ms.

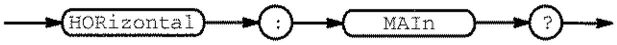
HORizontal:MAIn? (Query Only)

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 will be returned.

Group: Horizontal

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

Syntax: `HORizontal:MAIn?`



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

HORizontal:MAIn:SCAle

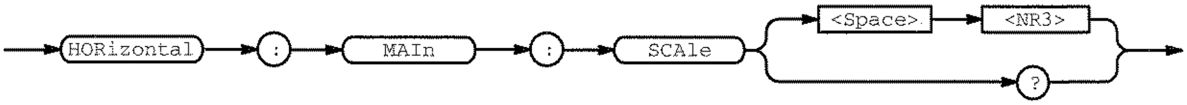
Sets the time per division for the main time base. This command is equivalent to setting **Main Scale** in the Horizontal Scale side menu.

Group: Horizontal

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

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

`HORizontal:MAIn:SCAle?`



Arguments: <NR3> is the time per division. For the TDS 620/640, the range is 5 s to 500 ps in a 1-2.5-5 sequence. For the TDS 520/540, the range is 10 s to 500 ps, in a 1-2-5 sequence. For the TDS 400 series, the range is 20 s to 1 ns.

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 digitizing oscilloscopes.

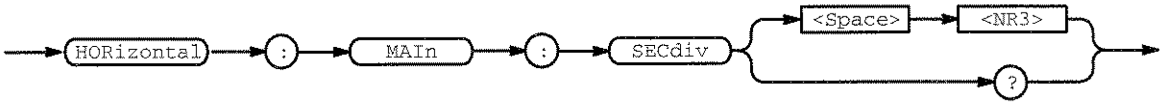
Group: Horizontal

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

Syntax:

```
HORizontal:MAIn:SECdiv <NR3>

HORizontal:MAIn:SECdiv?
```



HORizontal:MODe

Selects whether the horizontal display uses the main or delayed time base or both. This command is equivalent to setting **Time Base** in the Horizontal menu.

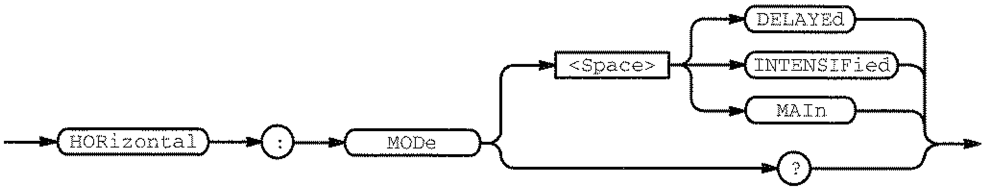
Group: Horizontal

Related Commands: DISplay:INTENSITy:CONTRast

Syntax:

```
HORizontal:MODe { DELAYEd | INTENSIFied | MAIn }

HORizontal:MODe?
```



Arguments: `DELAYEd` means that the selected waveform is horizontally scaled relative to the delayed time base.

`INTENSIFied` uses both the main and delay scales to display the waveform. The portion of the waveform that would be displayed in DELAYEd mode is intensified. 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

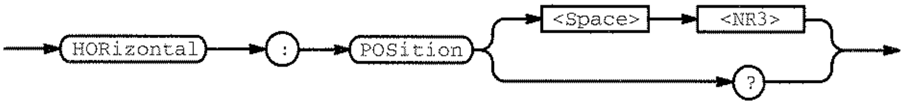
Positions the waveform horizontally on the display. This is used for both main and delayed time bases. This command is equivalent to adjusting the front-panel **Horizontal Position** knob or setting the position in the Horizontal Position side menu.

Group: Horizontal

Syntax:

```
HORizontal:POSition <NR3>

HORizontal:POSition?
```



Arguments: `<NR3>` is from 0 to 100, and is the percent of the waveform that is displayed left of the center graticule.

Examples: `HORIZONTAL:POSITION 10`
sets the horizontal position of the waveform such that 10% of the waveform is to the left of screen center.

HORizontal:RECOrdlength

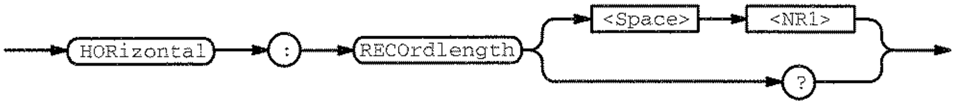
Sets the number of data points that are acquired for each record. This is equivalent to setting **Record Length** in the Horizontal menu.

Group: Horizontal

Syntax:

```
HORizontal:RECOrdlength <NR1>
```

```
HORizontal:RECOrdlength?
```



Arguments: For the TDS 420/460, `<NR1>` is 500, 1000, 2500, 5000, 15000.
For the TDS 520/540, `<NR1>` is 500, 1000, 2500, 5000, 15000.
For the TDS 620/640, `<NR1>` is 500, 1000, 2000.
If you have the TDS 420/460 option 1M, `<NR1>` can also be 30000 or 60000.
If you have the TDS 520/540 option 1M, `<NR1>` can also be 50000.

Examples:

```
HORIZONTAL:RECORDLENGTH 2500
```

specifies that 2500 data points will be acquired for each record.

```
HORIZONTAL:RECORDLENGTH?
```

might return `15000` as the number of data points per record.

HORizontal:SCAle

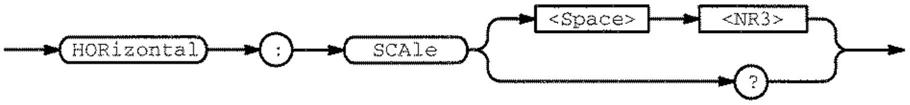
Sets the time per division for the main time base and is identical to the HORizontal:MAIn:SCAle command. It is included here for compatibility purposes.

Group: Horizontal

Syntax:

```
HORizontal:SCAle <NR3>
```

```
HORizontal:SCAle?
```



HORizontal:SECdiv

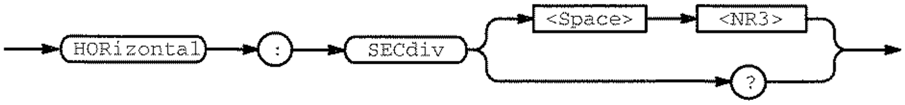
Sets the time per division for the main time base and is identical to the HORizontal:MAIn:SCAle command. It is included here for compatibility purposes.

Group: Horizontal

Syntax:

```
HORizontal:SECdiv <NR3>

HORizontal:SECdiv?
```



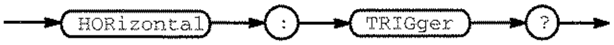
HORizontal:TRIGger? (Query Only)

Returns the horizontal trigger parameter.

Group: Horizontal

Syntax:

```
HORizontal:TRIGger?
```



Examples: `HORIZONTAL:TRIGGER?`
might return `:HORIZONTAL:TRIGGER:POSITION 50`.

HORizontal:TRIGger:POSition

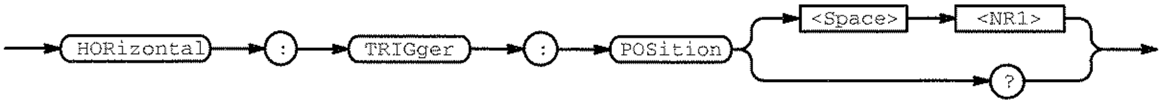
Sets or queries the position of the trigger. This is equivalent to setting **Trigger Position** in the Horizontal menu.

Group: Horizontal

Syntax:

```
HORizontal:TRIGger:POSition <NR1>

HORizontal:TRIGger:POSition?
```



Arguments: `<NR1>` is from 0 to 100 %, (20% to 80% in the TDS 620/640) and is the amount of pretrigger information in the waveform.

Examples: `HORIZONTAL:TRIGGER:POSITION?`
might return `50`.

ID? (Query Only)

Returns identifying information about the instrument and its firmware.

Group: Status and Error

Related Commands: *IDN?

Syntax: `ID?`

Returns: The instrument id in the following format:

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

Examples: `ID?`
might return `TEK/TDS540,CF:91.1CT,FV:2.0`

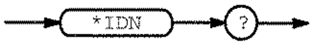
*IDN? (Query Only)

Returns the digitizing oscilloscope's unique identification code.

Group: Miscellaneous

Related Commands: ID

Syntax: `*IDN?`



Returns: The instrument id in the following format:

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

Examples: `*IDN?`
might return the response
`TEKTRONIX,TDS540,0,CF:91.1CT FV:2.0`

LIMit:BELl

Rings the bell when the waveform data exceeds the limits set in the limit test, if the limit state is on.

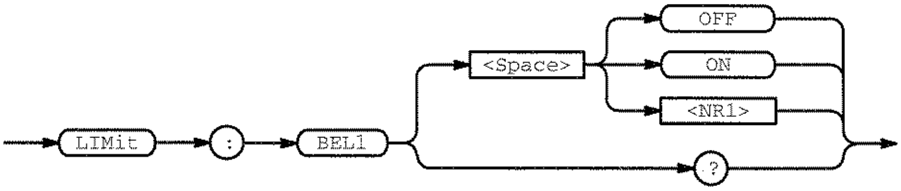
Group: Limit Test

Related Commands: LIMit:COMpare:CH<x>, LIMit:STATE

Syntax:

```
LIMit:BELl { OFF | ON | <NR1> }
```

```
LIMit:BELl?
```



Arguments: `OFF` or `<NR1>` = 0 turns off ringing the bell when any waveform data exceeds the limits set by the limit test.

`ON` or `<NR1>` ≠ 0 turns on ringing the bell.

Examples: `LIMit:BELl ON`

specifies that the bell is to ring when any waveform data exceeds the limits specified in the limit test.

`LIMit:BELl?`

returns either `0` or `1`, indicating whether the bell is to ring when any waveform data exceeds the limits specified in the limit test.

LIMit:COMpare:CH<x>

Sets or queries the template against which to compare the waveform acquired through the specified channel. The template can be a waveform saved in any of the reference locations REF1 through REF4, or none.

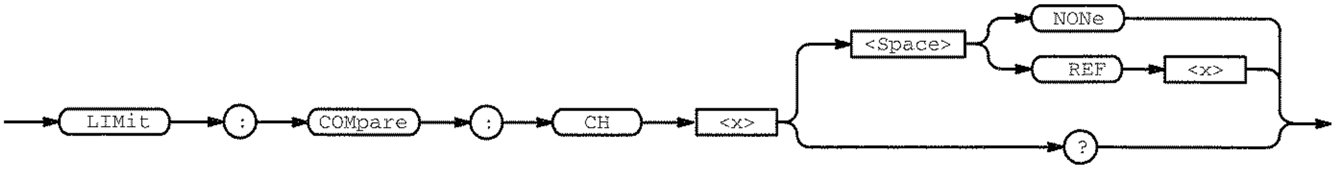
Group: Limit Test

Related Commands: CURve, LIMit:TEMPLate, LIMit:TEMPLate:DESTination, LIMit:TEMPLate:SOUrce, WFMPre

Syntax:

```
LIMit:COMpare:CH<x> { NONe | REF<x> }
```

```
LIMit:COMpare:CH<x>?
```



Arguments: `REF<x>` is a reference waveform.

`NONe` specifies that no template testing is to be done for the specified channel.

Examples: `LIMIT:COMPARE:CH1 REF1`
specifies REF1 as the template waveform against which to compare waveforms acquired using CH1.

`LIMIT:COMPARE:CH2?`
might return `LIMIT:COMPARE:CH2 REF4`, indicating that waveforms acquired using CH2 will be compared to the template waveform stored in REF4.

LIMit:HARDCopy

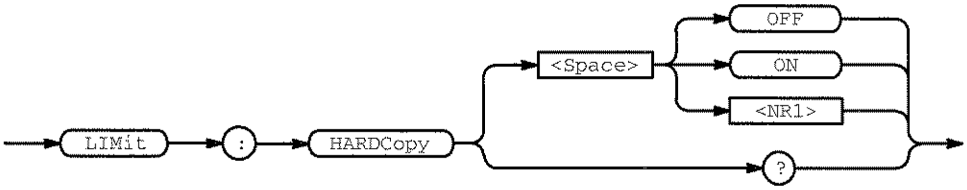
Executes a hardcopy operation on the waveform when any waveform data exceeds the limits set in the limit test, if the limit state is on. The hardcopy operation uses the port, and prints in the format and layout, specified using the HARDCopy commands.

Group: Limit Test

Related Commands: LIMit:COMpare:CH<x>, LIMit:STATE, HARDCopy

Syntax: `LIMit:HARDCopy { OFF | ON | <NR1> }`

`LIMit:HARDCopy?`



Arguments: `ON` or `<NR1>` ≠ 0 turns on the hardcopy operation for the waveform when any waveform data exceeds the limits set by the limit test.

`OFF` or `<NR1>` = 0 turns off the hardcopy operation.

Examples: `LIMit:HARDCopy ON`
specifies that the hardcopy operation occurs for the waveform when any waveform data exceeds the limits specified in the limit test.

`LIMit:HARDCopy?`
returns either `0` or `1`, indicating whether the hardcopy operation occurs for the waveform when any waveform data exceeds the limits specified in the limit test.

LIMit:STATE

Turns limit testing on or off, or queries whether limit testing is in effect.

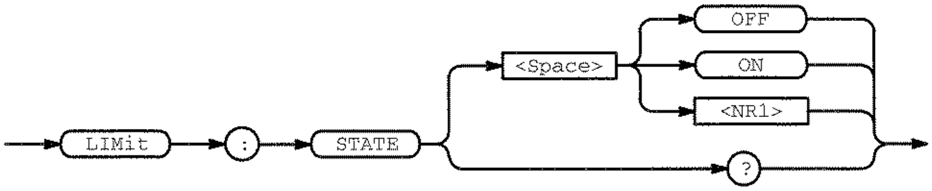
Group: Limit Test

Related Commands: CURve, LIMit:BELl, LIMit:COMpare:CH<x>, LIMit:HARDCopy, LIMit:TEMPLate, WFMPre

Syntax:

```
LIMit:STATE { OFF | ON | <NR1> }
```

```
LIMit:STATE?
```



Arguments: `OFF` or `<NR1>` = 0 turns off limit testing.

`ON` or `<NR1>` ≠ 0 turns on limit testing.

Examples: `LIMit:STATE ON`
specifies that limit testing of waveforms is in effect.

`LIMit:STATE?`
returns either `0` or `1`, indicating whether limit testing of waveforms is in effect.

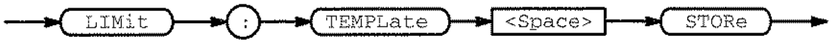
LIMit:TEMPLate (No Query Form)

Sets the template against which to compare the waveform acquired through the specified channel. The template can be a waveform saved in any of the reference locations REF1 through REF4, or none.

Group: Limit Test

Related Commands: LIMit:TEMPLate:DESTination, LIMit:TEMPLate:SOUrce, LIMit:TEMPLate:TOLerance

Syntax:
```
LIMit:TEMPLate STORe
```



Arguments: `STORe` creates a template with the specified source waveform and tolerances, and stores it in the destination reference waveform to be used in limit testing comparisons.

Examples: `LIMIT:TEMPLate STORe`
creates a template with the specified source waveform and tolerances, and stores it in the destination reference waveform to be used in limit testing comparisons.

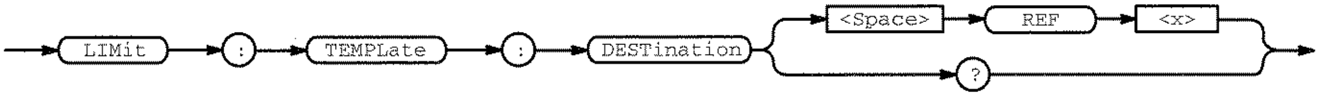
LIMit:TEMPLate:DESTination

Sets or queries the destination reference waveform in which to store the template waveform to use in limit tests. The LIMit:TEMPLate STORe command must be executed for this to take effect.

Group: Limit Test

Related Commands: LIMit:COMpare:CH<x>, LIMit:TEMPLate, LIMit:TEMPLate:SOUrce

Syntax:
```
LIMit:TEMPLate:DESTination REF<x>
LIMit:TEMPLate:DESTination?
```



Arguments: `REF<x>` specifies the reference waveform destination in which the template waveform is to be stored.

Examples: `LIMIT:TEMPLate:DESTination REF2`
specifies that the template waveform referred to with the LIMit:TEMPLate STORe command is stored as the REF2 waveform.

LIMit:TEMPLate:SOUrce

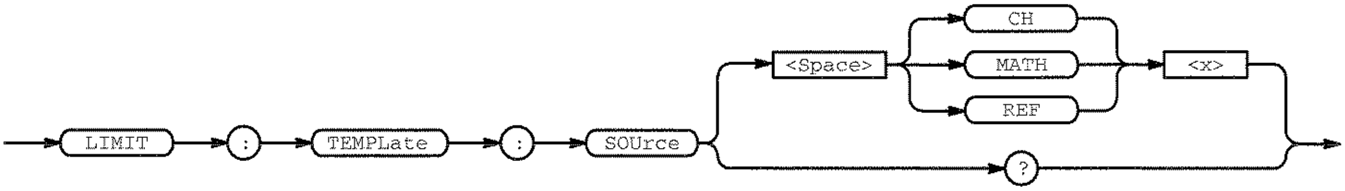
Sets or queries the channel, math waveform, or reference waveform to use as the source of the template waveform for limit tests. The LIMit:TEMPLate STORe command must be executed for this to take effect.

Group: Limit Test

Related Commands: LIMit:COMpare:CH<x>, LIMit:TEMPLate, LIMit:TEMPLate:DESTination

Syntax:

```
LIMit:TEMPLate:SOUrce { CH<x> | MATH<x> | REF<x> }
LIMit:TEMPLate:SOUrce?
```



Arguments: `CH<x>` specifies that the template waveform is the waveform currently being acquired using the specified channel.

`MATH<x>` specifies that the template waveform is the waveform currently stored as the specified math waveform.

`REF<x>` specifies that the template waveform is the waveform currently stored as the specified reference waveform.

Examples: `LIMIT:TEMPLate:SOUrce CH2`
specifies that the template waveform for limit tests is the waveform currently acquired using channel 2.

`LIMIT:TEMPLate:SOUrce?`
might return `MATH3`, specifying that the template waveform for limit tests is the waveform currently stored as the MATH3 waveform.

LIMit:TEMPLate:TOLerance:HORizontal

Sets or queries the amount by which the tested waveform can vary, in units of horizontal divisions, when comparing the current waveform to the template waveform for limit tests. The LIMit:TEMPLate STORe command must be executed for this to take effect.

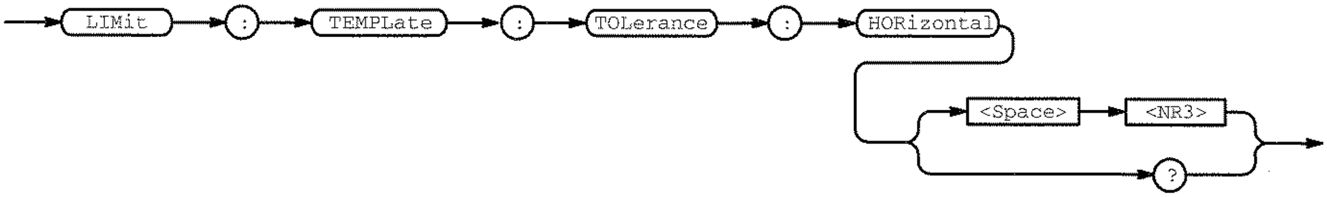
Group: Limit Test

Related Commands: LIMit:COMpare:CH<x>

Syntax:

```
LIMit:TEMPLate:TOLerance:HORizontal <NR3>
```

```
LIMit:TEMPLate:TOLerance:HORizontal?
```



Arguments: <NR3> is the amount, in horizontal divisions, by which the current waveform is allowed to deviate from the template waveform without being deemed to have exceeded the limits set in the limit test. The range is 0 to 5 divisions.

Examples:

```
LIMIT:TEMPLate:TOLerance:HORizontal 1.0
```

specifies that the current waveform is deemed to be close enough to the template waveform if it is within ±1.0 horizontal division.

```
LIMIT:TEMPLate:TOLerance:HORizontal?
```

might return `1.0`, specifying that the current waveform is deemed to be close enough to the template waveform if it is within ±1.0 horizontal division.

LIMit:TEMPLate:TOLerance:VERTical

Sets or queries the amount by which the tested waveform can vary, in units of vertical divisions, when comparing the current waveform to the template waveform for limit tests. The LIMit:TEMPLate STORe command must be executed for this to take effect.

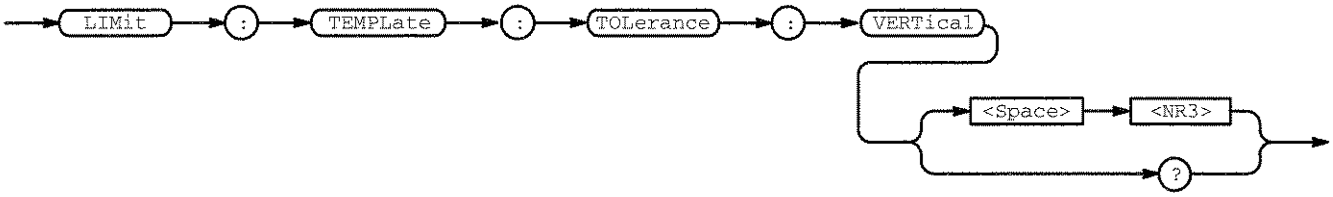
Group: Limit Test

Related Commands: LIMit:COMpare:CH<x>

Syntax:

```
LIMit:TEMPLate:TOLerance:VERTical <NR3>

LIMit:TEMPLate:TOLerance:VERTical?
```



Arguments: <NR3> is the amount, in vertical divisions, by which the current waveform is allowed to deviate from the template waveform without being deemed to have exceeded the limits set in the limit test. The range is 0 to 5 divisions.

Examples:

`LIMIT:TEMPLate:TOLerance:VERTical 1.0`

specifies that the current waveform is deemed to be close enough to the template waveform if it is within ±1.0 vertical division from the template waveform.

`LIMIT:TEMPLate:TOLerance:VERTical?`

might return `1.0`, specifying that the current waveform is deemed to be close enough to the template waveform if it is within ±1.0 vertical division.

LOCk

Enables and disables all front panel buttons and knobs. There is no front-panel equivalent.

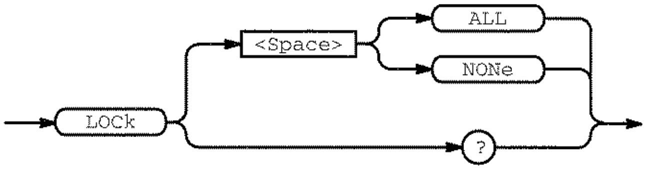
Group: Miscellaneous

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 digitizing 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.

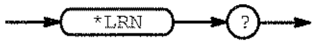
*LRN? (Query Only)

Returns a string listing the digitizing oscilloscope's settings, except for configuration information for the calibration values. You can use this string to return the digitizing oscilloscope to the state it was in when you made the *LRN? query.

Group: Miscellaneous

Related Commands: HEADer, SET?, VERBose

Syntax: `*LRN?`



NOTE

*The *LRN? query 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 digitizing oscilloscope as a command string. The VERBose command can still be used normally to specify whether the returned headers should be abbreviated.*

Examples: `*LRN?`

a partial response might look like this:

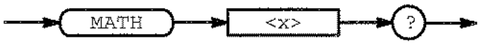
```
:ACQUIRE:STATE 1;MODE SAMPLE;NUMENV 10;NUMAVG 16;
REPET 1;STOPAFTER RUNSTOP;:DIAG:LOOP:OPTION ONCE;
COUNT 1;:DIAG:STATE HALT;:HEADER 1;:VERBOSE 1;
:CURSOR:FUNCTION OFF;VBARS:UNITS SECONDS;
POSITION1 1.00E-6;POSITION2 9.00E-6;SELECT CURSOR1;
```

MATH<x>? (Query Only)

Returns the definition for the math waveform specified by <x>.

Group: Vertical

Syntax: `MATH<x>?`



MATH<x>:DEFINE

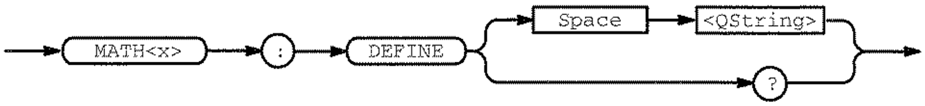
Allows the user to define new waveforms using mathematical expressions. This is equivalent to selecting **Change Math waveform definition** in the Math<x> side menu.

Group: Vertical

Syntax:

```
MATH<x>:DEFINE <QString>

MATH<x>:DEFINE?
```



Arguments: `<QString>` contains the mathematical expression. The expression can include any amount of white space. Expressions can be either single or dual waveform expressions. <src> and <function> elements are case independent.

The format for a single waveform expression is:

```
<function>(<source> [, <window>, <scaling>, <phase
suppression>])
```

The format for a dual waveform expression is:

```
<source><operator><source>
```

where:

`<function> ::= INV | DIF | FFT | INT`

- `INV` (for invert): inverts the defined waveform.
- `DIFferentiate` (Option 2F only): takes the derivative of the selected waveform.
- `FFT` (Option 2F only): provides an FFT of the selected waveform. It uses the format: “FFT(<source>, <window>, <scaling>, <phase suppression>)” where the window, scaling, and phase suppression arguments in the parentheses are optional. You can specify these arguments in any order.

  <source> refers to a signal channel. Valid choices are: `CH1, CH2, CH3, CH4, REF1, REF2, REF3, or REF4.`
  <window> refers to an FFT window. Valid choices are: `RECtangular`, `HAMming`, `HANning`, or `BLAckmanharris`.
  <scaling> provides vertical scaling. Valid choices are: `LOGrms`, `LINearrms`, `DEGreesphase`, or `RADiansphase`.
  <Phase suppression> is of the range: –100 dB to 100 dB.

- `INTegrate` (Option 2F only): takes the integral of the selected waveform.

```
<operator> ::= { + | - | * }
```

```
<source> ::= { CH<x> | REF<x> }
```

Examples: `MATH2:DEFINE "Ch1 + cH2"`
adds channel 1 and channel 2, and stores the result in MATH2.

`MATH1:DEFINE "INV( ref4 )"`
inverts the waveform stored in reference memory location 4 storing the result in MATH1.

`MATH1:DEFINE "FFT( CH1 )"`
takes an FFT on the waveform from channel 1 and stores the result in MATH1.

`MATH1:DEFINE "FFT( CH1, HAMM, LINEARRMS, 20 )"`
takes an FFT from channel1, using the HAMMING algorithm, with linear rms scaling, and 20 dB phase suppression. The result is stored in MATH1.

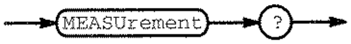
`MATH1:DEFINE?`
might return `"Ch2*Ref2"` as the expression that defines MATH1.

MEASUrement? (Query Only)

Returns all measurement parameters.

Group: Measurement

Syntax: `MEASUrement?`



Examples: MEASUREMENT?

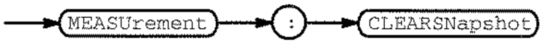
might return :MEASUREMENT:MEAS1:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH1;DELAY:EDGE1 RISE;EDGE2 RISE;DIRECTION FORWARDS;:MEASUREMENT:MEAS2:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH1;DELAY:EDGE1 RISE;EDGE2 RISE;DIRECTION FORWARDS;:MEASUREMENT:MEAS3:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH1;DELAY:EDGE1 RISE;EDGE2 RISE;DIRECTION FORWARDS;:MEASUREMENT:MEAS4:STATE 0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH1;DELAY:EDGE1 RISE;EDGE2 RISE;DIRECTION FORWARDS;:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH1;DELAY:EDGE1 RISE;EDGE2 RISE;DIRECTION FORWARDS;:MEASUREMENT:METHOD HISTOGRAM;REFLEVEL:METHOD PERCENT;ABSOLUTE:HIGH 0.0E+0;LOW 0.0E+0;MID 0.0E+0;MID2 0.0E+0;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0E+0;LOW 10.0E+0;MID 50.0E+0;MID2 50.0E+0

MEASUrement:CLEARSNapshot

Takes down the measurement snapshot display.

Group: Measurement

Syntax: MEASUrement:CLEARSNapshot



Examples: MEASUREMENT:CLEARSNAPSHOT

MEASUrement:GATing

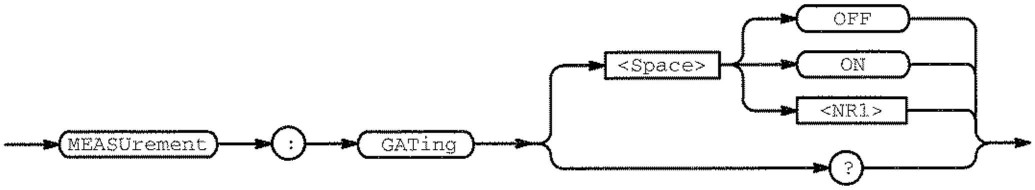
Sets or queries measurement gating.

Group: Measurement

Related Commands: CURSor:VBARS

Syntax: MEASUrement:GATing { ON | OFF | <NR1> }

MEASUrement:GATing?



Arguments: ON (or 1) turns on measurement gating.

OFF (or 0) turns off measurement gating.

Examples: MEASUREMENT:GATING ON

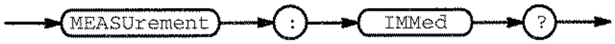
MEASUREMENT:GATING?
might return MEASUREMENT:GATING 1
showing gating is turned on.
It might also return MEASUREMENT:GATING 0
showing gating is turned off.

MEASUrement:IMMed? (Query Only)

Returns all immediate measurement setup parameters.

Group: Measurement

Syntax: MEASUrement:IMMed?



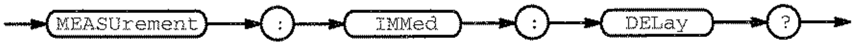
Examples: MEASUREMENT:IMMED?
might return :MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s"; SOURCE1 CH1;SOURCE2 CH1;DELAY:EDGE1 RISE;EDGE2 RISE; DIRECTION FORWARDS

MEASUrement:IMMed:DELay? (Query Only)

Returns information about the immediate delay measurement.

Group: Measurement

Syntax: MEASUrement:IMMed:DELay?



Examples: `MEASUREMENT:IMMED:DELAY?`
might return `:MEASUREMENT:IMMED:DELAY:EDGE1 RISE;EDGE2 RISE; DIRECTION FORWARDS`

MEASUrement:IMMed:DELay:DIRection

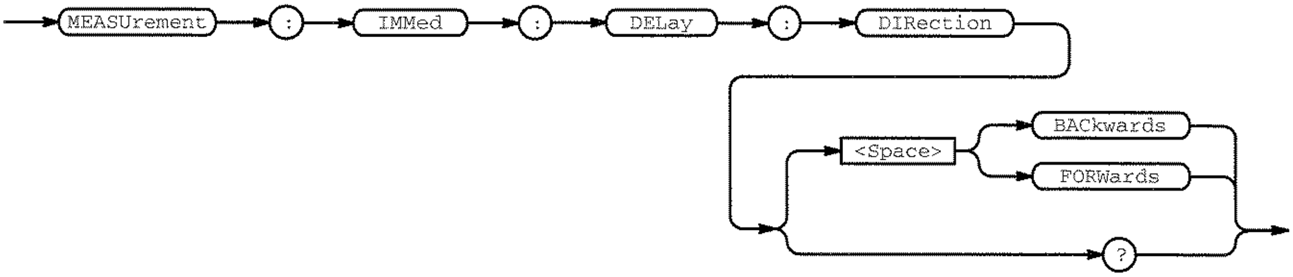
Sets or queries the starting point and direction that determines the delay "to" edge when taking an immediate delay measurement. Use the MEASUrement:IMMed:SOURCE2 command to specify the delay "to" waveform.

Group: Measurement

Syntax:

```
MEASUrement:IMMed:DELay:DIRection {BACkwards | FORWards}
```

```
MEASUrement:IMMed:DELay:DIRection?
```



Arguments: `BACkwards` means that the search starts at the end of the waveform and looks for the last rising or falling edge in the waveform. The slope of the edge is specified by MEASUrement:IMMed:DELay:EDGE2.

`FORWards` means that the search starts at the beginning of the waveform and looks for the first rising or falling edge in the waveform. The slope of the edge is specified by MEASUrement:IMMed:DELay:EDGE2.

Examples: `MEASUREMENT:IMMED:DELAY:DIRECTION FORWARDS`
starts searching from the beginning of the waveform record.

`MEASUREMENT:IMMED:DELAY:DIRECTION?`
returns either `BACkwards` or `FORWARDS`.

MEASUrement:IMMed:DELay:EDGE1

Sets or queries the slope of the edge that is used for the delay "from" waveform when taking an immediate delay measurement. The waveform is specified by MEASUrement:IMMed:SOURCE1.

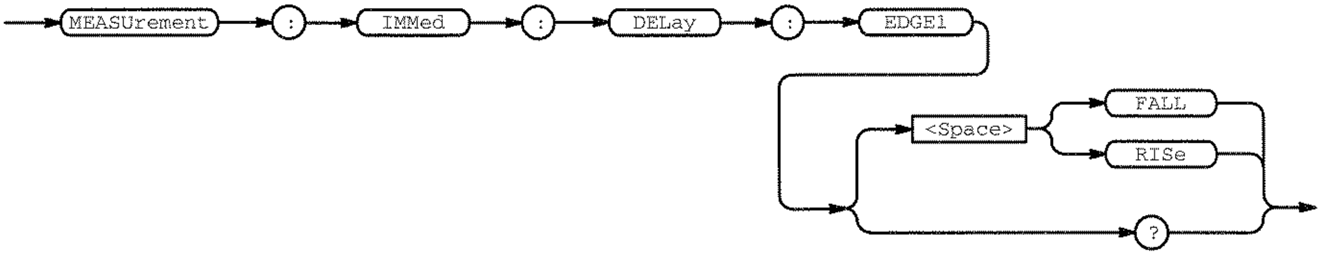
Group: Measurement

Related Commands: MEASUrement:IMMed:SOURCE1

Syntax:

```
MEASUrement:IMMed:DELay:EDGE1 { FALL | RISe }

MEASUrement:IMMed:DELay:EDGE1?
```



Arguments: FALL specifies the falling edge.

RISe specifies the rising edge.

Examples:

MEASUREMENT:IMMED:DELAY:EDGE1 RISE
specifies that the rising edge be used for the immediate delay measurement.

MEASUREMENT:IMMED:DELAY:EDGE1?
returns either RISE or FALL.

MEASUrement:IMMed:DELay:EDGE2

Sets or queries the slope of the edge that is used for the delay "to" waveform when taking an immediate delay measurement. The waveform is specified by MEASUrement:IMMed:SOURCE2.

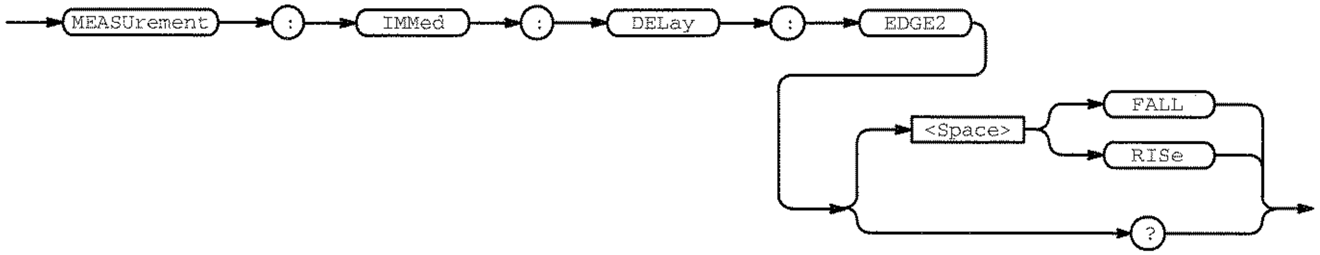
Group: Measurement

Related Commands: MEASUrement:IMMed:SOURCE2

Syntax:

```
MEASUrement:IMMed:DELay:EDGE2 { FALL | RISe }

MEASUrement:IMMed:DELay:EDGE2?
```



Arguments: FALL specifies the falling edge.

RISe specifies the rising edge.

Examples: MEASUREMENT:IMMED:DELAY:EDGE2 RISE
specifies that the rising edge be used for the immediate delay measurement.

MEASUREMENT:IMMED:DELAY:EDGE2?
returns FALL showing that the falling or negative edge of the waveform is used for the immediate delay measurement.

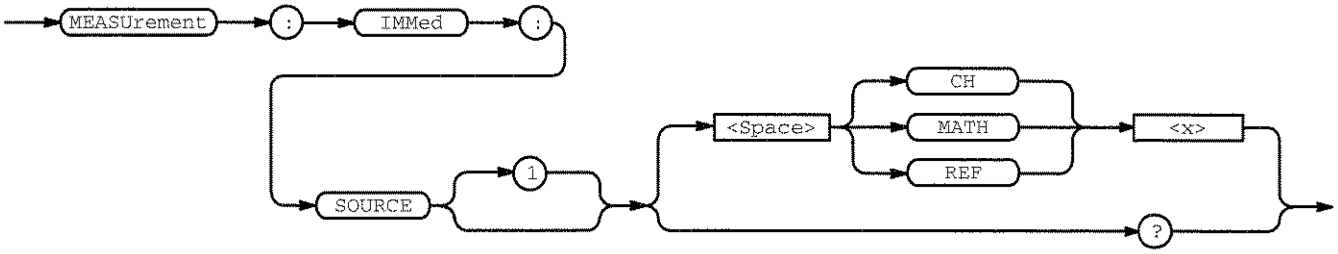
MEASUrement:IMMed:SOURCE[1]

Sets or queries the source for all single channel immediate measurements and specifies the source to measure "from" when taking an immediate delay measurement or phase measurement.

Group: Measurement

Syntax: MEASUrement:IMMed:SOURCE[1] { CH<x> | MATH<x> | REF<x> }

MEASUrement:IMMed:SOURCE[1]?



Arguments: CH<x> is an input channel.

MATH<x> is a math waveform.

REF<x> is a reference waveform.

Examples: MEASUREMENT:IMMED:SOURCE MATH1
specifies MATH1 as the immediate measurement source.

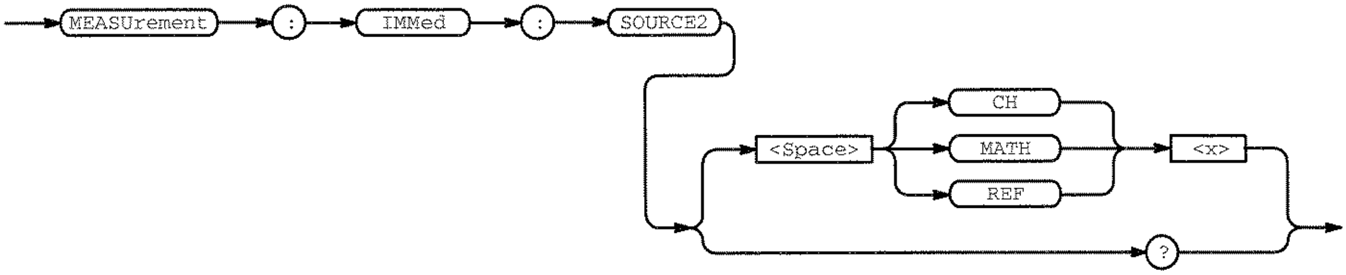
MEASUrement:IMMed:SOURCE2

Specifies the source to measure “to” when taking an immediate delay measurement or phase measurement.

Group: Measurement

Syntax:

```
MEASUrement:IMMed:SOURCE2 { CH<x> | MATH<x> | REF<x> }
MEASUrement:IMMed:SOURCE2?
```



Arguments: `CH<x>` is an input channel.

`MATH<x>` is a math waveform.

`REF<x>` is a reference waveform.

Examples: `MEASUREMENT:IMMED:SOURCE2 REF3`
sets the waveform in reference memory location 3 as the delay “to” source when making delay measurements.

`MEASUREMENT:IMMED:SOURCE2?`
might return `MATH1`.

MEASUrement:IMMed:TYPe

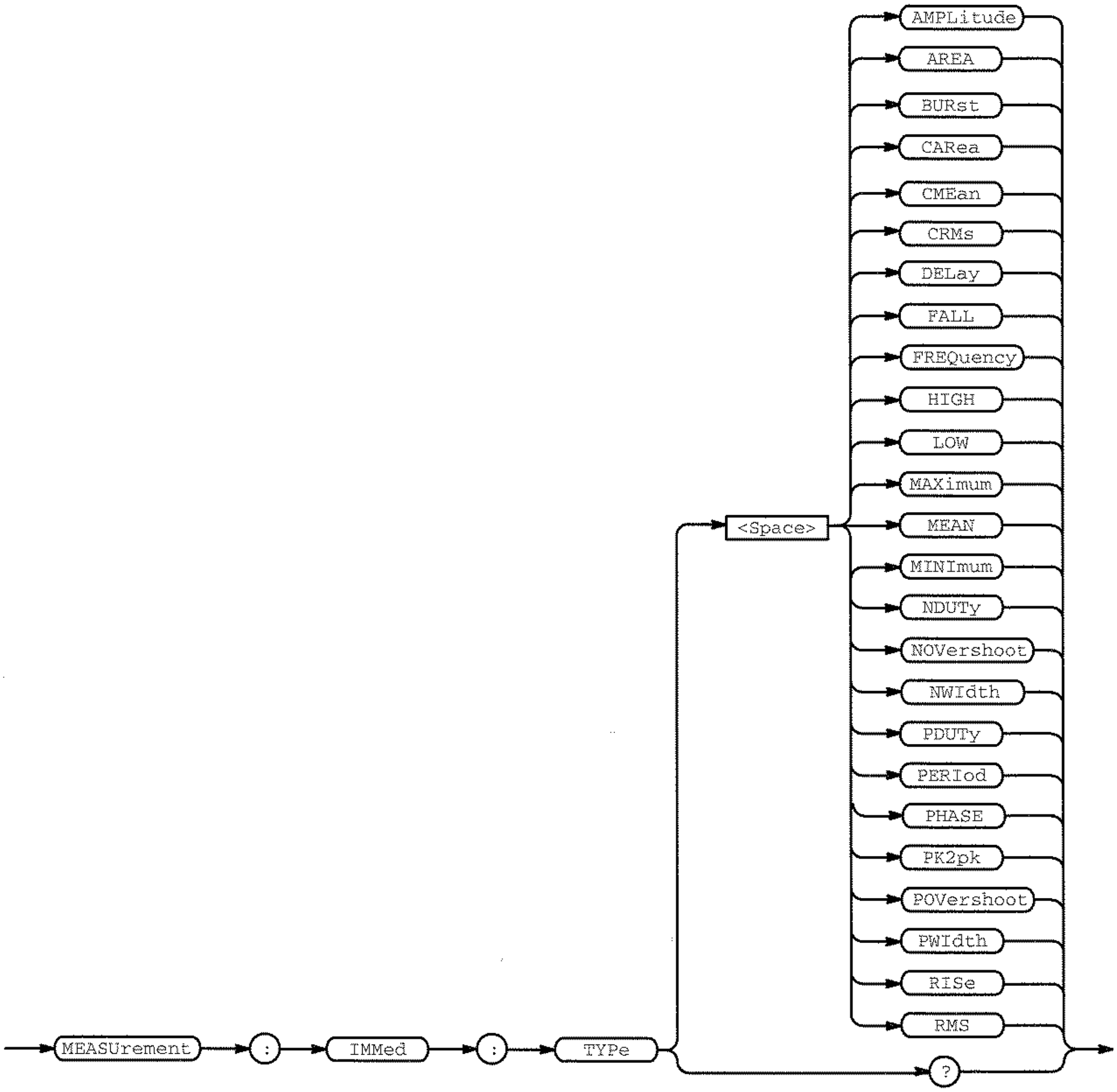
Specifies the immediate measurement.

Group: Measurement

Syntax:

```
MEASUrement:IMMed:TYPe { AMPLitude | AREA | BURst | CARea
    | CMEan | CRMs | DELay | FALL | FREQuency | HIGH | LOW
    | MAXimum | MEAN | MINImum | NDUTy | NOVershoot |
   NWIdth | PDUTy | PERIod | PHASE | PK2pk | POVershoot |
   PWIdth | RISe | RMS }
```

```
MEASUrement:IMMed:TYPe?
```



Arguments: `AMPLitude` is the high value minus the low value.

`AREA` is the area between the curve and ground over the entire waveform.

`BURst` is the time from the first MidRef crossing to the last MidRef crossing.

`CARea` (cycle area) is the area between the curve and ground over one cycle.

`CMEan` is the arithmetic mean over one cycle.

`CRMs` is the true Root Mean Square voltage over one cycle.

`DELay` is the time between the MidRef crossings of two different waveforms.

`FALL` is the time that it takes for the falling edge of a pulse to fall from a HighRef value to a LowRef value of its final 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:

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

`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.

`PHASE` is the phase difference from the selected waveform to the designated waveform.

`PK2pk` is the absolute difference between the maximum and minimum amplitude.

`POVershoot` is the positive overshoot, expressed as:

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

`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 of its final value.

`RMS` is the true Root Mean Square voltage.

Examples: `MEASUREMENT:IMMED:TYPE FREQUENCY`
defines the immediate measurement to be a frequency measurement.

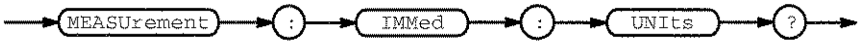
MEASUrement:IMMed:UNIts? (Query Only)

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, `"VV"` for volts^2,or `"%"` for percent.

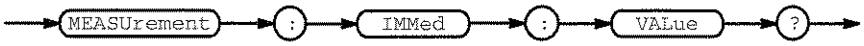
Examples: `MEASUREMENT:IMMED:UNITS?`
might return `"s"`, indicating that the units for the immediate measurement are seconds.

MEASUrement:IMMed:VALue? (Query Only)

Immediately executes the immediate measurement specified by the MEASUrement:IMMed:TYPe command. The measurement is taken on the source(s) specified by the SELect:CH<x> command.

Group: Measurement

Syntax: `MEASUrement:IMMed:VALue?`



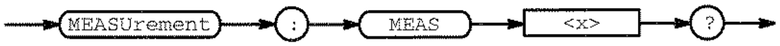
Returns: `<NR3>`

MEASUrement:MEAS<x>? (Query Only)

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

Group: Measurement

Syntax: `MEASUrement:MEAS<x>?`



Examples:

```
MEASUREMENT:MEAS3?
```

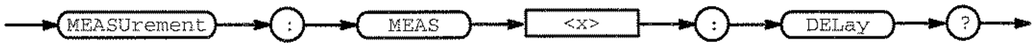
might return `:MEASUREMENT:MEAS3:STATE 0;TYPE PERIOD; UNITS "s";SOURCE1 CH1;SOURCE2 CH2;DELAY:EDGE1 RISE; EDGE2 RISE;DIRECTION FORWARDS.`

MEASUrement:MEAS<x>:DELay? (Query Only)

Returns the delay measurement parameters for the measurement specified by <x>.

Group: Measurement

Syntax: `MEASUrement:MEAS<x>:DELay?`



Examples:

```
MEASUREMENT:MEAS3:DELAY?
```

might return `:MEASUREMENT:MEAS3:DELAY:EDGE1 RISE; EDGE2 RISE;DIRECTION FORWARDS.`

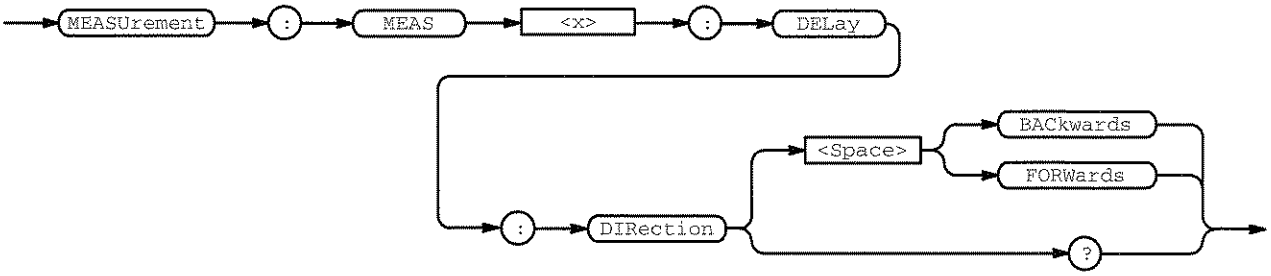
MEASUrement:MEAS<x>:DELay:DIRection

Sets or queries the starting point and direction that determines the delay "to" edge when taking a delay measurement. The waveform is specified by MEASUrement:MEAS<X>:SOURCE2. This command is equivalent to setting the direction in the Delay Edges & Direction side menu.

Group: Measurement

Syntax: `MEASUrement:MEAS<x>:DELay:DIRection { BACkwards | FORWards }`

`MEASUrement:MEAS<x>:DELay:DIRection?`



Arguments: `BACkwards` means that the search starts at the end of the waveform and looks for the last rising or falling edge in the waveform. The slope of the edge is specified by MEASUrement:MEAS<x>:DELay:EDGE2.

`FORWards` means that the search starts at the beginning of the waveform and looks for the first rising or falling edge in the waveform. The slope of the edge is specified by MEASUrement:MEAS<x>:DELay:EDGE2.

Examples: `MEASUREMENT:MEAS1:DELAY:DIRECTION BACKWARDS`
starts searching from the end of the waveform record.

`MEASUREMENT:MEAS3:DELAY:DIRECTION?`
might return `FORWARDS` for the search direction.

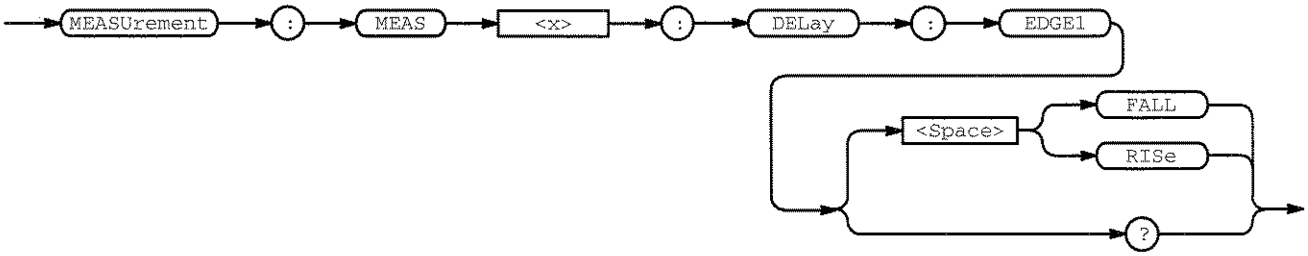
MEASUrement:MEAS<x>:DELay:EDGE1

Sets or queries the slope of the edge that is used for the delay "from" waveform when taking a delay measurement. The waveform is specified by MEASUrement:MEAS<x>:SOURCE1. This command is equivalent to selecting the edges in the Delay Edges & Direction side menu.

Group: Measurement

Syntax: `MEASUrement:MEAS<x>:DELay:EDGE1 { FALL | RISe }`

`MEASUrement:MEAS<x>:DELay:EDGE1?`



Arguments: `FALL` specifies the falling edge.

`RISe` specifies the rising edge.

Examples: `MEASUREMENT:MEAS3:DELAY:EDGE1 RISE`
specifies that the rising edge be used for measurement 3.

`MEASUREMENT:MEAS1:DELAY:EDGE1?`
returns either `RISE` or `FALL` for measurement 1.

MEASUrement:MEAS<x>:DELay:EDGE2

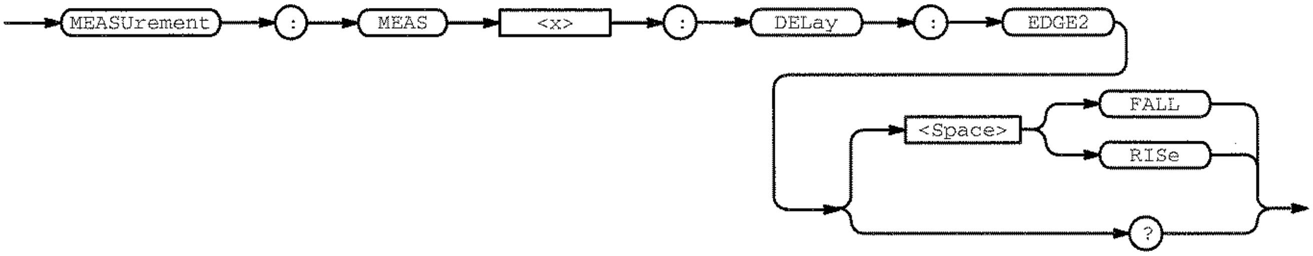
Sets or queries the slope of the edge that is used for the delay “to” waveform when taking a delay measurement. The waveform is specified by MEASUrement:MEAS<x>:SOURCE2. This command is equivalent to selecting the edges in the Delay Edges & Direction side menu.

Group: Measurement

Syntax:

```
MEASUrement:MEAS<x>:DELay:EDGE2 { FALL | RISe }
```

```
MEASUrement:MEAS<x>:DELay:EDGE2?
```



Arguments: `FALL` specifies the falling edge.

`RISe` specifies the rising edge.

Examples: `MEASUREMENT:MEAS2:DELAY:EDGE2 RISE`
specifies that the rising edge be used for the second delay measurement.

`MEASUREMENT:MEAS2:DELAY:EDGE2?`
might return `FALL` showing that the falling or negative edge of the waveform is used for the second measurement.

MEASUrement:MEAS<x>:SOURCE[1]

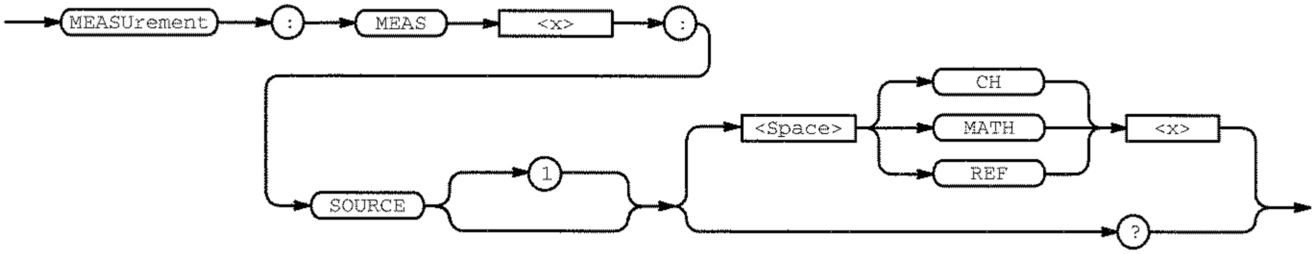
Sets or queries the source for all single channel measurements and specifies the source to measure "from" when taking a delay measurement or phase measurement.

Group: Measurement

Syntax:

```
MEASUrement:MEAS<x>:SOURCE[1] { CH<x> | MATH<x> | REF<x> }

MEASUrement:MEAS<x>:SOURCE[1]?
```



Arguments: CH<x> is an input channel.

MATH<x> is a math waveform.

REF<x> is a reference waveform.

Examples: `MEASUREMENT:MEAS2:SOURCE1 MATH1`

specifies MATH1 as the measurement 2 source.

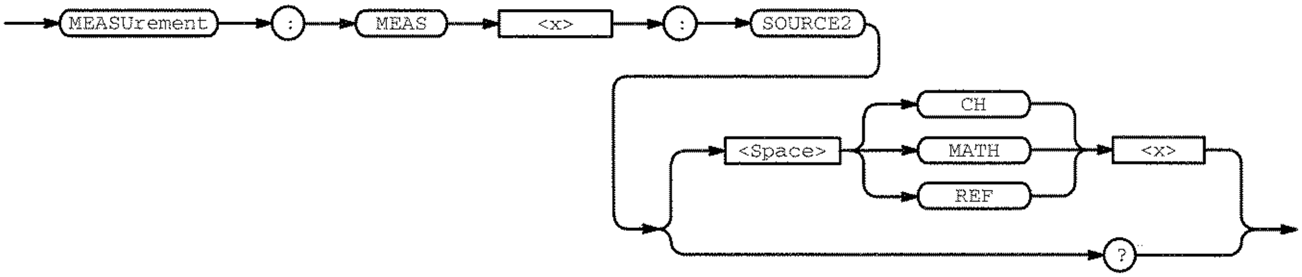
MEASUrement:MEAS<x>:SOURCE2

Sets or queries the source to measure “to” when taking a delay measurement or phase measurement. This is equivalent to setting the source in the Delay from Selected Wfm side menu or the Phase from Selected Wfm side menu.

Group: Measurement

Syntax: `MEASUrement:MEAS<x>:SOURCE2 { CH<x> | MATH<x> | REF<x> }`

`MEASUrement:MEAS<x>:SOURCE2?`



Arguments: `CH<x>` is an input channel.

`MATH<x>` is a math waveform.

`REF<x>` is a reference waveform.

Examples: `MEASUREMENT:MEAS4:SOURCE2 CH<x>`

sets channel 1 as the delay “to” source when making delay measurements.

`MEASUREMENT:MEAS2:SOURCE2?`

might return `MATH1`.

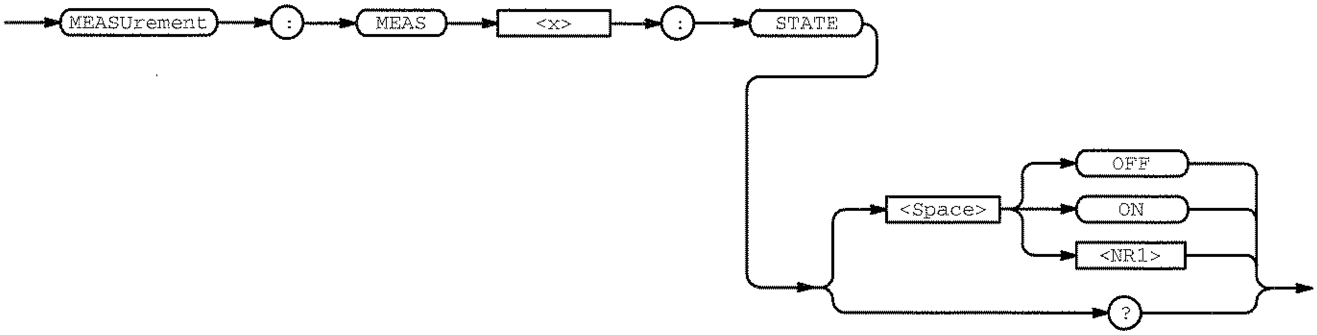
MEASUrement:MEAS<x>:STATE

Controls the measurement system. The source specified by MEASUrement:MEAS<x>:SOURCE1 must be selected for the measurement to be displayed. The source can be 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 deselecting 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.

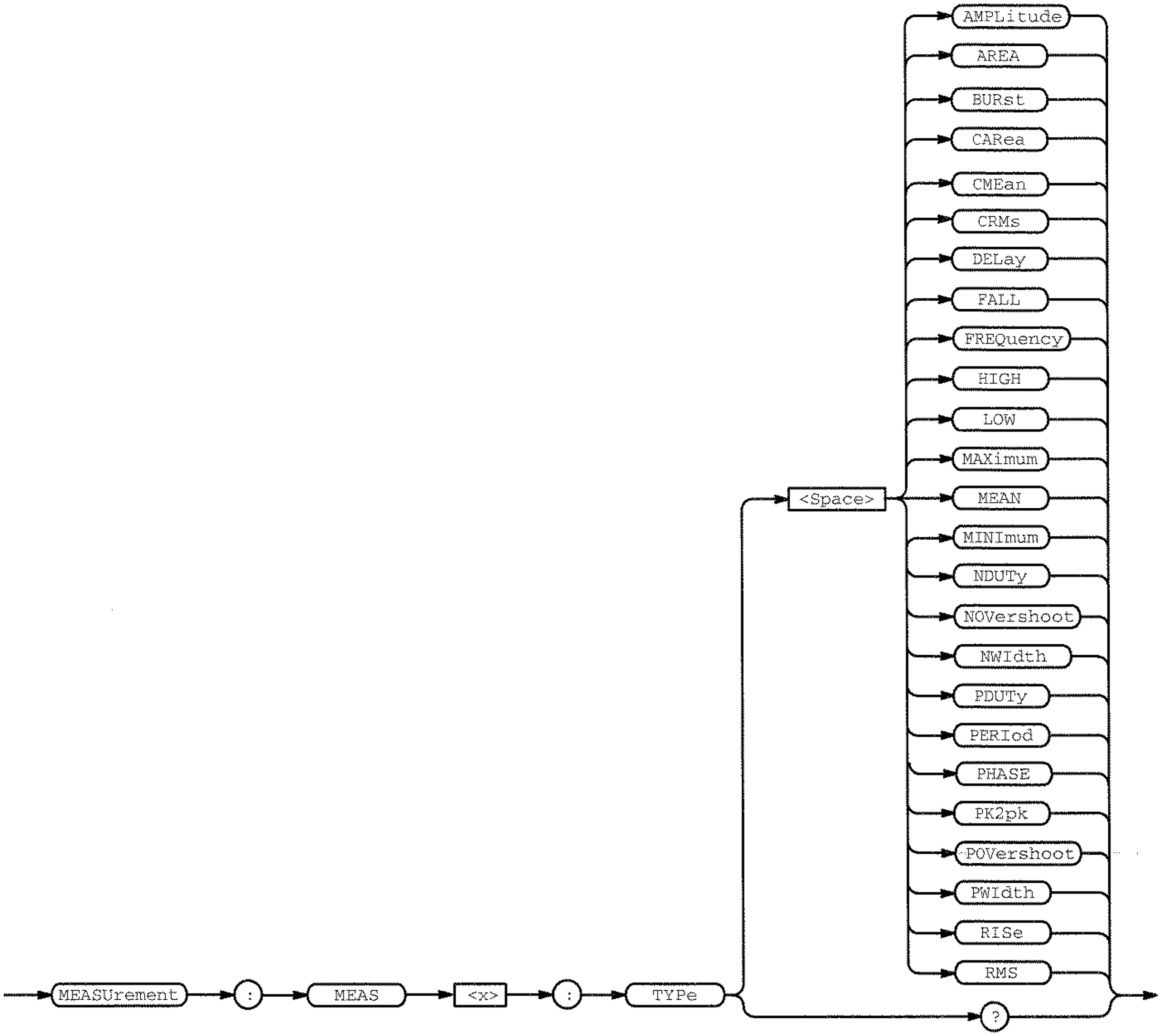
MEASUrement:MEAS<x>:TYPe

Sets or queries the measurement type for the measurement specified by MEAS<x>. This is equivalent to selecting the measurement in the Select Measurement side menu.

Group: Measurement

Syntax:
```
MEASUrement:MEAS<x>:TYPe { AMPLitude | AREA | BURst |
    CARea | CMEan | CRMs | DELay | FALL | FREQuency | HIGH
    | LOW | MAXimum | MEAN | MINImum | NDUTy | NOVershoot
    | NWIdth | PDUTy | PERIod |PHASE | PK2pk | POVershoot
    | PWIdth | RISe | RMS }
```

`MEASUrement:MEAS<x>:TYPe?`



Arguments: AMPLitude is the high value minus the low value or HIGH – LOW.

AREA is the area between the curve and ground over the entire waveform.

BURst is the time from the first MidRef crossing to the last MidRef crossing.

CARea (cycle area) is the area between the curve and ground over one cycle.

CMEan is the arithmetic mean over one cycle.

CRMs is the true Root Mean Square voltage over one cycle.

DELay is the time between the MidRef crossings of two different waveforms.

FALL is the time that it takes for the falling edge of a pulse to fall from a HighRef value to a LowRef value of its final 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:

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

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.

PHASE is the phase difference from the selected waveform to the designated waveform.

PK2pk is the absolute difference between the maximum and minimum amplitude.

POVershoot is the positive overshoot, expressed as:

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

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 of its final value.

RMS is the true Root Mean Square voltage.

Examples:

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

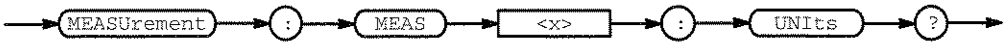
specifies MEAS3 to calculate the Root Mean Square voltage.

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, `"VV"` for volts^2, or `"%"` for percent.

Examples: `MEASUREMENT:MEAS3:UNITS?`
might return `"%"`, indicating the units for Measurement 3 are percent.

MEASUrement:MEAS<x>:VALue? (Query Only)

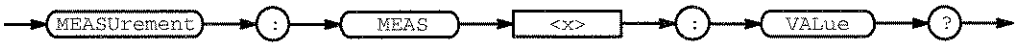
Returns the value that has been calculated for the measurement specified by <x>.

NOTE

This value is a display value and will be updated every 1/3 second.

Group: Measurement

Syntax: `MEASUrement:MEAS<x>:VALue?`



Returns: `<NR3>`

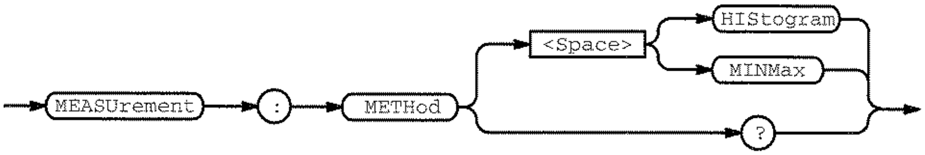
MEASUrement:METHod

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.

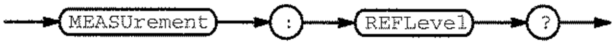
`MEASUREMENT:METHOD?`
returns `MINMAX` when the reference levels are set to MIN and MAX.

MEASUrement:REFLevel? (Query Only)

Returns the reference levels.

Group: Measurement

Syntax: `MEASUrement:REFLevel?`



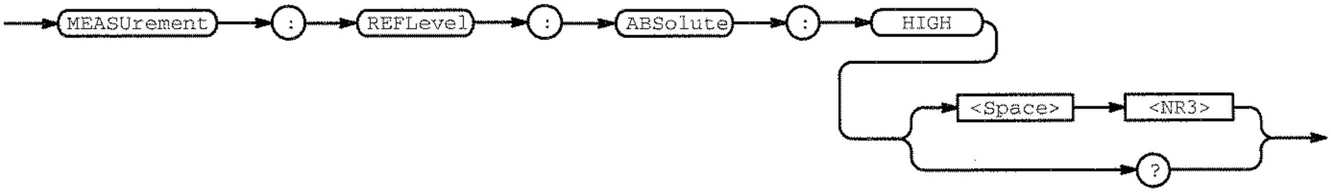
MEASUrement:REFLevel:ABSolute:HIGH

Sets or queries the high reference level, and is the 100% reference level when MEASUrement:REFLevel:METHod is set to ABSolute. This command is equivalent to setting the **Reference Levels** in the Measure menu.

Group: Measurement

Syntax:

```
MEASUrement:REFLevel:ABSolute:HIGH <NR3>
MEASUrement:REFLevel:ABSolute:HIGH?
```



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

Examples: `MEASUREMENT:REFLEVEL:ABSOLUTE:HIGH 1.71`
sets the high reference level to 1.71 V.

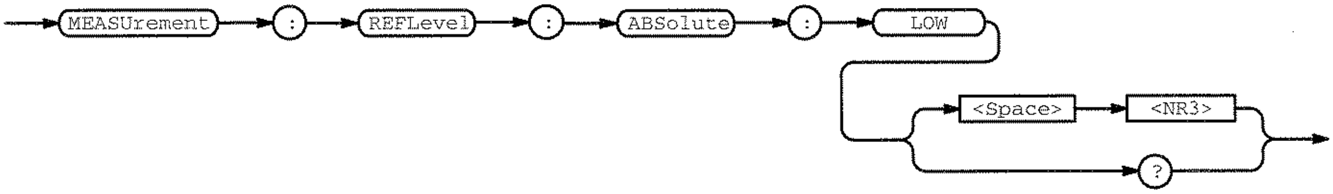
MEASUrement:REFLevel:ABSolute:LOW

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

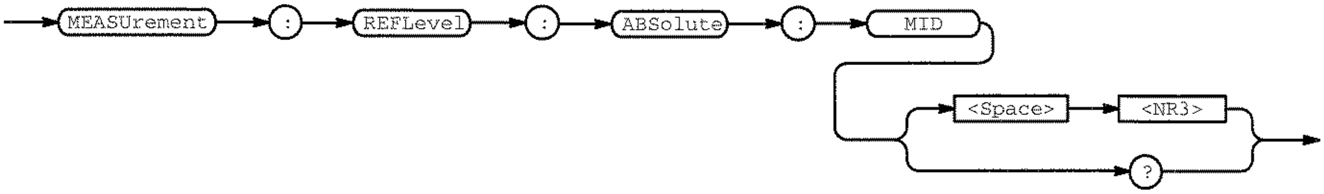
Sets or queries the mid reference level, and is the 50% reference level when MEASUrement:REFLevel:METHod is set to ABSolute. This command is equivalent to setting the **Reference Levels** in the Measure menu.

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 .71`
sets the mid reference level to .71 volts.

MEASUrement:REFLevel:ABSolute:MID2

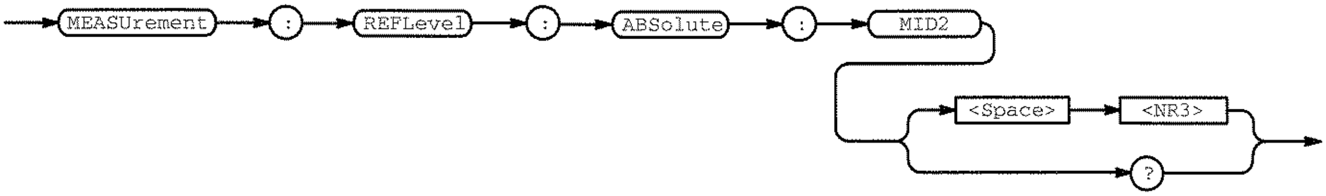
Sets or queries the mid reference level for the "to" waveform when taking a delay measurement, and is the 50% reference level when MEASUrement:REFLevel:METHod is set to ABSolute. This command is equivalent to setting the **Reference Levels** in the Measure menu.

Group: Measurement

Syntax:

```
MEASUrement:REFLevel:ABSolute:MID2 <NR3>
```

```
MEASUrement:REFLevel:ABSolute:MID2?
```



Arguments: `<NR3>` is the mid reference level, in volts. The default is 0.0 V.

Examples: `MEASUREMENT:REFLEVEL:ABSOLUTE:MID2 0.5`
sets the mid reference level for the delay waveform to 0.5 volts.

MEASUrement:REFLevel:METHod

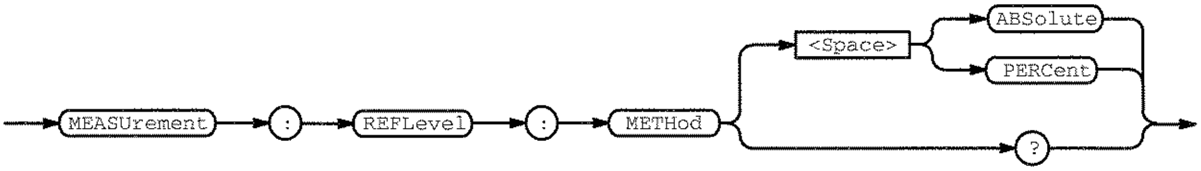
Specifies which reference levels are used for measurement calculations. This command is equivalent to setting the levels in the Reference Levels side menu.

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. For instance, when designing to published interface specifications such as RS-232-C.

`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.

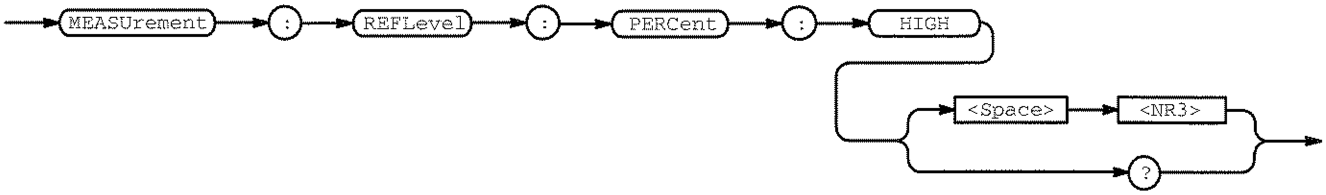
MEASUrement:REFLevel:PERCent:HIGH

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 percent, 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.

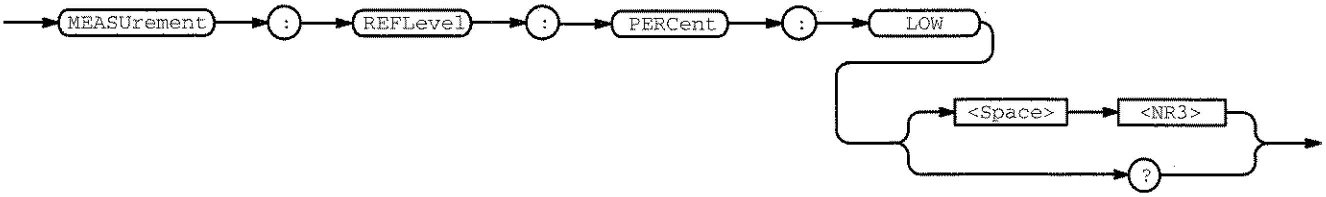
MEASUrement:REFLevel:PERCent:LOW

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 percent, 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.

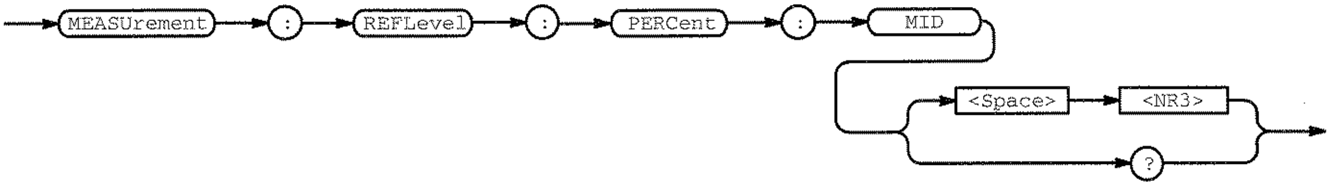
MEASUrement:REFLevel:PERCent:MID

Sets or queries the percent, relative to HIGH, 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 percent, 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 HIGH.

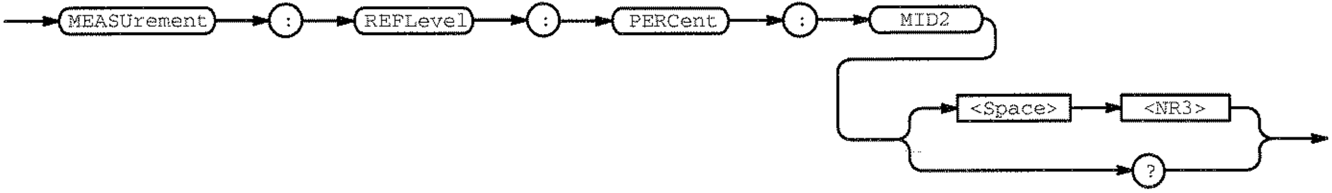
MEASUrement:REFLevel:PERCent:MID2

Sets or queries the percent, relative to HIGH, that is used to calculate the mid reference level for the second waveform specified when taking a delay measurement. This command is equivalent to setting the **Reference Levels** in the Measure menu.

Group: Measurement

Syntax:

```
MEASUrement:REFLevel:PERCent:MID2 <NR3>
MEASUrement:REFLevel:PERCent:MID2?
```



Arguments: <NR3> ranges from 0 to 100 percent, and is the mid reference level. The default is 50%.

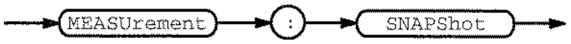
Examples: MEASUREMENT:REFLEVEL:PERCENT:MID2 40
specifies that the mid reference level is set to 40% of HIGH.

MEASUrement:SNAPShot

Displays the measurement snapshot.

Group: Measurement

Syntax: MEASUrement:SNAPShot



Examples: MEASUREMENT:SNAPSHOT

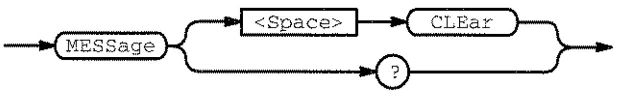
MESSage

Clears the message window and the MESSage? query returns the current message parameters.

Group: Display

Syntax: MESSage CLEar

MESSage?



Arguments: CLEar removes the message from the message window. This is equivalent to sending MESSage SHOW "".

Examples: MESSAGE CLEAR
clears the message from the window.

MESSage:BOX

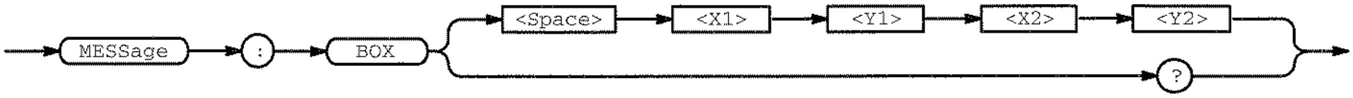
Defines the size and position of the message window. This command does not display the window unless MESSage:STATE is ON.

Group: Display

Syntax:

```
MESSage:BOX <X1>,<Y1>,<X2>,<Y2>

MESSage:BOX?
```



Arguments: <X1> and <X2> = 0 to 640, and are pixel positions along the horizontal axis. <X1> defines the left and <X2> defines the right side of the window.

<Y1> and <Y2> = 0 to 480, and are pixel positions along the vertical axis. <Y1> defines the top and <Y2> defines the bottom of the window. The reserved height of all characters is 15 pixels so the window must be at least that high to fully display characters. For a complete list of character widths in pixels, see Table A-1 on page A-1. Shorter windows clip characters.

Figure 3-1 shows the coordinate system relative to the screen.

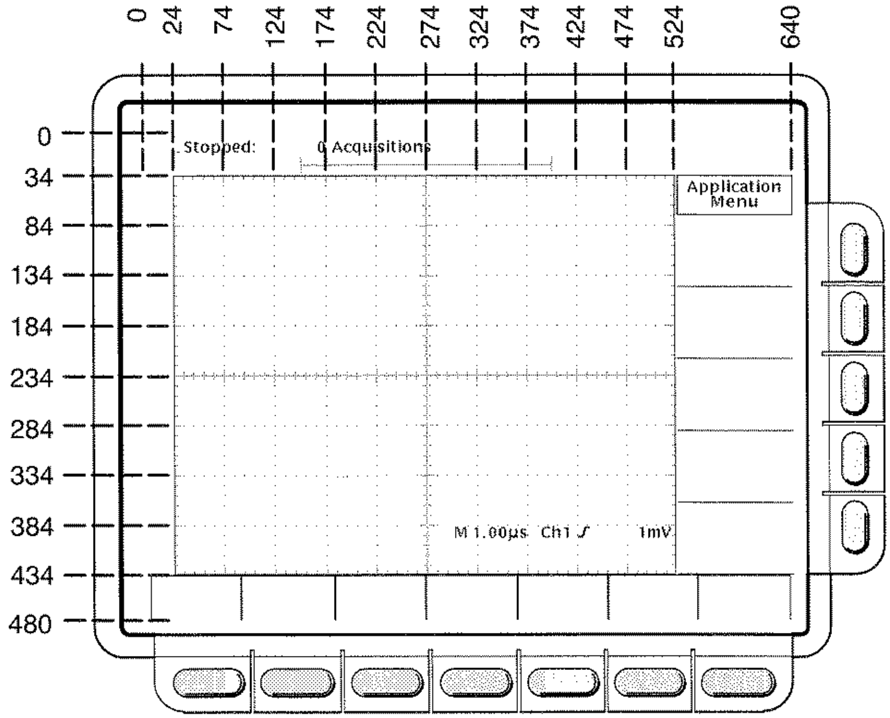


Figure 2-4: Message Window Coordinates

MESSage:SHOW

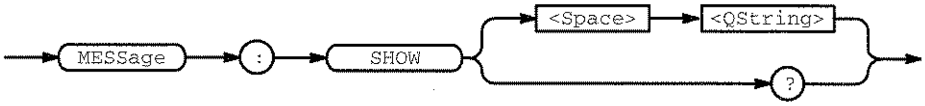
Clears the contents of the message window and displays the new message in the window.

Group: Display

Syntax:

```
MESSage:SHOW <QString>

MESSage:SHOW?
```



Arguments: `<QString>` is the message and can include any of the characters shown in the TDS Character Chart in Appendix A. The maximum length of the message is 1000 characters.

The message is left-justified, and is displayed on a single line starting with the top most line in the window. A line feed character can be embedded in the string to position the message on multiple lines. You can also use white space and tab characters to position the message within a line.

A tab can be sent by sending a tab character (decimal 9) followed by two numeric characters that specify the pixel column relative to the left margin of the message window.

The ESC character followed by the @ character turns inverse video on and off, and can be embedded in the message string. The first ESC character followed by a @ character displays all the text that follows in inverse video until another ESC character followed by a @ character is found in the string.

NOTE

The use of any escape codes other than what is described above may produce unpredictable results.

The label area is the height and width you have set using the MESSage:Box command. The length of the label that fits in the label area depends on the contents of the label, because the width of characters varies. For a complete list of character widths in pixels, see Table A-1 on page A-1.

If the message exceeds the limits of the window, either horizontally or vertically, the portion of the message that exceeds the limits will not be displayed. The message string itself is not altered. The entire message can be returned as a query response regardless of what is displayed in the window.

Examples: `MESSAGE:SHOW "Hello world"`
displays "Hello world" in the upper left corner of the box (you can define the box size with the MESSAGE BOX command).

`MESSAGE:SHOW "Hello ⧫@world⧫@ ... hello`
displays "Hello world ... hello" in the upper left corner of the box and the word "world" is displayed in inverse video. In this example, ⧫ stands for the escape character. The escape character may appear differently for you depending on your GPIB talker-listener program.

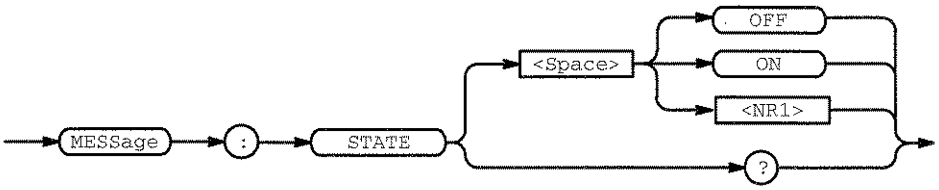
MESSage:STATE

Controls the display of the message window.

Group: Display

Syntax:

```
MESSage:STATE { OFF | ON | <NR1> }
MESSage:STATE?
```



Arguments: `<OFF>` or `<NR1>` = 0 removes the message window from the screen.

`<ON>` or `<NR1>` ≠ 0 displays the message window and its contents on the screen. The size of the window is defined by MESSage:BOX.

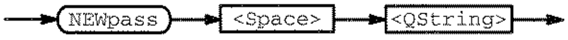
NEWpass (No Query Form)

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 NO TAG.

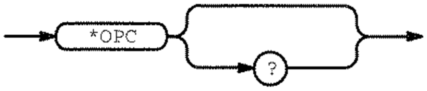
Group: Status and Error

Related Commands: BUSY?, *WAI

Syntax:

```
*OPC

*OPC?
```



The *OPC command allows you to synchronize the operation of the digitizing oscilloscope with your application program. Synchronization methods are described starting on page NO TAG.

Table 2-26: Commands that Generate an Operation Complete Message

Operation	Command
Automatic scope adjustment	AUTOSet EXECute
Internal self-calibration	*CAL
Single sequence acquisition	ACQuire:STATE ON or ACQuire:STATE RUN (when ACQuire:STOPAfter is set to SEQuence)
Hardcopy output	HARDCopy STARt

PASSWord (No Query Form)

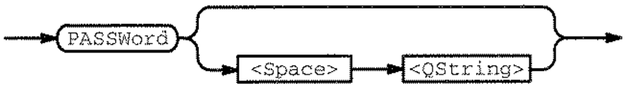
Enables the *PUD and NEWpass set commands. Sending PASSWord without any arguments disables these same commands. Once the password is successfully entered, the *PUD and NEWpass commands are enabled until the digitizing oscilloscope is powered off, or until the FACtory command, the PASSWord command with 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 *PUB and NEWpass set commands.

`PASSWORD`
Disables the *PUB and NEWpass set commands. You can still use the query version of *PUB.

*PSC

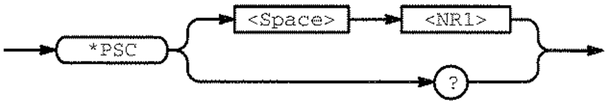
Sets and queries the power-on status flag that controls the automatic power-on handling of the DESER, SRER, and ESER registers. When *PSC is true, the DESER register is set to 255 and the SRER and ESER registers are set to 0 at power-on. When *PSC is false, the current values in the DESER, SRER, and ESER registers are preserved in non-volatile memory when power is shut off and are restored at power-on. For a complete discussion of the use of these registers, see page NO TAG.

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, and disables the power-on clear and allows the digitizing 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

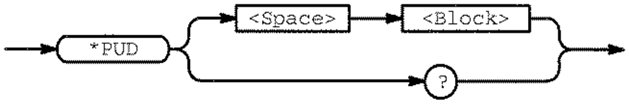
Sets or queries a string of Protected User Data. This data is protected by the PASSWord command. You can modify it only by first entering the correct password. The password is not necessary to query the data.

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.`

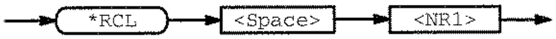
*RCL (No Query Form)

Restores the state of the digitizing 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** item in the front-panel Save/Recall Setup menu.

Group: Save and Recall

Related Commands: DELEte:SETUp, 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 digitizing oscilloscope from a copy of the settings stored in memory location 3.

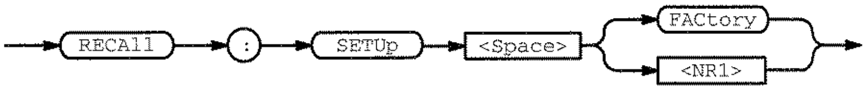
RECAll:SETUp (No Query Form)

Restores a stored or factory front-panel setup of the digitizing oscilloscope. This command is equivalent to selecting **Recall Saved Setup** or **Recall Factory Setup** in the Save/Recall Setup menu.

Group: Save and Recall

Related Commands: DELEte:SETUp, FACtory, *RCL, *RST, *SAV, SAVe:SETUp

Syntax: `RECAll:SETUp { FACtory | <NR1> }`



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`

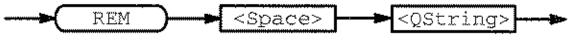
recalls the front panel setup to its factory defaults.

REM (No Query Form)

Specifies a comment. This line is ignored by the instrument.

Group: Miscellaneous

Syntax: `REM <QString>`



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

Examples: `REM "This is a comment"`
is ignored by the instrument.

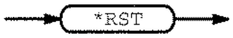
*RST (No Query Form)

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

Group: Status and Error

Related Commands: FACtory, *PSC, *RCL, RECAll:SETUp, *SAV, SAVe:SETUp

Syntax: `*RST`



*RST does the following:

- Returns the instrument settings to the factory defaults (see Appendix D).

The *RST command does not alter the following:

- The state of the IEEE Std 488.1-1987 interface.
- The selected IEEE Std 488.1-1987 address of the digitizing 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.
- The *PUD? response

RS232:BAUd
Option 13 Only

Sets or queries RS-232-C interface transmission speed.

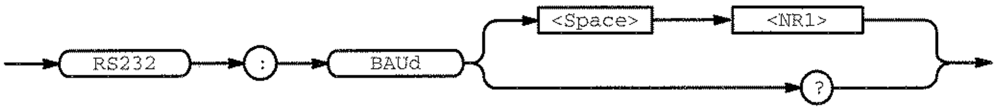
Group: Miscellaneous

Related Commands: RS232:HARDFLAGGING, RS232:PARITY, RS232:SOFTFLAGGING, RS232: STOPBITS, RS232?

Syntax:

```
RS232:BAUd <NR1>
```

```
RS232:BAUd?
```



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

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

RS232:HARDFlagging
Option 13 Only

Sets or queries the input and output hard flagging over the RS-232 port. It uses the RFR (Ready For Receive) and CTS (Clear To Send) lines to control data transmission. On output, the oscilloscope transmits data only when CTS is asserted. When CTS is not asserted, the oscilloscope stops transmitting data. On input, it asserts RFR until the receive queue is full. Then it unasserts RFR to stop transmission from an external printer. CTS remains unasserted until the receive queue is not full. At that time, CTS is asserted again to restart transmission.

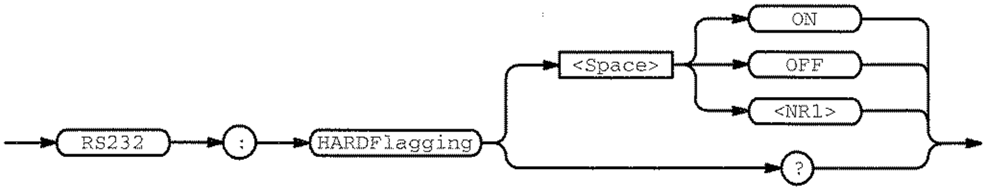
Group: Miscellaneous

Related Commands: RS232:BAUD, RS232:PARITY, RS232:SOFTFLAGGING, RS232:STOPBITS, RS232?

Syntax:

```
RS232:HARDFlagging { ON | OFF | <NR1> }
```

```
RS232:HARDFlagging?
```



Arguments: `<ON> or <NR1>≠ 0` turn on hardflagging.

`<OFF> or <NR1> = 0` turn off hardflagging.

Examples: `RS232:HARDFLAGGING ON`
turns on hard flagging.

RS232:PARity
Option 13 Only

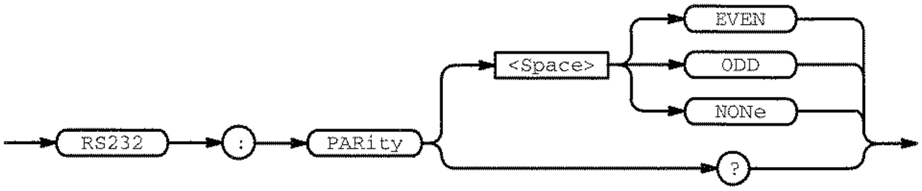
Sets or queries the parity used for all RS-232-C data transfers. When parity is odd or even, the digitizing oscilloscope generates the selected parity on output and checks all input against the selected parity. When parity is none, the digitizing oscilloscope performs no input parity error checks and generates no output parity.

Group: Miscellaneous

Related Commands: RS232:BAUD, RS232:HARDFLAGGING, RS232:SOFTFLAGGING, RS232:STOPBITS, RS232?

Syntax: `RS232:PARity { EVEN | ODD | NONe }`

```
RS232:PARity?
```



Arguments: `EVEN` even parity.

`ODD` odd parity.

`NONe` no parity.

Examples: `RS232:PARITY EVEN`
sets the parity to be even.

RS232:SOFTFlagging
Option 13 Only

Sets or queries the input and output soft flagging over the RS-232 port. It stops transmitting data any time it receives an XOFF (DC3) character. It sends an XOFF character when its 512 byte input buffer has 80 free bytes. The digitizing oscilloscope begins transmitting data again when it receives an XON (DC1) characters. It sends XON when its input buffer is has 100 free bytes.

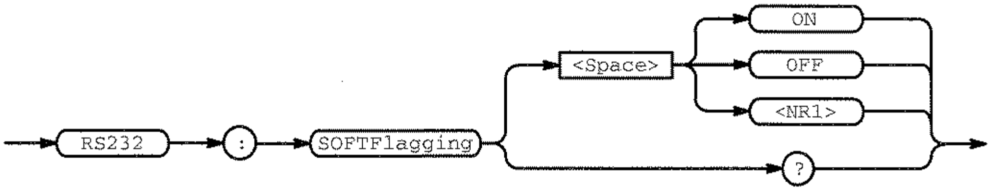
Group: Miscellaneous

Related Commands: RS232:BAUD, RS232:HARDFLAGGING, RS232:PARITY, RS232:STOPBITS, RS232?

Syntax:

```
RS232:SOFTFlagging { ON | OFF | <NR1> }

RS232:SOFTFlagging?
```



Arguments: `<ON> or <NR1>≠ 0` turn on softflagging.

`<OFF> or <NR1> = 0` turn off softflagging.

Examples: `RS232:SOFTFLAGGING ON`
turns on soft flagging.

RS232:STOPBits
Option 13 Only

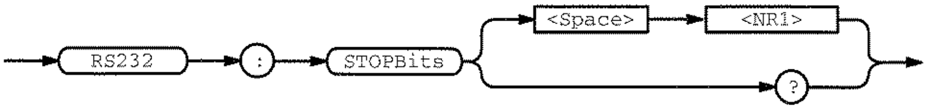
Sets or queries the number of transmission stop bits sent with each character to identify the end of data for that character.

Group: Miscellaneous

Related Commands: RS232:BAUD, RS232:HARDFLAGGING, RS232:PARITY, RS232:SOFTFLAGGING, RS232?

Syntax:

```
RS232:STOPBits <NR1>
RS232:STOPBits?
```



Arguments: `<1>` use one stop bit.

`<2>` use two stop bits.

Examples: `RS232:STOPBITS 1` sets the number of stop bits to 1.

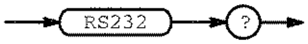
RS232? (Query Only)
Option 13 Only

Queries the RS232 settings.

Group: Miscellaneous

Related Commands: RS232: BAUD, RS232: HARDFLAGGING, RS232: PARITY, RS232:SOFTFLAGGING, RS232: STOPBITS

Syntax: `RS232?`



Arguments: None

Examples: `RS232?` queries for RS232 settings.

It might return:

```
RS232 BAUD: 9600, SOFTFLAGGING: OFF, HARDFLAGGING: ON,
   PARITY: NONE, STOPBITS: 1
```

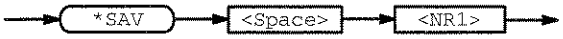
*SAV (No Query Form)

(Save) Stores the state of the digitizing oscilloscope into a specified memory location. You can later use the *RCL command to restore the digitizing 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: DELEte:SETUp, FACtory, *RCL, RECAll:SETUp, SAVe:SETUp

Syntax: `*SAV <NR1>`



Arguments: `<NR1>` is a value in the range from 1 to 10 and specifies a location. Using an out-of-range value causes an execution error. Any settings that have been stored previously at this location will be overwritten.

Examples: `*SAV 2`
saves the current settings in memory location 2.

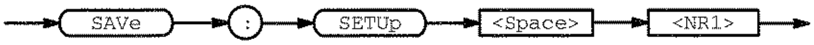
SAVe:SETUp (No Query Form)

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

Group: Save and Recall

Related Commands: DELEte:SETUp, 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.

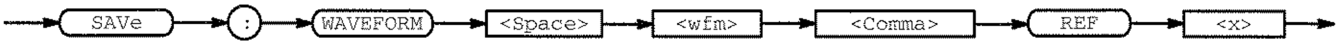
SAVe:WAVEFORM (No Query Form)

Stores a waveform in one of four reference memory locations. This command is equivalent to selecting the **Save Waveform** item in the Save/Recall Waveform menu.

Group: Save and Recall

Related Commands: DELEte:WAVEFORM

Syntax: `SAVe:WAVEFORM <wfm><Comma>REF<x>`



Arguments: `<wfm>` is CH<x>, MATH<x>, or REF<x>, and is the waveform that will be saved.

`REF<x>` is the location where the waveform will be stored.

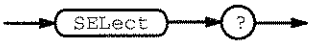
Examples: `SAVE:WAVEFORM MATH2,REF1`
saves the math 2 waveform in reference memory location 2.

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;CH3 0;CH4 0;MATH1 0; MATH2 0;MATH3 0;REF1 0;REF2 0;REF3 0;REF4 0`

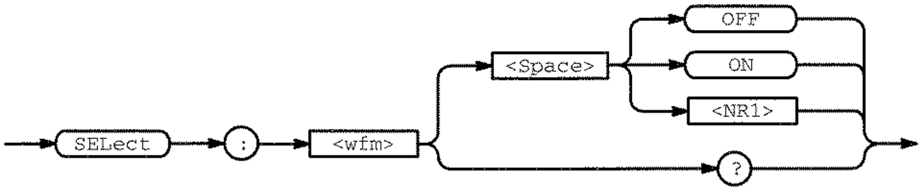
SELect:<wfm>

Controls the display and selection of waveforms. There can be up to eleven 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 **CH** or **MORE** button. <wfm> can be CH<x>, MATH<x>, 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 selected.

SELect:CONTROl

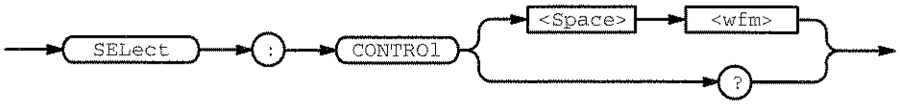
Sets or queries the waveform that is currently affected by the cursor and vertical commands.

Group: Vertical

Syntax:

```
SELect:CONTROl <wfm>

SELect:CONTROl?
```



Arguments: `<wfm>` is CH<x>, MATH<x>, 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 digitizing oscilloscope's settings, except for configuration information for the calibration values. You can use this string to return the digitizing oscilloscope to the state it was in when you made the SET? query. This command is identical to the *LRN? command.

Group: Miscellaneous

Related Commands: HEADer, *LRN?, VERBose

Syntax:

```
SET?
```

NOTE

The SET? query always returns a string with command headers, regardless of the setting of the HEADer command. This is because the returned string is intended to be able to be sent back to the digitizing 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: `SET?`

a partial return string may look like this:

```
:ACQUIRE:STOPAFTER RUNSTOP;STATE 1;MODE SAMPLE;NUMENV
10;NUMAVG 16;REPET 1;:APPMENU:TITLE "Application
Menu";LABEL:BOTTOM1 "";BOTTOM2 "";BOTTOM3 "";BOTTOM4
""; BOTTOM5 "";BOTTOM6 "";BOTTOM7 "";RIGHT1 "";RIGHT2
""; RIGHT3 "";RIGHT4 "";RIGHT5 "";:HEADER 1;:VERBOSE
1; :ALIAS:STATE 0;:DISPLAY:FORMAT YT;STYLE VEC-
TORS;FILTER SINX;PERSISTENCE 500.0E-3;GRATICULE
FULL;TRIGT 1;INTENSITY:OVERALL 85;WAVEFORM 75;TEXT
60;CONTRAST 150;:MESSAGE:SHOW "hello";STATE 1;BOX
74,84,475,135;:LOCK NONE; :HARDCOPY:FORMAT EPSI-
MAGE;PORT GPIB;LAYOUT PORTRAIT;
```

*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 NO TAG.

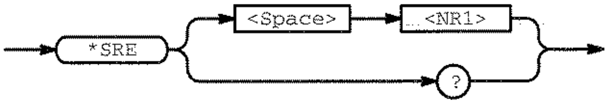
Group: Status and Error

Related Commands: *CLS, DESE, *ESE, *ESR?, EVENT?, EVMSg?, FACtory, *PSC, *STB?

Syntax:

```
*SRE <NR1>
```

```
*SRE?
```



Arguments: `<NR1>` is a value in the range from 0 to 255. The binary bits of the SRER are set according to this value. Using an out-of-range value causes an execution error. The power-on default for SRER is 0 if `*PSC` is 1. If `*PSC` is 0, the SRER maintains its value through a power cycle.

Examples: `*SRE 48`

sets the bits in the SRER to 00110000 binary.

`*SRE?`

might return a value of `32`, showing that the bits in the SRER have the binary value 00100000.

*STB? (Query Only)

(Read Status Byte) query returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. For a complete discussion of the use of these registers, see page NO TAG.

Group: Status and Error

Related Commands: *CLS, DESE, *ESE, *ESR?, EVENT?, EVMSg?, FACtory, *SRE

Syntax: `*STB?`

Returns: `<NR1>`

Examples: `*STB?`

might return the value `96`, showing that the SBR contains the binary value 01100000.

TEKSecure

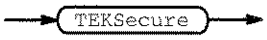
Initializes both waveform and setup memories. This overwrites any previously stored data.

It writes 0's in all waveform reference memory regardless of selected record length and puts all setups in the factory init state.

It then verifies that the waveform and setup memory are in the desired state. It displays a pass or a fail notifier on completion.

Group: Miscellaneous

Syntax: `TEKSecure`



TIMe

Sets or queries the time that the digitizing oscilloscope can display.

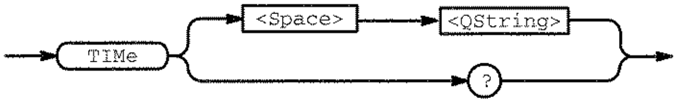
Group: Miscellaneous

Related Commands: DATE, DISplay: CLOCk

Syntax:

```
TIMe <QString>

TIMe?
```



Arguments:

`<QString>` is a date in the form `"hh:mm:ss"`.
`hh` refers to the hour number from 1 to 24.
`mm` refers to the minute number in the hour from 0 to 59.
`ss` refers to the seconds number in the minute from 0 to 59.
There must be a colon after the `hh` and after the `mm`.

Examples:

```
TIME "01:24:00"
```

specifies that the time is set to 01:24 AM.

TRIGger

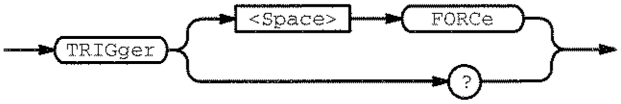
Forces a trigger event to occur and the TRIGger query returns the current trigger parameters.

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:LOGIC:CLASS PATTERN;FUNCTION AND;WHEN TRUE;THRESHOLD:CH1 1.40E+0; CH2 1.200E+0;CH3 1.200E+0;CH4 1.200E+0; :TRIGGER:MAIN:LOGIC:INPUT:CH1 HIGH;CH2 X;CH3 X; :TRIGGER:MAIN:LOGIC:PATTERN:INPUT:CH4 X; :TRIGGER:MAIN:LOGIC:STATE:INPUT:CH4 RISE; :TRIGGER:MAIN:PULSE:CLASS GLITCH;SOURCE CH1; GLITCH:WIDTH 2.0E-9;FILTER ACCEPT;POLARITY POSITIVE; :TRIGGER:MAIN:PULSE:RUNT:POLARITY POSITIVE; THRESHOLD:HIGH 2.00E+0;LOW 800.0E-3; :TRIGGER:MAIN:PULSE:WIDTH:LOWLIMIT 2.0E-9; HIGHLIMIT 2.0E-9;WHEN WITHIN;POLARITY POSITIVE; :TRIGGER:DELAY:TYPE EDGE;LEVEL -480.0E-3;BY TIME; EDGE:SOURCE CH1;SLOPE RISE;COUPLING DC; :TRIGGER:DELAY:TIME 16.0E-9;EVENTS:COUNT 2`

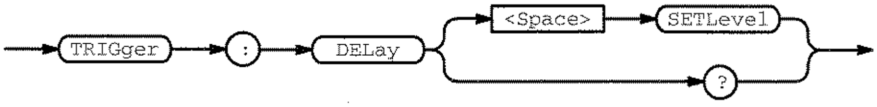
TRIGger:DELay

Sets the delayed trigger level and returns the current delayed trigger parameters.

Group: Trigger

Syntax: `TRIGger:DELay SETLevel`

`TRIGger:DELay?`



Arguments: `SETLevel` sets the delayed trigger level to half way between the MIN and MAX amplitudes of the trigger source input. This is equivalent to selecting **Set to 50%** in the Delayed Edge Level side menu.

Examples: `TRIGGER:DELAY SETLEVEL`
sets the delayed trigger level to 50% of MAX and MIN.

`TRIGGER:DELAY?`
might return `:TRIGGER:DELAY:TYPE EDGE;LEVEL 0.0E+0;BY TIME;EDGE:SOURCE CH1;SLOPE RISE;COUPLING DC;:TRIGGER:DELAY:TIME 16.0E-9;EVENTS:COUNT 2`

TRIGger:DELay:BY

Selects whether the delayed trigger occurs after a specified number of events or a specified period of time after the main trigger. This is equivalent to setting **Delay by** in the Delayed Trig menu.

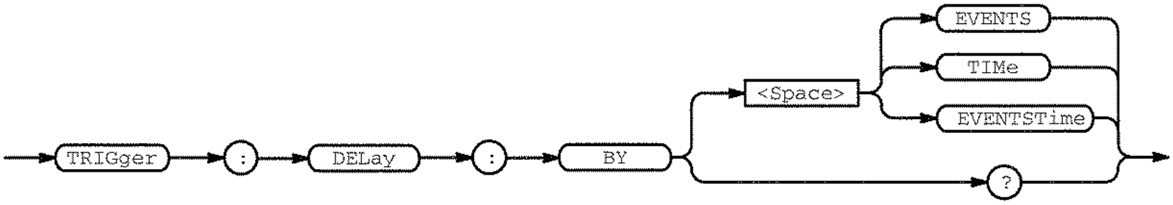
Group: Trigger

Related Commands: TRIGger:DELay:EVENTS:COUNt, TRIGger:DELay:TIMe

Syntax:

```
TRIGger:DELay:BY
    { EVENTS | TIMe | EVENTSTime (TDS 520/540/620/640 only)}

TRIGger:DELay:BY?
```



Arguments: `EVENTS` sets the delayed trigger to occur after a set number of trigger events after the main trigger. The number of events is specified by TRIGger:DELay:EVENTS:COUNt.

`TIMe` sets the delayed trigger to occur a set time after the main trigger event. The time period is specified by TRIGger:DELay:TIMe.

`EVENTSTime` (TDS 520/540/620/640 only) only sets a specified time after a specified number of delay trigger trigger events—after the main trigger event. For example in examining a pulse train, you might use the main trigger to detect the start of the train, then use the delay by events to go to the position of interest within the pulse train, then use the time delay to wait a specified time period before starting the data acquisition.

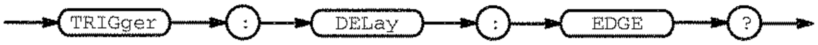
Examples: `TRIGGER:DELAY:BY?`
might return `EVENTS`.

TRIGger:DELay:EDGE? (Query Only)

Returns the coupling, slope, and source for the delayed trigger.

Group: Trigger

Syntax: `TRIGger:DELay:EDGE?`



Examples: `TRIGGER:DELAY:EDGE?`
might return `:TRIGGER:EDGE:SOURCE CH1;SLOPE RISE; COUPLING DC`

TRIGger:DELay:EDGE:COUPling

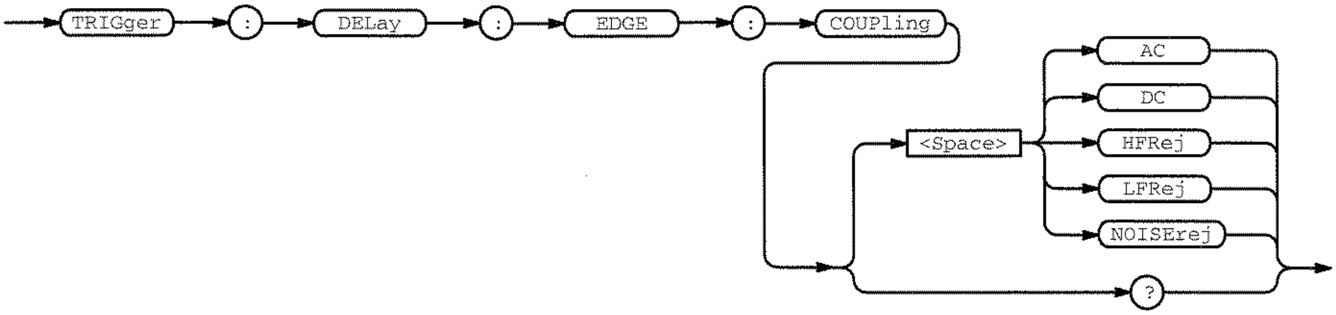
Selects the type of coupling for the delayed trigger. This command is equivalent to selecting **Coupling** in the Delayed Trig menu.

Group: Trigger

Syntax:

```
TRIGger:DELay:EDGE:COUPling { AC | DC | HFRej | LFRej |
   NOISErej }
```

```
TRIGger:DELay: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.

Examples: `TRIGGER:DELAY:EDGE:COUPLING DC`
sets the delay trigger to DC coupling.

`TRIGGER:DELAY:EDGE:COUPLING?`
might return `LFREJ` for the delayed trigger coupling.

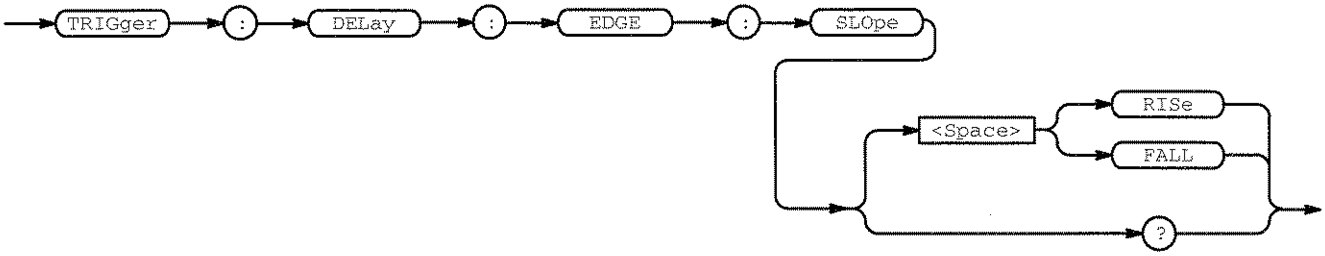
TRIGger:DELay:EDGE:SLOpe

Selects either a rising or falling edge for the delayed trigger. This command is equivalent to selecting **Slope** in the Delayed Trig menu.

Group: Trigger

Syntax: `TRIGger:DELay:EDGE:SLOpe { RISe | FALL }`

`TRIGger:DELay: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:DELAY:EDGE:SLOPE?`
might return `RISE`, indicating that the delayed trigger occurs on the rising edge.

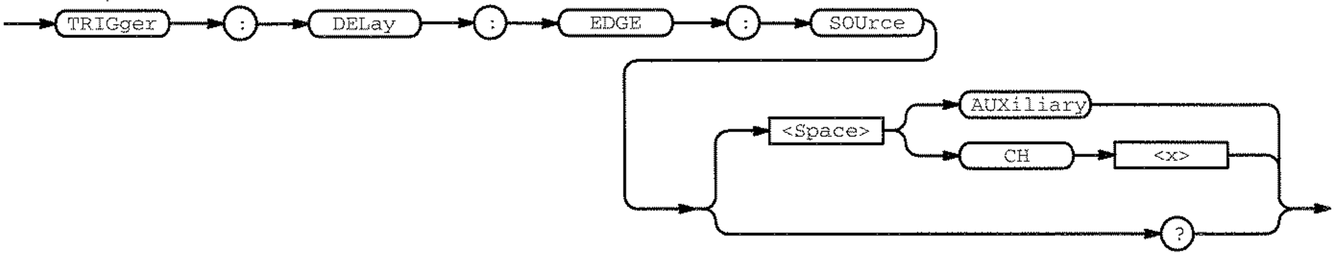
TRIGger:DELay:EDGE:SOUrce

Selects the source for the delayed trigger. This command is equivalent to selecting **Source** in the Delayed Trig menu.

Group: Trigger

Syntax: `TRIGger:DELay:EDGE:SOUrce { AUXiliary` (not available on TDS 520) `| CH<x> }`

`TRIGger:DELay:EDGE:SOUrce?`



Arguments: `AUXiliary` specifies an external trigger using the Auxiliary Trigger Input connector that is located on the rear panel of the instrument. The TDS 520 doesn't have an Auxiliary Trigger input and so doesn't supprt this argument.

`CH<x>` specifies one of the input channels.

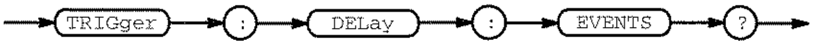
Examples: `TRIGGER:DELAY:EDGE:SOURCE CH1`
selects channel 1 as the input source for the delayed trigger.

TRIGger:DELay:EVENTS? (Query Only)

Returns the current delayed trigger event parameter.

Group: Trigger

Syntax: `TRIGger:DELay:EVENTS?`



Examples: `TRIGGER:DELAY:EVENTS?`
might return `:TRIGGER:DELAY:EVENTS:COUNT 2`

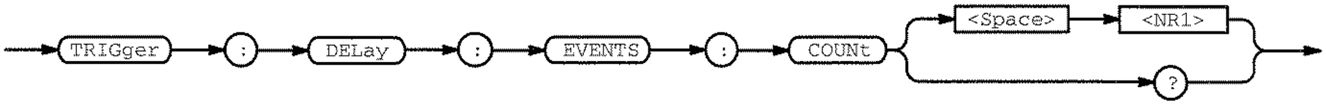
TRIGger:DELay:EVENTS:COUNt

Sets or queries the number of events that must occur before the delayed trigger occurs when TRIGger:DELay:BY is set to EVENTS. This is equivalent to setting the **Delay by Events** count in the Delayed Edge Delay side menu.

Group: Trigger

Syntax: `TRIGger:DELay:EVENTS:COUNt <NR1>`

`TRIGger:DELay:EVENTS:COUNt?`



Arguments: `<NR1>` is the number of delayed edge trigger events. The range is 2 to 10E7.

Examples: `TRIGGER:DELAY:EVENTS:COUNT 4`
specifies that the delayed trigger will occur four trigger events after the main trigger.

`TRIGGER:DELAY:EVENTS:COUNT?`
might return `2`, indicating that 2 events must occur after the main trigger and before the delayed trigger can occur.

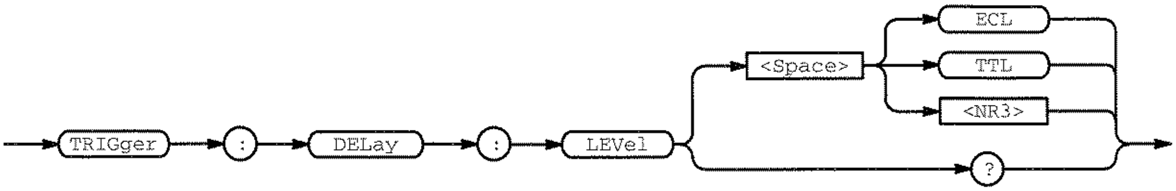
TRIGger:DELay:LEVel

Selects the level of the delayed trigger. This command is equivalent to setting **LEVel** in the Delayed Trig menu.

Group: Trigger

Syntax: `TRIGger:DELay:LEVel { ECL | TTL | <NR3> }`

`TRIGger:DELay:LEVel?`



Arguments: `ECL` specifies a preset ECL level of –1.3 V.

`TTL` specifies a preset TTL level of 1.4 V.

`<NR3>` is the delayed trigger level, in volts.

Examples: `TRIGGER:DELAY:LEVEL 2E-3`
sets the delayed trigger level to 2 mV.

TRIGger:DELay:TIMe

Sets or queries the delay time when HORizontal:DELay:MODe is set to TRIGAfter. This command is identical to the HORizontal:DELay:TIMe:TRIGAfter command, and is equivalent to setting the **Delay by Time** value in the Delayed Edge Delay side menu.

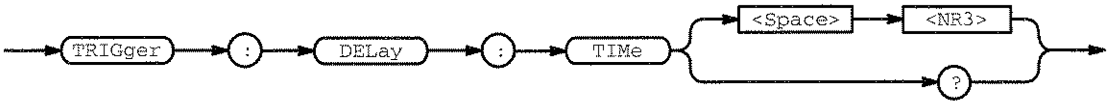
When HORizontal:DELay:MODe is set to RUNSAfter, the delay time is set by the HORizontal:DELay:TIMe:RUNSAfter command.

Group: Trigger

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

Syntax: `TRIGger:DELay:TIMe <NR3>`

`TRIGger:DELay:TIMe?`



Arguments: `<NR3>` is the delay time, in seconds.

Examples: `TRIGGER:DELAY:TIME 4E-6`
sets the delay time to 4 µs.

TRIGger:DELay:TYPe

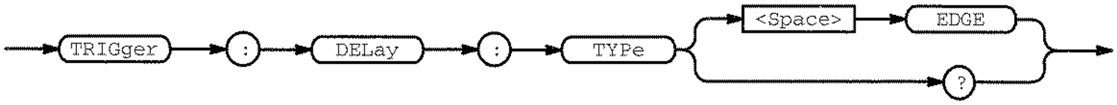
Sets or queries the type of delayed trigger.

Group: Trigger

Syntax:

```
TRIGger:DELay:TYPe EDGE

TRIGger:DELay:TYPe?
```



Arguments: `EDGE` is a normal trigger. A trigger event occurs when a signal passes through a specified voltage level in a specified direction. Use the TRIGger:DELay:LEVel and TRIGger:DELay:EDGE:SLOpe commands to set the voltage level and direction respectively.

Examples: `TRIGGER:DELAY:TYPE?`
always returns `EDGE` as the type of delayed trigger.

TRIGger:MAIn

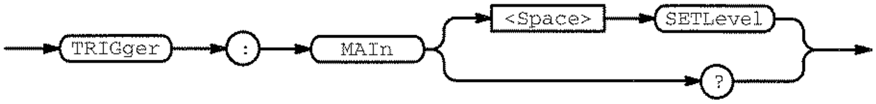
Sets the main trigger level and returns the current main trigger parameters.

Group: Trigger

Syntax:

```
TRIGger:MAIn SETLevel

TRIGger:MAIn?
```



Arguments: `SETLevel` sets the main trigger level to half way between the MIN and MAX amplitudes of the trigger source input. This is equivalent to pressing the front-panel **SET LEVel TO 50%** button.

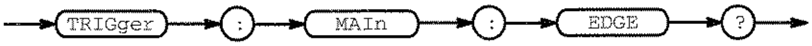
Examples: `TRIGGER:MAIN SETLEVEL`
sets the main trigger level mid way between MAX and MIN.

TRIGger:MAIn:EDGE? (Query Only)

Returns the trigger coupling, source, and slope for the main edge trigger.

Group: Trigger

Syntax: `TRIGger:MAIn:EDGE?`



Examples: `TRIGGER:MAIN:EDGE?`
might return `SOURCE CH1;COUPLING DC;SLOPE RISE`

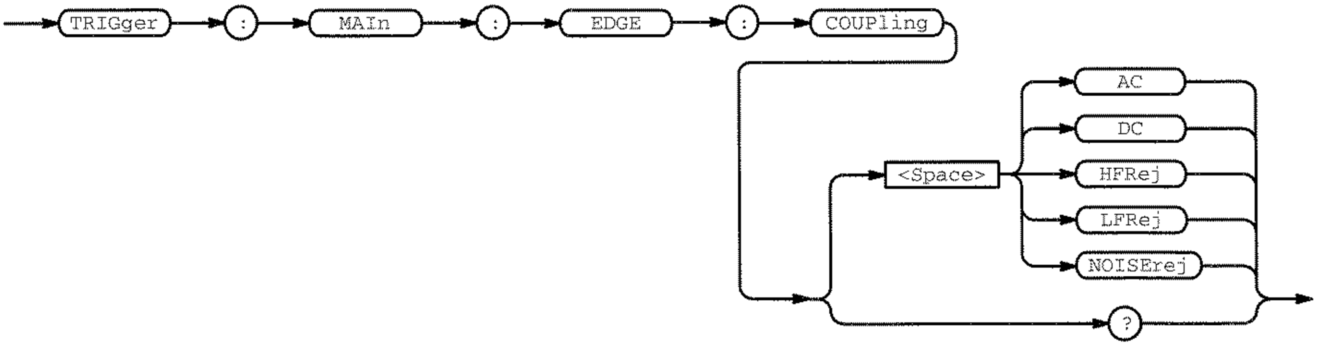
TRIGger:MAIn:EDGE:COUPling

Sets or queries the type of coupling for the main edge trigger. This is equivalent to setting **Coupling** in the Trigger menu.

Group: Trigger

Syntax: `TRIGger:MAIn:EDGE:COUPling { AC | DC | HFRej | LFRej | NOISErej }`

`TRIGger:MAIn:EDGE:COUPling?`



Arguments: `AC` selects AC trigger coupling.

`DC` selects DC trigger coupling.

`HFRej` coupling removes the high frequency components of the DC signal.

`LFRej` coupling removes the low frequency components of the AC signal.

`NOISErej` selects DC low sensitivity. It requires added signal amplitude for more stable, less false triggering.

Examples: `TRIGGER:MAIN:EDGE:COUPLING DC`
sets the main edge trigger coupling to DC.

TRIGger:MAIn:EDGE:SLOpe

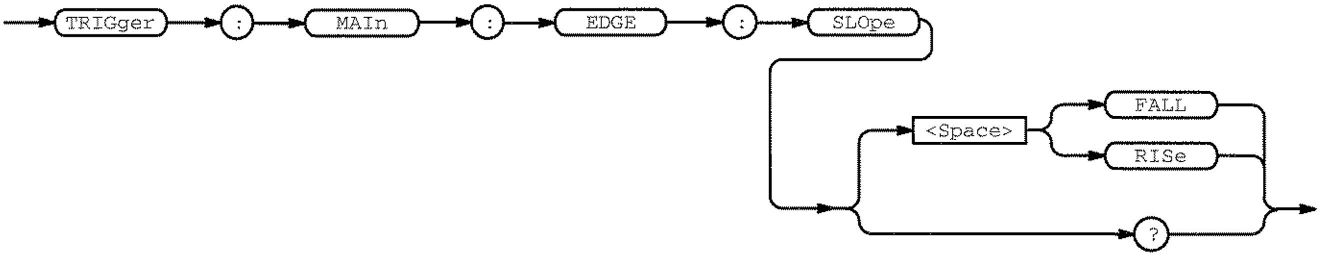
Selects a rising or falling slope for the main edge trigger. This is equivalent to setting **Slope** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:EDGE:SLOpe { FALL | RISe }
```

```
TRIGger:MAIn:EDGE:SLOpe?
```



Arguments: `FALL` specifies to trigger on the falling or negative edge of a signal.

`RISe` specifies to trigger on the rising or positive edge of a signal.

Examples: `TRIGGER:MAIN:EDGE:SLOPE RISE`
sets the main edge trigger to occur on the rising slope.

TRIGger:MAIn:EDGE:SOUrce

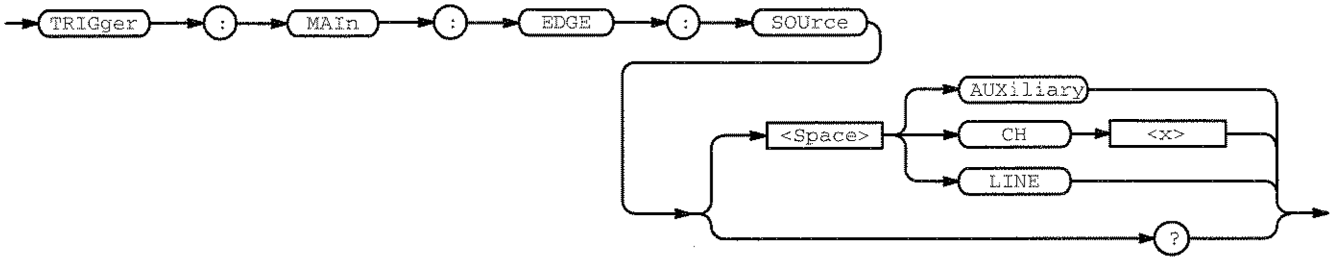
Sets or queries the source for the main edge trigger. This is equivalent to setting **Source** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:EDGE:SOUrce { AUXiliary (not available on TDS 520)
    | CH<x> | LINE }
```

```
TRIGger:MAIn:EDGE:SOUrce?
```



Arguments: `AUXiliary` specifies an external trigger using the Auxiliary Trigger Input connector that is located on the rear panel of the instrument. The TDS 520 doesn't have an Auxiliary Trigger input and so doesn't supprt this argument.

`CH<x>` specifies one of the input channels.

`LINE` specifies AC line voltage.

Examples: `TRIGGER:MAIN:EDGE:SOURCE LINE`
specifies the AC line voltage as the main edge trigger source.

`TRIGGER:MAIN:EDGE:SOURCE?`
might return `CH2` for the main edge trigger source.

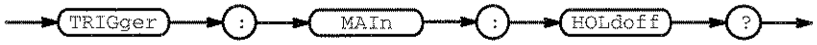
`TRIGGER:MAIN:EDGE:SOURCE?`
might return `CH2` for the main edge trigger source.

TRIGger:MAIn:HOLdoff? (Query Only)

Returns the main trigger holdoff value.

Group: Trigger

Syntax: `TRIGger:MAIn:HOLdoff?`



Examples: `TRIGGER:MAIN:HOLDOFF?`
might return `:TRIGGER:MAIN:HOLDOFF:VALUE 0`.

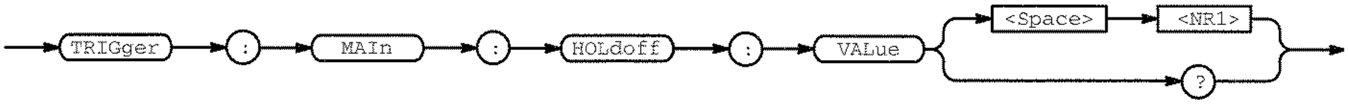
TRIGger:MAIn:HOLdoff:VALue

Sets or queries the main trigger holdoff value. This is equivalent to setting **Holdoff** in the Mode & Holdoff side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:HOLdoff:VALue <NR1>`

```
TRIGger:MAIn:HOLdoff:VALue?
```



Arguments: `<NR1>` is from 0 to 100, and is a percent of the holdoff range.

Examples: `TRIGGER:MAIN:HOLDOFF:VALUE 10`
set the holdoff value to be 10% of the holdoff range.

TRIGger:MAIn:LEVel

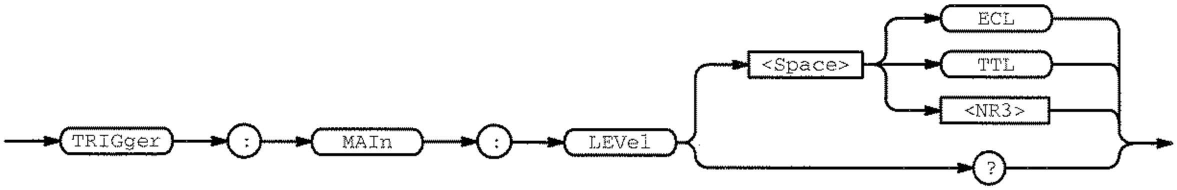
Sets the main trigger level. This command is equivalent to adjusting the front-panel **TRIGGER MAIN LEVEL** knob.

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>` is the main trigger level, in volts.

Examples: `TRIGGER:MAIN:LEVEL?`
might return `TTL`, indicating that the main edge trigger is set to 1.4 V.

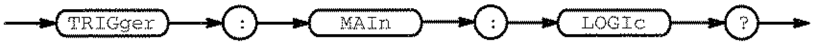
TRIGger:MAIn:LOGIc? (Query Only)

TDS 520/540/620/640 Only

Returns all main logic trigger parameters.

Group: Trigger

Syntax: `TRIGger:MAIn:LOGIc?`



Examples: `TRIGGER:MAIN:LOGIC?`
might return `:TRIGGER:MAIN:LOGIC:CLASS PATTERN; FUNCTION AND;WHEN TRUE;THRESHOLD:CH1 0;CH2 0;CH3 0; CH4 0;:TRIGGER:MAIN:LOGIC:INPUT:CH1 HIGH;CH2 X; CH3 X;:TRIGGER:MAIN:LOGIC:PATTERN:INPUT:CH4 X; :TRIGGER:MAIN:LOGIC:STATE:INPUT:CH4 RISE`

TRIGger:MAIn:LOGIc:CLAss

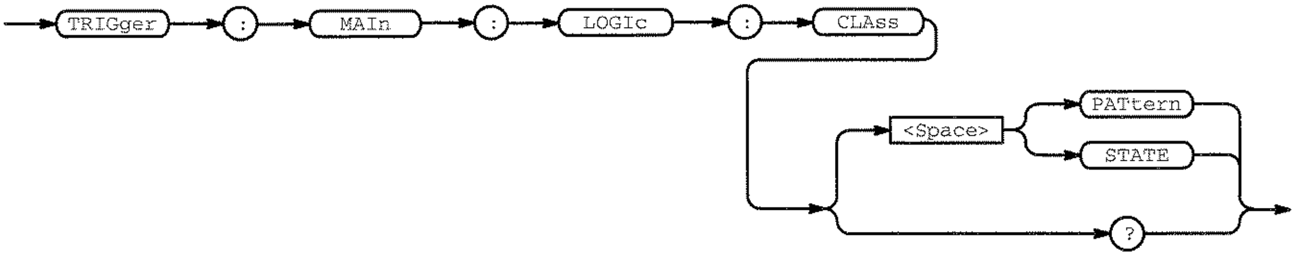
TDS 520/540/620/640 Only

Sets or queries the type of main logic trigger. This command is equivalent to selecting **Class** in the Trigger menu when the **Type** is set to Logic.

Group: Trigger

Syntax: `TRIGger:MAIn:LOGIc:CLAss { PATtern | STATE }`

`TRIGger:MAIn:LOGIc:CLAss?`



Arguments: `PATtern` means that the instrument triggers when the specified logical combinations of channels 1, 2, 3, and 4 are met.

`STATE` means that the instrument triggers when the specified conditions of channels 1, 2, and 3 are met after the channel 4 condition is met.

Examples: `TRIGGER:MAIN:LOGIC:CLASS?`
might return `STATE`.

TRIGger:MAIn:LOGIc:FUNCtion

TDS 520/540/620/640 Only

Sets or queries the logical combination of the input channels for the main logic trigger.

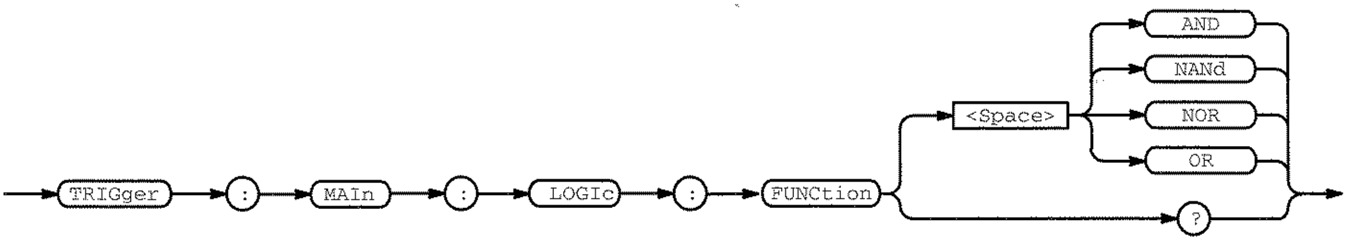
When TRIGger:MAIn:LOGIc:CLAss is PATtern, this command applies to channels 1, 2, 3, and 4. When TRIGger:MAIn:LOGIc:CLAss is STATE, only channels 1, 2, and 3 are logically combined. This command is equivalent to selecting the function in the Logic Pattern Function side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:LOGIc:FUNCtion { AND | NANd | NOR | OR }
```

```
TRIGger:MAIn:LOGIc:FUNCtion?
```



Arguments: AND specifies that the instrument will trigger if all the conditions are true.

NANd specifies that the instrument will trigger if any of the conditions are false.

NOR specifies that the instrument will trigger if all of the conditions are false.

OR specifies that the instrument will trigger if any of the conditions are true.

Examples:

```
TRIGGER:MAIN:LOGIC:FUNCTION NOR
```

sets the logical combination of channels to be true when none of the conditions are true.

```
TRIGGER:MAIN:LOGIC:FUNCTION?
```

might return NAND.

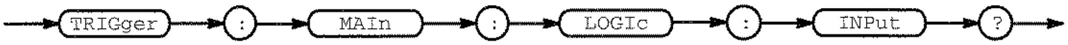
TRIGger:MAIn:LOGIc:INPut? (Query Only)

TDS 520/540/620/640 Only

Returns the main logic trigger input for all channels.

Group: Trigger

Syntax: `TRIGger:MAIn:LOGIc:INPut?`



Examples: `TRIGGER:MAIN:LOGIC:INPUT?`
might return `:TRIGGER:MAIN:LOGIC:INPUT:CH1 HIGH;CH2 X; CH3 X`

TRIGger:MAIn:LOGIc:INPut:CH<x>

TDS 520/540/620/640 Only

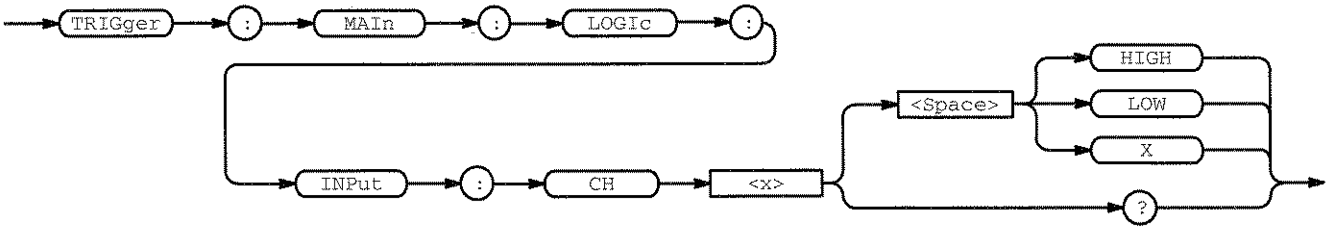
Sets or queries the main logic trigger input for the specified channel. The channel is specified by <x> and is 1, 2, or 3. This is equivalent to setting the inputs in the Logic Pattern Inputs side menu.

Group: Trigger

Related Commands: TRIGger:MAIn:LOGIc:CLAss

Syntax: `TRIGger:MAIn:LOGIc:INPut:CH<x> { HIGH | LOW | X }`

`TRIGger:MAIn:LOGIc:INPut:CH<x>?`



Arguments: `HIGH` specifies logic high.

`LOW` specifies logic low.

`X` specifies a don't care state.

Examples: `TRIGGER:MAIN:LOGIC:INPUT:CH2 LOW`
sets the main logic trigger input to logic low for channel 2.

TRIGger:MAIn:LOGIc:PATtern:INPut:CH4

TDS 520/540/620/640 Only

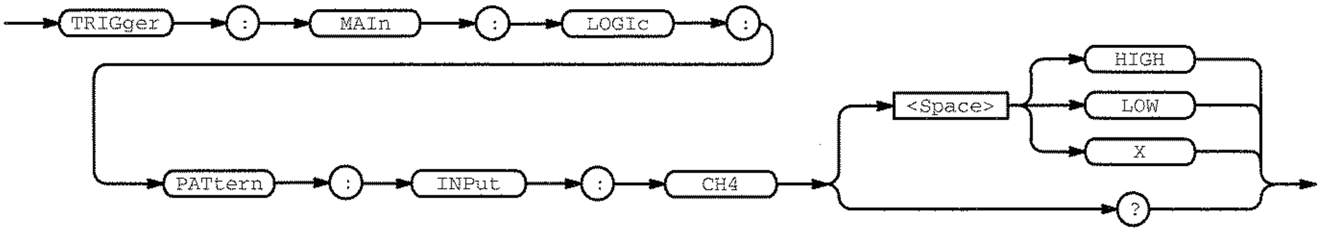
Sets or queries the main logic trigger input for channel 4. These are the inputs used when TRIGger:MAIn:LOGIc:CLAss is set to PATtern. This is equivalent to setting the channel 4 input in the Logic Pattern Inputs side menu.

Group: Trigger

Related Commands: TRIGger:MAIn:LOGIc:CLAss

Syntax: `TRIGger:MAIn:LOGIc:PATtern:INPut:CH4 { HIGH | LOW | X }`

`TRIGger:MAIn:LOGIc:PATtern:INPut:CH4?`



Arguments: `HIGH` specifies logic high.

`LOW` specifies logic low.

`X` specifies a don't care state.

Examples: `TRIGGER:MAIN:LOGIC:PATTERN:INPUT:CH4 LOW`
sets the main logic trigger input to logic low for channel 4 when the logic class is set to `PATtern`.

`TRIGGER:MAIN:LOGIC:PATTERN:INPUT:CH4?`
might return `X`, indicating that the logic input for channel 4 is don't care.

TRIGger:MAIn:LOGIc:PATtern:WHEn

TDS 520/540/620/640 Only

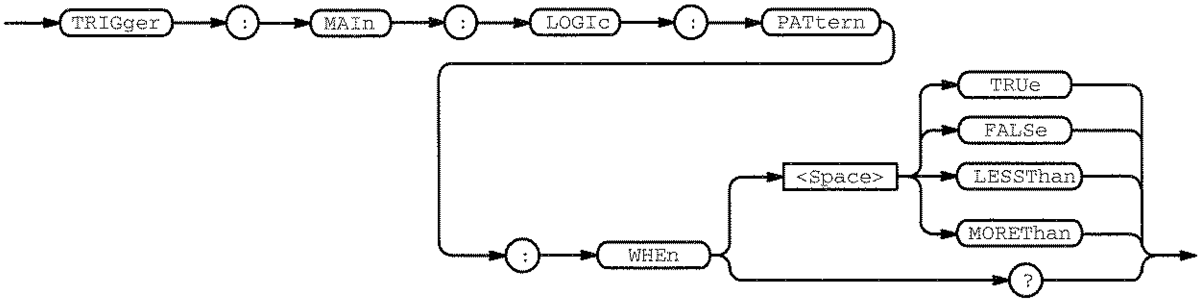
Sets or queries a condition for generating a main logic pattern trigger.

Group: Trigger

Syntax:

```
TRIGger:MAIn:LOGIc:PATtern:WHEn { TRUe | FALSe | LESSThan
    | MOREThan }
```

```
TRIGger:MAIn: LOGIc: PATtern: WHEn?
```



Arguments: `TRUe` specifies the trigger to occur when the pattern becomes true.

`FALSe` specifies the trigger to occur when the pattern becomes false.

`LESSThan` specifies trigger to occur if the specific pattern is true less than the LESSLimit. (see Figure 2-5 and `TRIGger:MAIn:LOGic:PATtern:WHEn:LESSLimit`) Trigger is evaluated at the true–false transition.

`MOREThan` specifies trigger to occur if the specific pattern is true longer than the more limit. (see Figure 2-5 and `TRIGger:MAIn:LOGic:PATtern:WHEn:MORELimit`) Trigger is evaluated at the true–false transition.

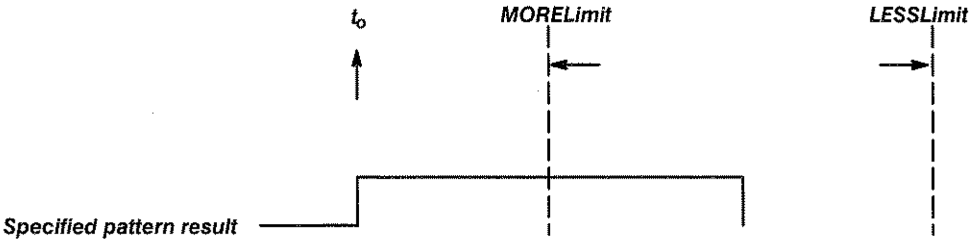


Figure 2-5: LESSThan and MOREThan Arguments

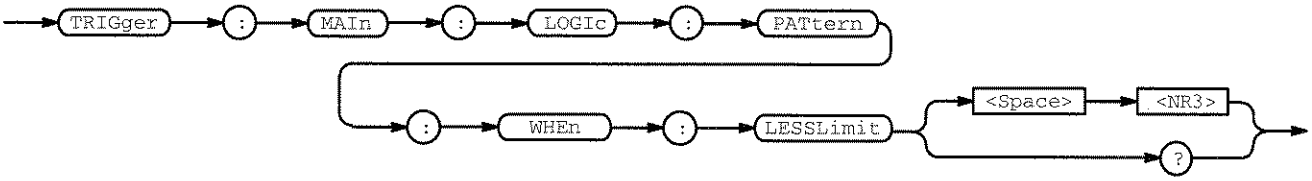
TRIGger:MAIn:LOGIc:PATtern:WHEn:LESSLimit
TDS 520/540/620/640 Only

Sets or queries the maximum time the selected pattern may be true and still generate a main logic pattern trigger.

Group: Trigger

Syntax:

```
TRIGger:MAIn:LOGIc:PATtern:WHEn:LESSLimit <NR3>
TRIGger:MAIn: LOGIc: PATtern: WHEn: LESSLimit?
```



Arguments: <NR3> time to hold pattern true.

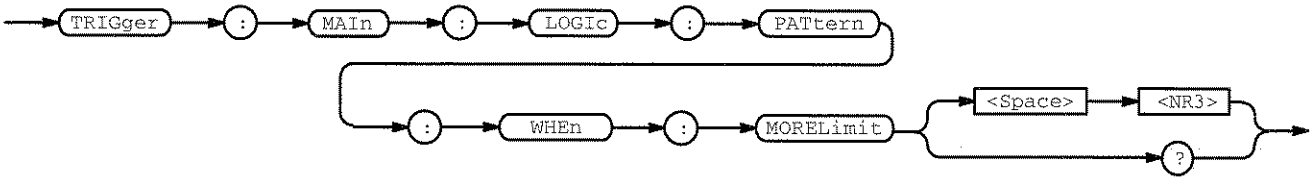
TRIGger:MAIn:LOGIc:PATtern:WHEn:MORELimit
TDS 520/540/620/640 Only

Sets or queries the minimum time the selected pattern may be true and still generate a main logic pattern trigger.

Group: Trigger

Syntax:

```
TRIGger:MAIn:LOGIc:PATtern:WHEn:MORELimit <NR3>
TRIGger:MAIn: LOGIc: PATtern: WHEn: MORELimit?
```



Arguments: <NR3> time to hold pattern true.

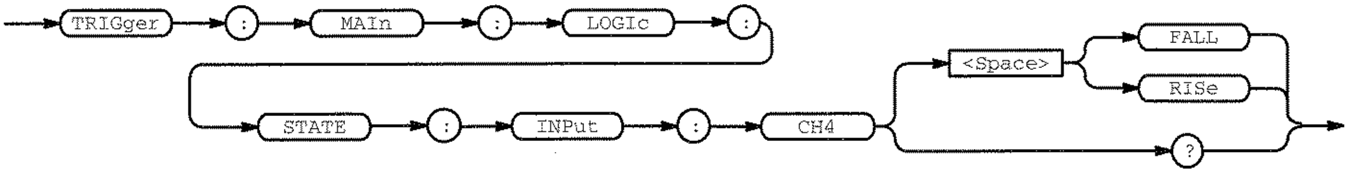
TRIGger:MAIn:LOGIc:STATE:INPut:CH4
TDS 520/540/620/640 Only

Sets or queries the main logic trigger input for channel 4. This input is used when TRIGger:MAIn:LOGIc:CLAss is set to STATE. This is equivalent to setting the channel 4 input in the Logic Pattern Inputs side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:LOGIc:STATE:INPut:CH4 { FALL | RISe }
TRIGger:MAIn:LOGIc:STATE:INPut:CH4?
```



Arguments: `FALL` specifies falling edge.

`RISe` specifies rising edge.

Examples: `TRIGGER:MAIN:LOGIC:STATE:INPUT:CH4 RISE`
specifies that the main logic trigger input for channel 4 is the rising edge when the logic class is set to `STATE`.

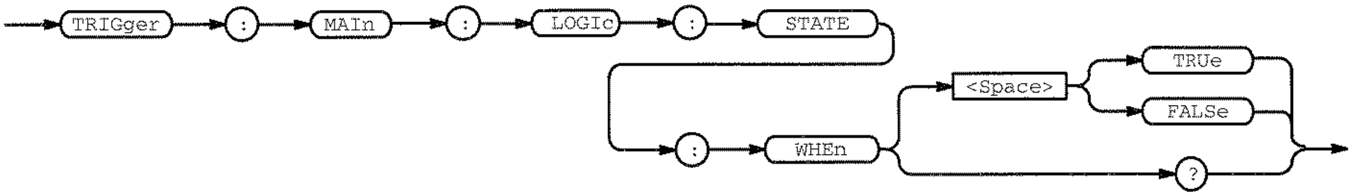
TRIGger:MAIn:LOGIc:STATE:WHEn
TDS 520/540/620/640 Only

Sets or queries the main logic state trigger.

Group: Trigger

Syntax:

```
TRIGger:MAIn:LOGIc:STATE:WHEn { TRUe | FALSe }
TRIGger:MAIn:LOGIc:STATE:WHEn?
```



Arguments: `TRUe` specifies the trigger to occur when the fourth channel's condition is met and the pattern of the first three channels are at the desired states.

`FALSe`

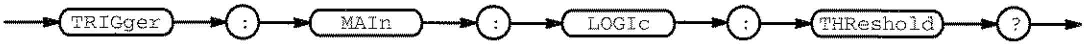
TRIGger:MAIn:LOGIc:THReshold? (Query Only)

TDS 520/540/620/640 Only

Returns the main logic trigger threshold voltage for all channels.

Group: Trigger

Syntax: `TRIGger:MAIn:LOGIc:THReshold?`



Examples: `TRIGGER:MAIN:LOGIC:THRESHOLD?`
might return `:TRIGGER:MAIN:LOGIC:THRESHOLD:CH1 0;CH2 0; CH3 0;CH4 0`

TRIGger:MAIn:LOGIc:THReshold:CH<x>

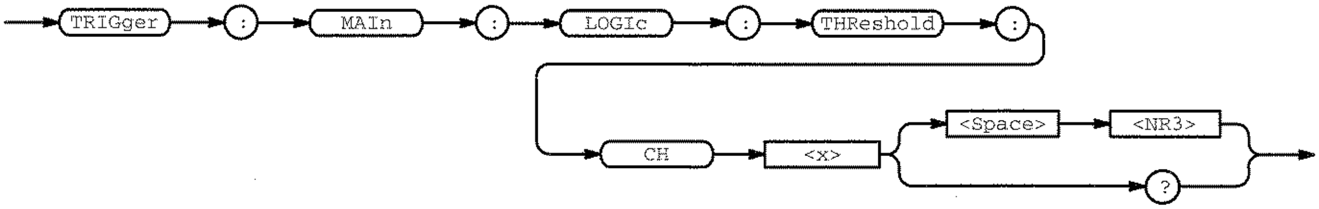
TDS 520/540/620/640 Only

Sets or queries the main logic trigger threshold voltage for the channel specified by <x>. This is equivalent to setting the thresholds in the Logic State Threshold and Logic Pattern Threshold side menus.

Group: Trigger

Syntax: `TRIGger:MAIn:LOGIc:THReshold:CH<x> <NR3>`

`TRIGger:MAIn:LOGIc:THReshold:CH<x>?`



Arguments: <NR3> specifies the threshold voltage.

Examples: `TRIGGER:MAIN:LOGIC:THRESHOLD:CH1 .5`
sets the main logic trigger threshold for channel 1 to .5 volts.

TRIGger:MAIn:LOGIc:WHEn

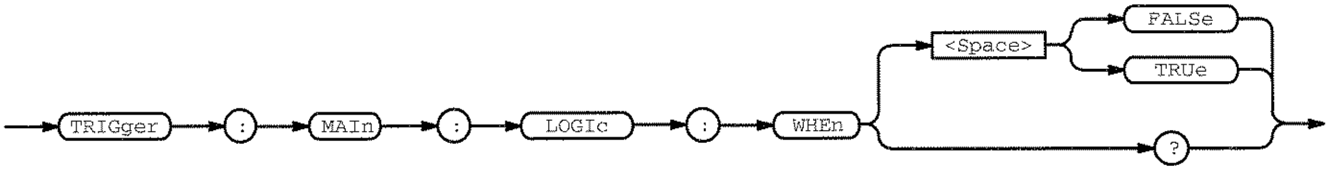
TDS 520/540/620/640 Only

Specifies whether the main logic trigger occurs when the specified state goes true or false when TRIGger:MAIn:LOGIc:CLAss is set to PATtern. This is equivalent to setting the selecting **Trigger When** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:LOGIc:WHEn { FALSe | TRUe }
TRIGger:MAIn:LOGIc:WHEn?
```



Examples:

```
TRIGGER:MAIN:LOGIC:WHEN TRUE
```

specifies that the main logic trigger when the logic pattern is true.

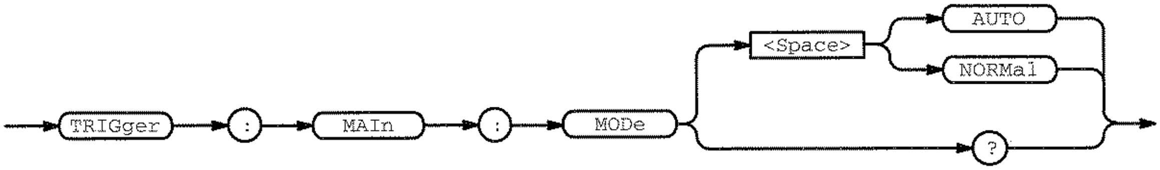
TRIGger:MAIn:MODe

Sets or queries the main trigger mode. This command is equivalent to selecting **Mode & Holdoff** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:MODe { AUTO | NORMal }
TRIGger:MAIn:MODe?
```



Arguments: AUTO generates a trigger if a trigger isn't 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.

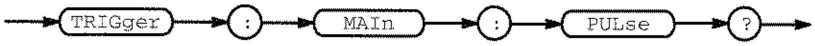
TRIGger:MAIn:PULse? (Query Only)

TDS 520/540/620/640 Only

Returns the main pulse trigger parameters.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse?`



Examples: `TRIGGER:MAIN:PULSE?`

might return `:TRIGGER:MAIN:PULSE:CLASS GLITCH;SOURCE CH1; GLITCH:WIDTH 2.0E-9;FILTER ACCEPT;POLARITY POSITIVE; :TRIGGER:MAIN:PULSE:RUNT:POLARITY POSITIVE;THRESHOLD:HIGH 2.00E+0;LOW 800.0E-3; :TRIGGER:MAIN:PULSE:WIDTH:LOWLIMIT 2.0E-9;HIGHLIMIT 2.0E-9;WHEN WITHIN;POLARITY POSITIVE` as the current main pulse trigger parameters.

TRIGger:MAIn:PULse:CLAss

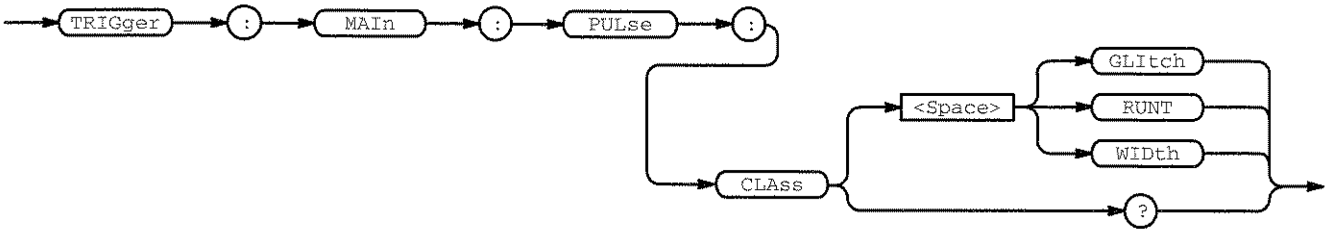
TDS 520/540/620/640 Only

Sets or queries the type of pulse to trigger on. This command is equivalent to selecting **Class** in the Trigger menu.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:CLAss { GLItch | RUNT | WIDth }`

`TRIGger:MAIn:PULse:CLAss?`



Arguments: `GLItch` triggers when a pulse is found that is of the specified polarity and width. These are set with the commands TRIGger:MAIn:PULse:GLItch:POLarity and TRIGger:MAIn:PULse:GLItch:WIDth.

`RUNT` triggers when a pulse crosses the first preset voltage threshold but doesn't cross the second preset threshold before recrossing the first. The thresholds are set with the TRIGger:MAIn:PULse:RUNT:THReshold:LOW and TRIGger:MAIn:PULse:RUNT:THReshold:HIGH commands. The crossing can be either positive or negative as specified by TRIGger:MAIn:PULse:RUNT:POLarity.

`WIDth` triggers when a pulse is found that has the specified polarity and is either inside or outside the limits as specified by TRIGger:MAIn:PULse:WIDth:LOWLimit and TRIGger:MAIn:PULse:WIDth:HIGHLimit. The polarity is selected using the TRIGger:MAIn:PULse:WIDth:POLarity command.

Examples: `TRIGGER:MAIN:PULSE:CLASS WIDTH`
specifies a width pulse for the main trigger.

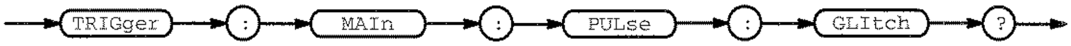
TRIGger:MAIn:PULse:GLItch? (Query Only)

TDS 520/540/620/640 Only

Returns the current main glitch pulse trigger parameters.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:GLItch?`



Examples: `TRIGGER:MAIN:PULSE:GLITCH?`
might return `:TRIGGER:MAIN:PULSE:CLASS GLITCH;SOURCE CH1; GLITCH:WIDTH 2.0E-9;FILTER ACCEPT;POLARITY POSITIVE.`

TRIGger:MAIn:PULse:GLItch:FILTer

TDS 520/540/620/640 Only

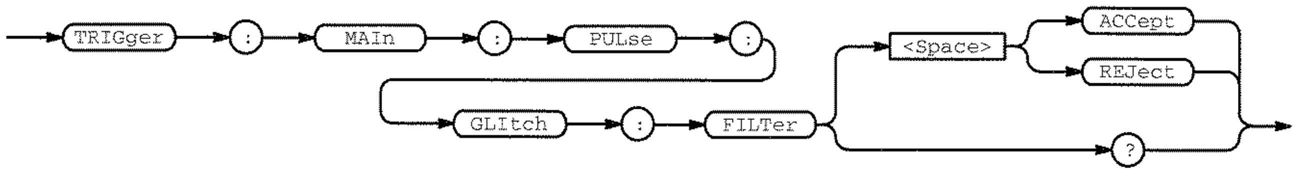
Controls glitch detection. This command is equivalent to selecting **Filter** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:PULse:GLItch:FILTer { ACCept | REJect }

TRIGger:MAIn:PULse:GLItch:FILTer?
```



Arguments: `ACCept` specifies that the digitizing oscilloscope will trigger only on pulses that are narrower than the specified width when the main trigger type is set to pulse glitch. The width is specified using TRIGger:MAIn:PULse:GLItch:WIDth command.

`REJect` specifies that the digitizing oscilloscope will trigger only on pulses that are wider than the specified width when the main trigger type is set to pulse glitch. The width is specified using TRIGger:MAIn:PULse:GLItch:WIDth command.

Examples: `TRIGGER:MAIN:PULSE:GLITCH:FILTER?`
returns either `ACCept` or `REJect`, indicating whether glitches are filtered.

TRIGger:MAIn:PULse:GLItch:POLarity
TDS 520/540/620/640 Only

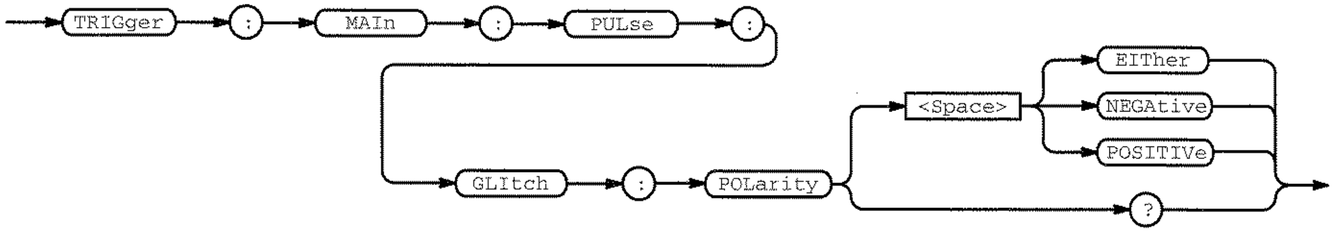
Sets or queries the polarity for the main pulse glitch trigger. This command is equivalent to selecting **Polarity & Width** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:PULse:GLItch:POLarity { EITher | NEGAtive |
    POSITIVe }
```

```
TRIGger:MAIn:PULse:GLItch:POLarity?
```



Examples: `TRIGGER:MAIN:PULSE:GLITCH:POLARITY EITHER`
specifies that the polarity of the glitch can be either positive or negative.

TRIGger:MAIn:PULse:GLItch:WIDth

TDS 520/540/620/640 Only

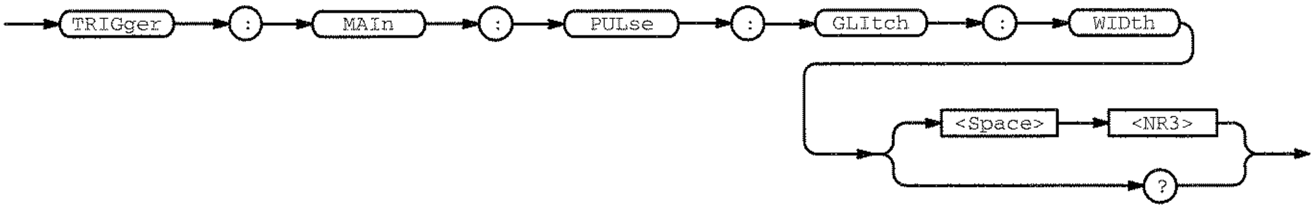
Sets or queries the width for the main pulse glitch trigger. This command is equivalent to selecting **Polarity & Width** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:PULse:GLItch:WIDth <NR3>

TRIGger:MAIn:PULse:GLItch:WIDth?
```



Arguments: `<NR3>` is the width of the glitch, in seconds.

Examples: `TRIGGER:MAIN:PULSE:GLITCH:WIDTH 15E-6`
sets the width of the glitch to 15 μs.

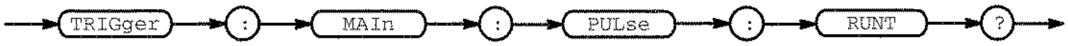
TRIGger:MAIn:PULse:RUNT? (Query Only)

TDS 520/540/620/640 Only

Returns the current parameters for the main pulse runt trigger.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:RUNT?`



Examples: `TRIGGER:MAIN:PULSE:RUNT?`
might return `:TRIGGER:MAIN:PULSE:RUNT:POLARITY POSITIVE;THRESHOLD:HIGH 2.00E+0;LOW 800.0E-3.`

TRIGger:MAIn:PULse:RUNT:POLarity
TDS 520/540/620/640 Only

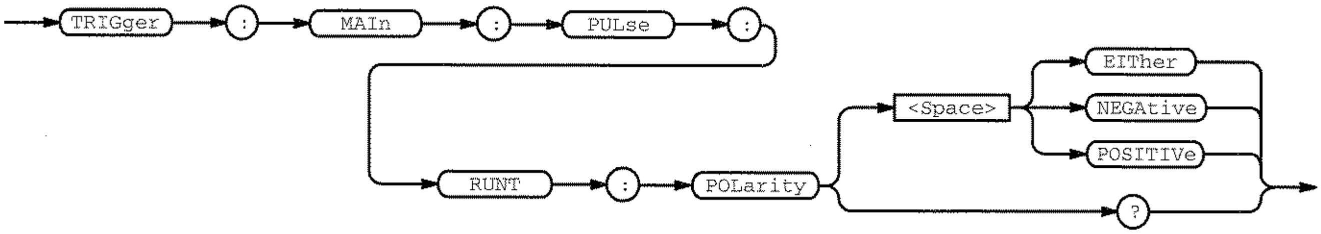
Sets or queries the polarity for the main pulse runt trigger. This command is equivalent to selecting **Polarity** in the Trigger menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:PULse:RUNT:POLarity { EITher | NEGAtive |
    POSITIVe }
```

```
TRIGger:MAIn:PULse:RUNT:POLarity?
```



Arguments: NEGAtive indicates that the falling edge crosses the high threshold and the rising edge recrosses the high threshold without either edge ever crossing the low threshold.

POSITIVe indicates that the rising edge crosses the low threshold and the falling edge recrosses the low threshold without either edge ever crossing the high threshold.

EITher indicates either NEGAtive or POSITIVe polarity.

Examples: `TRIGGER:MAIN:PULSE:RUNT:POLARITY NEGATIVE`
specifies that the polarity of the main pulse runt trigger is negative.

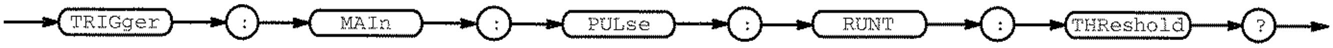
TRIGger:MAIn:PULse:RUNT:THReshold? (Query Only)
TDS 520/540/620/640 Only

Returns the upper and lower thresholds for the main pulse runt trigger.

Group: Trigger

Syntax:

```
TRIGger:MAIn:PULse:RUNT:THReshold?
```



Examples: `TRIGGER:MAIN:PULSE:RUNT:THRESHOLD?`
might return `:TRIGGER:MAIN:PULSE:RUNT:THRESHOLD:HIGH 2.00E+0;LOW 800.0E-3.`

TRIGger:MAIn:PULse:RUNT:THReshold:HIGH
TDS 520/540/620/640 Only

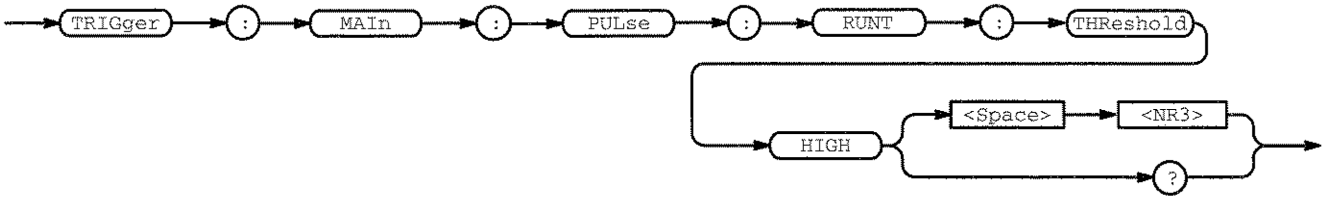
Sets or queries the upper limit for the main pulse runt trigger. This command is equivalent to setting the threshold in the Pulse Runt Threshold side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:PULse:RUNT:THReshold:HIGH <NR3>

TRIGger:MAIn:PULse:RUNT:THReshold:HIGH?
```



Arguments: `<NR3>` is the threshold, in volts.

Examples: `TRIGGER:MAIN:PULSE:RUNT:THRESHOLD:HIGH 120E-3`
sets the upper limit of the pulse runt trigger to 120 mV.

TRIGger:MAIn:PULse:RUNT:THReshold:LOW
TDS 520/540/620/640 Only

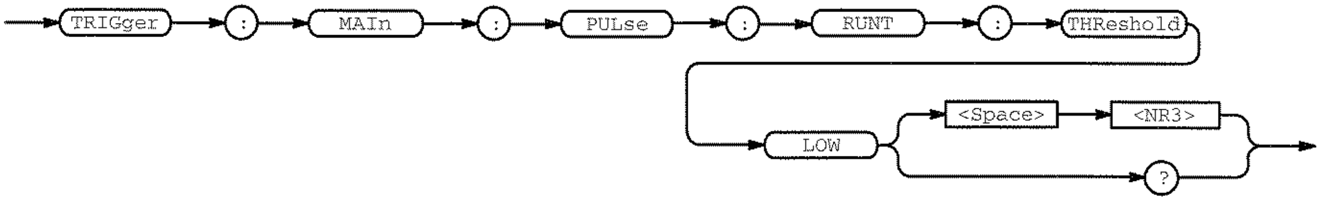
Sets or queries the lower limit for the main pulse runt trigger. This command is equivalent to setting the threshold in the Pulse Runt Threshold side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:PULse:RUNT:THReshold:LOW <NR3>

TRIGger:MAIn:PULse:RUNT:THReshold:LOW?
```



Arguments: <NR3> is the threshold, in volts.

Examples: `TRIGGER:MAIN:PULSE:RUNT:THRESHOLD:LOW 50E-3`
sets the lower limit of the pulse runt trigger to 50 mV.

TRIGger:MAIn:PULse:SOUrce

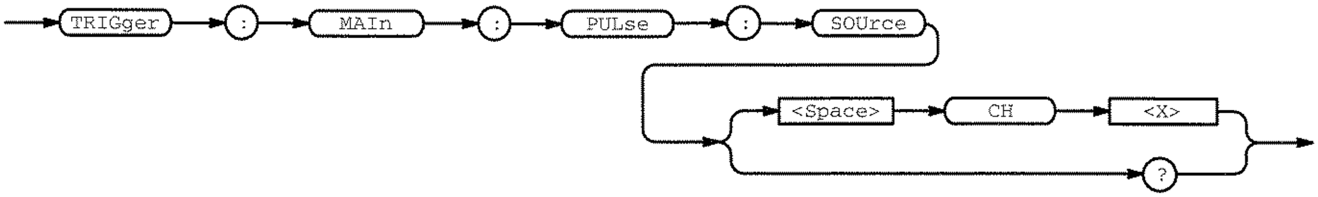
TDS 520/540/620/640 Only

Sets or queries the source for the main pulse trigger. This is equivalent to selecting the source in the Pulse Runt Source side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:SOUrce CH<x>`

`TRIGger:MAIn:PULse:SOUrce?`



Arguments: CH<x> specifies one of the input channels.

Examples: `TRIGGER:MAIN:PULSE:SOURCE CH2`
selects Channel 2 as the source for the main pulse trigger.

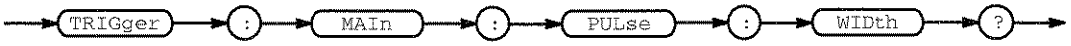
TRIGger:MAIn:PULse:WIDth? (Query Only)

TDS 520/540/620/640 Only

Returns the width parameters for the main pulse width trigger.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:WIDth?`



Examples: `TRIGGER:MAIN:PULSE:WIDTH?`
might return `:TRIGGER:MAIN:PULSE:WIDTH:LOWLIMIT 2.0E-9;HIGHLIMIT 2.0E-9;WHEN WITHIN;POLARITY POSITIVE` as the current main pulse trigger parameters.

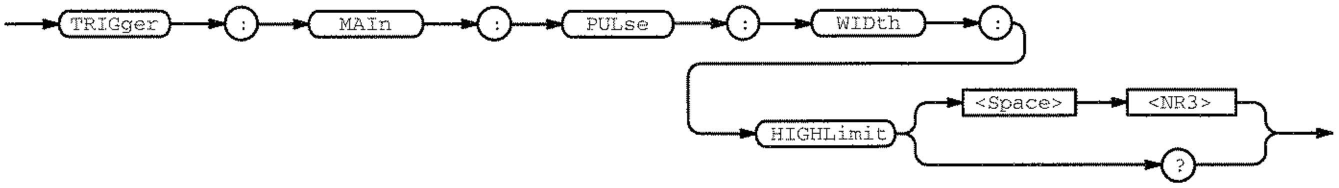
TRIGger:MAIn:PULse:WIDth:HIGHLimit
TDS 520/540/620/640 Only

Sets or queries the upper limit for the main pulse width trigger. This is equivalent to setting the **Upper Limit** in the Pulse Width Trig When side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:WIDth:HIGHLimit <NR3>`

`TRIGger:MAIn:PULse:WIDth:HIGHLimit?`



Arguments: `<NR3>` is the upper limit, in seconds.

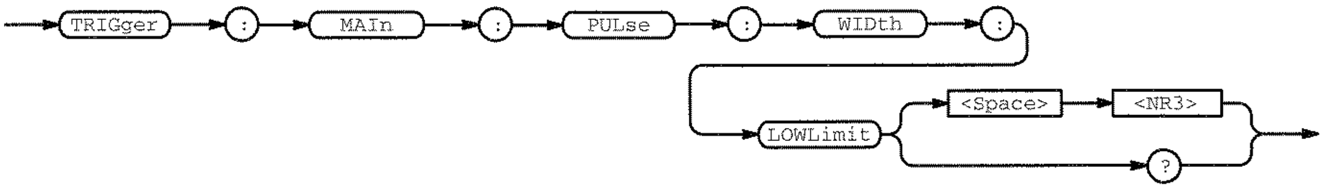
TRIGger:MAIn:PULse:WIDth:LOWLimit
TDS 520/540/620/640 Only

Sets or queries the lower limit for the main pulse width trigger. This is equivalent to setting the **Lower Limit** in the Pulse Width Trig When side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:WIDth:LOWLimit <NR3>`

`TRIGger:MAIn:PULse:WIDth:LOWLimit?`



Arguments: <NR3> is the lower limit, in seconds.

TRIGger:MAIn:PULse:WIDth:POLarity

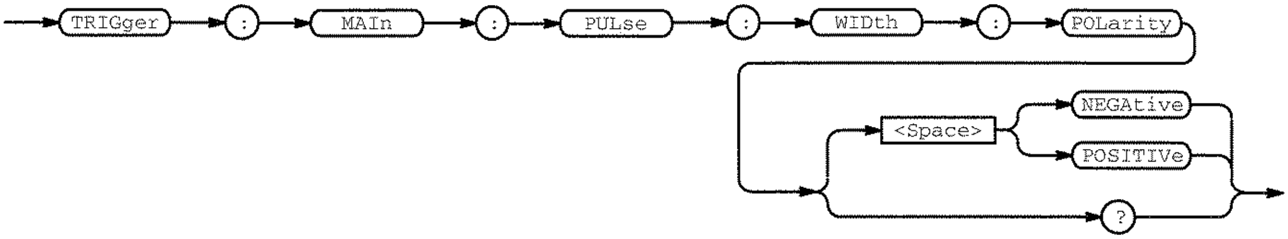
TDS 520/540/620/640 Only

Sets or queries the polarity for the main pulse width trigger. This is equivalent to selecting the polarity in the Pulse Width Polarity side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:WIDth:POLarity { NEGAtive | POSITIVe }`

`TRIGger:MAIn:PULse:WIDth:POLarity?`



Arguments: `NEGAtive` specifies a negative pulse.

`POSITIVe` specifies a positive pulse.

TRIGger:MAIn:PULse:WIDth:WHEn

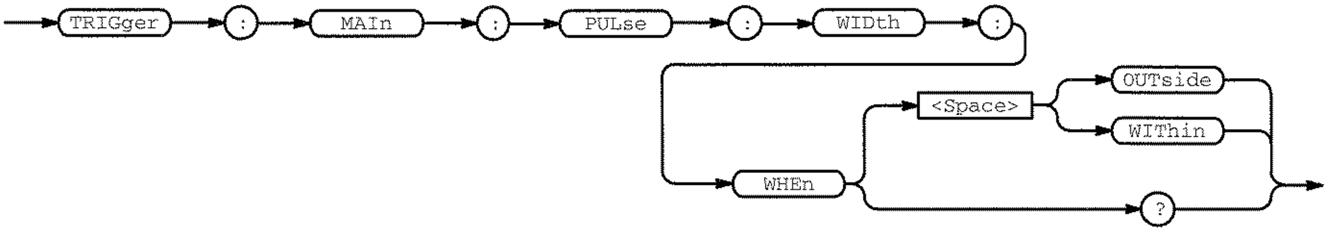
TDS 520/540/620/640 Only

Selects the condition when the trigger occurs. This is equivalent to selecting the condition in the Pulse Width Trig When side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:PULse:WIDth:WHEn { OUTside | WIThin }`

`TRIGger:MAIn:PULse:WIDth:WHEn?`



Arguments: `OUTside` specifies a trigger when the duration of the pulse is greater than the high limit or less than the low limit specified. The high and low limits are specified with the TRIGger:MAIn:PULse:WIDth:HIGHLimit and TRIGger:MAIn:PULse:WIDth:LOWLimit commands respectively.

`WIThin` specifies a trigger when the duration of the pulse is within the high and low limits. The high and low limits are specified with the TRIGger:MAIn:PULse:WIDth:HIGHLimit and TRIGger:MAIn:PULse:WIDth:LOWLimit commands respectively.

Examples: `TRIGGER:MAIN:PULSE:WIDTH:WHEN?`
returns either `OUTSIDE` or `WITHIN`, indicating the conditions for generating a pulse trigger.

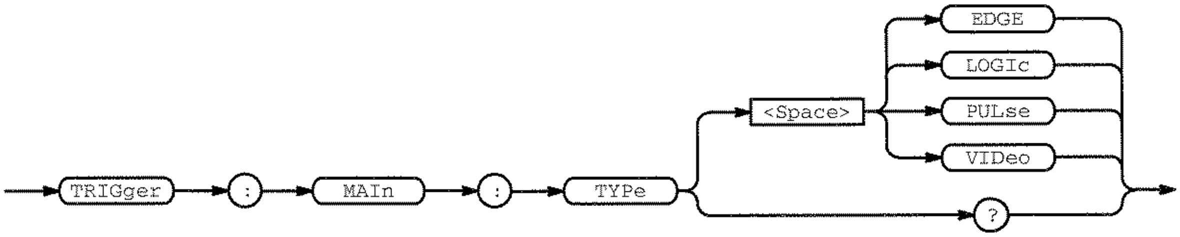
TRIGger:MAIn:TYPe

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 | LOGIc | PULse | VIDeo }`
(Note: only the TDS 520/540/620/640 use the LOGIc and PULse arguments. Only the TDS 420 and 460 with option 5 use the VIDeo argument.)

`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 and is controlled by the TRIGger:MAIn:EDGE commands.

`LOGIc` (TDS 520/540/620/640 only) specifies that a trigger occurs when specified conditions are met and is controlled by the TRIGger:MAIn:LOGIc commands.

`PULse` (TDS 520/540/620/640 only) specifies that a trigger occurs when a specified pulse is found and is controlled by the TRIGger:MAIn:PULse commands.

`VIDeo` (TDS 420/460 option 5 only) specifies that a trigger occurs when a specified signal is found and is controlled by the TRIGger:MAIn:VIDeo commands.

Examples: `TRIGGER:MAIN:TYPE?`
might return `PULSE` indicating that the main trigger type is a pulse trigger.

TRIGger:MAIn:VIDeo:BY
TDS 420/460 Option 5 Only

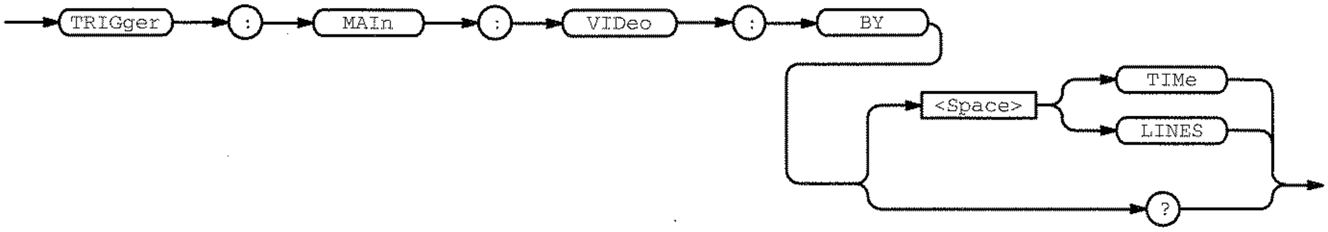
Sets or queries the video trigger delay mode. This is equivalent to using the Video **TV Delay Mode** side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:VIDeo:BY { TIMe | LINES }

TRIGger:MAIn:VIDeo:BY?
```



Arguments: TIMe specifies a delay by time.

LINES specifies a delay by a number of video lines.

Examples: TRIGGER:MAIN:VIDEO:BY TIME
specifies a delay by time.

TRIGger:MAIn:VIDeo:FIELD
TDS 420/460 Option 5 Only

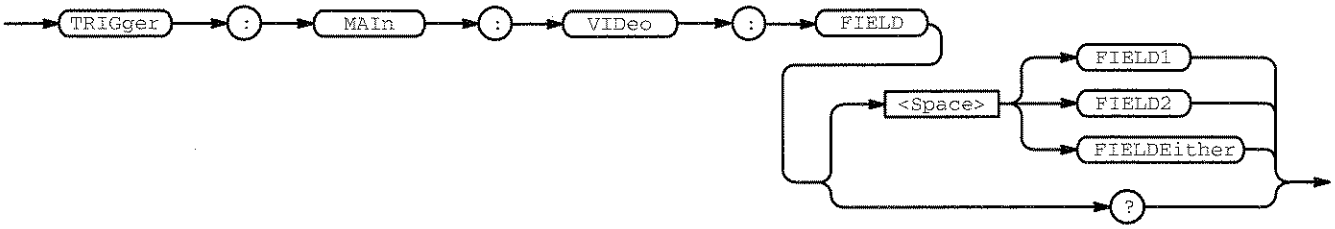
Sets or queries the field the video trigger acts on. This is equivalent to using the Video **Scan Rate and Interlace** side menu when **Class** is NOT set to **Custom**.

Group: Trigger

Syntax:

```
TRIGger:MAIn:VIDeo:FIELD { FIELD1 | FIELD2 | FIELDEither }

TRIGger:MAIn:VIDeo:FIELD?
```



Arguments: FIELD1 specifies interlaced video field 1.

FIELD2 specifies interlaced video field 2.

FIELDEither specifies alternating both video field 1 and video field 2.

Examples: TRIGGER:MAIN:VIDEO:SCAN FIELD1
selects field 1.

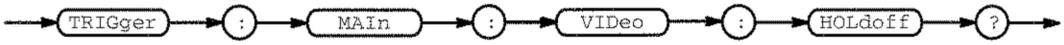
TRIGger:MAIn:VIDeo:HOLdoff? (Query Only)

TDS 420/460 Option 5 Only

Returns the video trigger holdoff value.

Group: Trigger

Syntax: `TRIGger:MAIn:VIDeo:HOLdoff?`



Examples: `TRIGGER:MAIN:VIDEO:HOLDOFF?`
might return `:TRIGGER:MAIN:VIDEO:HOLDOFF:VALUE 0.`

TRIGger:MAIn:VIDeo:HOLdoff:VALue

TDS 420/460 Option 5 Only

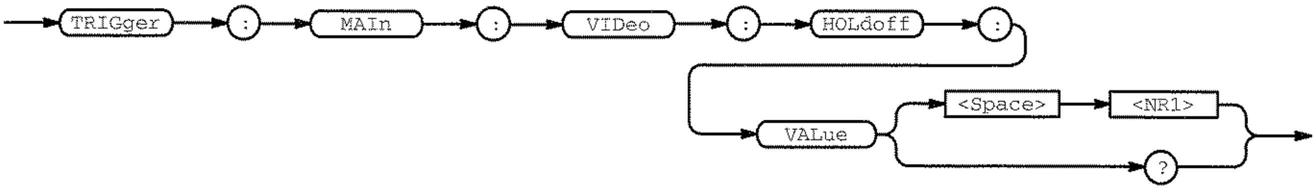
Sets or queries the video trigger holdoff value. This is equivalent to setting **Holdoff** in the video trigger menu's Mode & Holdoff side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:VIDeo:HOLdoff:VALue <NR1>

TRIGger:MAIn:VIDeo:HOLdoff:VALue?
```



Arguments: `<NR1>` is from 0 to 100, and is a percent of the holdoff range.

Examples: `TRIGGER:MAIN:HOLDOFF:VALUE 10`
set the holdoff value to be 10% of the holdoff range.

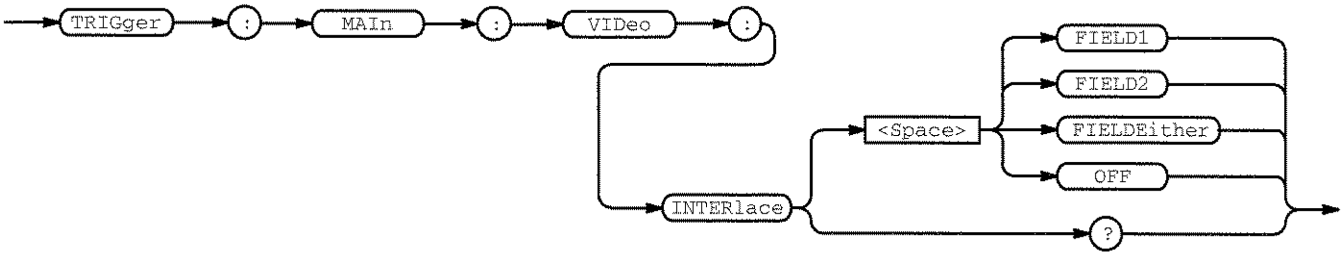
TRIGger:MAIn:VIDeo:INTERLAce
TDS 420/460 Option 5 Only

Sets or queries the video trigger interlace format. This is equivalent to setting **Interlace** in the video trigger menu's **Scan Rate and Interlace** main menu when **Class** is set to **Custom**.

Group: Trigger

Syntax: `TRIGger:MAIn:VIDeo:INTERLAce { FIELD1 | FIELD2 | FIELDEither | OFF }`

`TRIGger:MAIN:VIDeo:INTERLAce?`



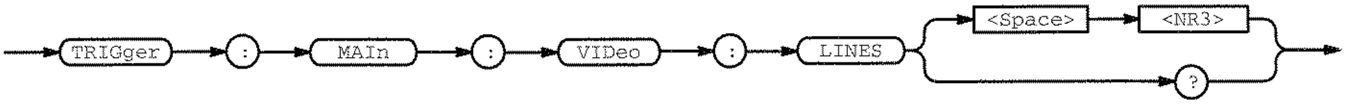
TRIGger:MAIn:VIDeo:LINES
TDS 420/460 Option 5 Only

Sets or queries the video trigger delay in terms of a number of lines. This is equivalent to entering data in the **Delay by Lines** item in the Video **TV Delay Mode** side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:VIDeo:LINES { NR3 }`

`TRIGger:MAIn:VIDeo:LINES?`



Arguments: <nr3> specifies a number of lines to delay by.

Examples: TRIGGER:MAIN:VIDEO:LINES 5
selects 5 lines for the desired delay period.

TRIGger:MAIn:VIDeo:SCAN

TDS 420/460 Option 5 Only

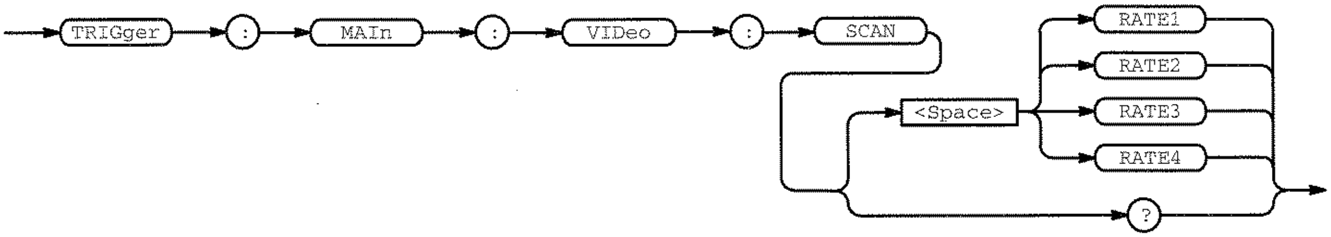
Sets or queries the video trigger scan parameters. This is equivalent to using the Video **Scan Parameters** side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:VIDeo:SCAN { RATE1 | RATE2 | RATE3 | RATE4 }

TRIGger:MAIn:VIDeo:SCAN?
```



Arguments: RATE1 specifies a 15 to 20 kHz video line rate.

RATE2 specifies a 20 to 25 kHz video line rate.

RATE3 specifies a 25 to 35 kHz video line rate.

RATE4 specifies a 35 to 64 kHz video line rate.

Examples: TRIGGER:MAIN:VIDEO:SCAN RATE1
selects rate 1.

TRIGger:MAIn:VIDeo:SOUrce

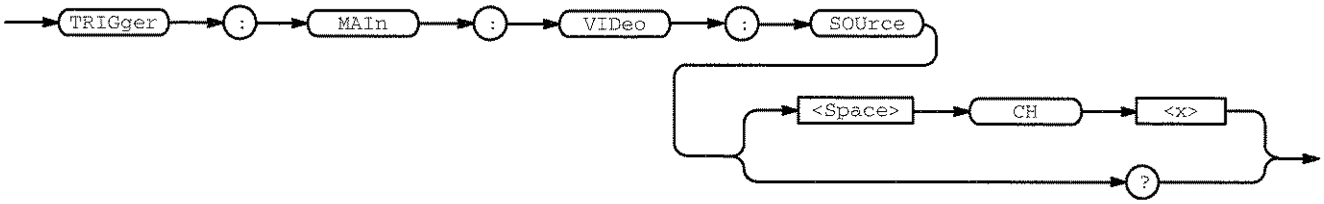
TDS 420/460 Option 5 Only

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, CH2, CH3, or CH4).

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

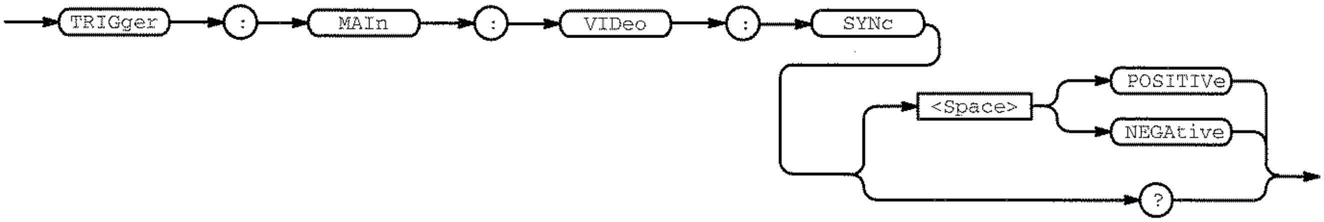
TRIGger:MAIn:VIDeo:SYNc
TDS 420/460 Option 5 Only

Sets or queries the video trigger sync polarity This is equivalent to selecting the source in the Video **Sync Polarity** side menu.

Group: Trigger

Syntax: `TRIGger:MAIn:VIDeo:SYNc { POSITIVe | NEGAtive }`

`TRIGger:MAIn:VIDeo:SYNc?`



Arguments: POSITIVe specifies a positive going voltage.

NEGAtive specifies a negative going voltage.

Examples: TRIGGER:MAIN:VIDEO:SYNC POSITIVE
selects a positive going voltage for the desired synchronization pulse.

TRIGger:MAIn:VIDeo:SYStem
TDS 420/460 Option 5 Only

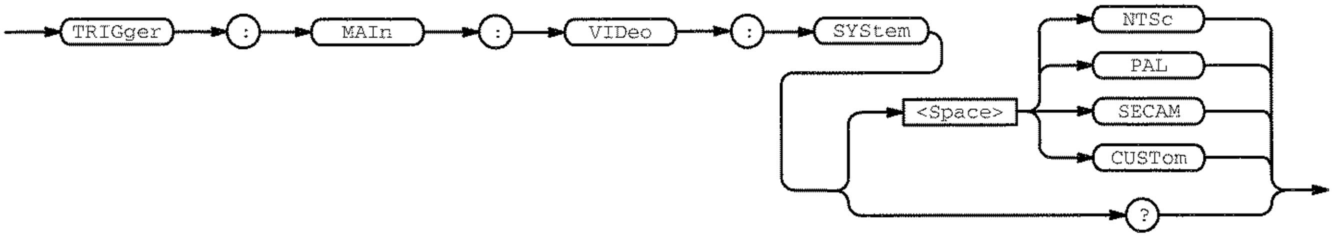
Sets or queries the video trigger class. This is equivalent to selecting the class in the Video menu's **Video Class** side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:VIDeo:SYStem { NTSc | PAL | SECAM | CUSTom }

TRIGger:MAIn:VIDeo:SYStem?
```



Arguments: NTSc selects a condition that adheres to the National Television System Committee standards. Specifically, it assumes a line rate of 525 lines per frame and a frame rate of 30 Hz.

PAL selects a condition that adheres to the Phase Alternate Line standard. Specifically, it assumes a line rate of 625 lines per frame and a frame rate of 25 Hz.

SECAM selects a condition that adheres to the SECAM standard.

CUSTom selects a condition that adheres to the frequency range of the video signal as you have defined them from the available ranges.

Examples:

```
TRIGGER:MAIN:SYSTEM NTSC
```

selects triggering to occur on an NTSC compatible signal.

TRIGger:MAIn:VIDeo:TIMe

TDS 420/460 Option 5 Only

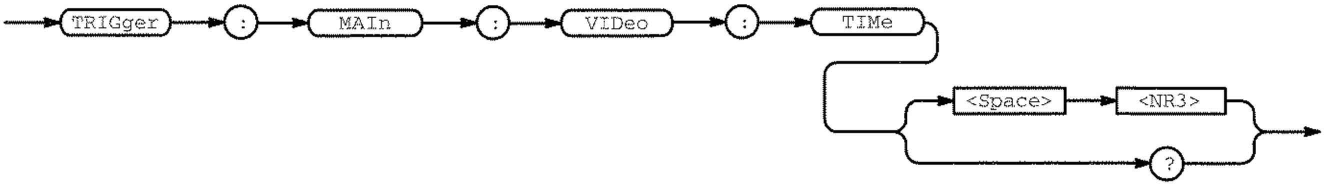
Sets or queries the video trigger delay time. This is equivalent to entering the time in the **Delay by Time** item of the Video **TV Delay Mode** side menu.

Group: Trigger

Syntax:

```
TRIGger:MAIn:VIDeo:TIMe { <NR3> }

TRIGger:MAIn:VIDeo:TIMe?
```



Arguments: <NR3> specifies a delay time.

Examples: TRIGGER:MAIN:VIDEO:TIME 5E−6
selects 5 μs for the desired delay time.

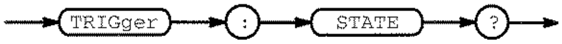
TRIGger:STATE? (Query Only)

Returns the current state of the triggering system.

Group: Trigger

Syntax:

```
TRIGger:STATE?
```



Returns: `ARMed` indicates that the instrument is acquiring pretrigger information. All triggers are ignored when TRIGger:STATE is ARMING.

`PARTial` indicates that the main trigger has occurred and the digitizing oscilloscope is waiting for trigger(s) for the delay by events.

`READy` indicates that all pretrigger information has been acquired and the instrument is ready to accept a trigger.

`TRIGger` indicates that the instrument has seen a trigger and is acquiring the posttrigger information.

`SAVe` indicates that the instrument is in save mode and is not acquiring data.

`AUTO` indicates that the instrument is in auto mode and acquires data even in the absence of a trigger.

Examples: `TRIGGER:STATE?`
might return `ARMED`, indicating that pretrigger data is being acquired.

*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`

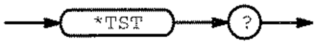
Examples: `*TRG`
immediately executes all commands that have been defined by *DDT.

*TST? (Query Only)

(Self-Test) Tests the GPIB interface and returns a 0.

Group: Miscellaneous

Syntax: `*TST?`



Returns: `<NR1>` and is always 0.

UNLOCk (No Query Form)

Unlocks the front panel. This command is equivalent to LOCk NONe.

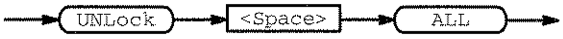
NOTE

If the digitizing 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

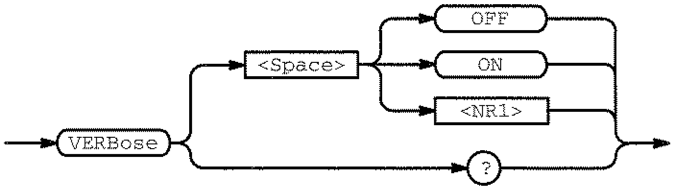
Sets and queries the Verbose State that controls the length of keywords on query responses. Keywords can be both headers and arguments. This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk).

Group: Miscellaneous

Related Commands: HEADer, *LRN?, SET?

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

`VERBose?`



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.

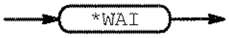
*WAI (No Query Form)

(Wait) Prevents the digitizing oscilloscope from executing further commands or queries until all pending operations finish. This command allows you to synchronize the operation of the digitizing oscilloscope with your application program. Synchronization methods are described on page NO TAG.

Group: Status and Error

Related Commands: BUSY?, *OPC

Syntax: *WAI



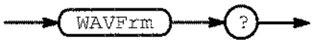
WAVFrm? (Query Only)

Returns WFMPre? and CURVe? data for the waveform or waveforms as specified by the DATa:SOUrce command. This command is equivalent to sending WFMPre?; CURVe?.

Group: Waveform

Related Commands: CURVe?, DATa:SOUrce, WFMPre?

Syntax: WAVFrm?



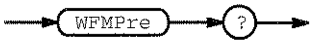
WFMPre? (Query Only)

Returns the waveform formatting data for the first ordered waveform as specified by the DATa:SOUrce command. All channel and math waveforms must be displayed.

Group: Waveform

Related Commands: WAVFrm?

Syntax: `WFMPre?`



Returns: The format of the response is:

```
BYT_Nr <NR1>;BIT_Nr <NR1>;ENCdg { ASC | BIN };
BN_Fmt { RI | RP };BYT_Or { LSB | MSB };
<wfm>:WFID <Qstring>;NR_PT <NR1>;PT_FMT { ENV | Y };
XUNit <QString>;XINcr <NR3>;PT_Off <NR1>;YUNit
<QString>;YMUlt <NR3>; YOFf <NR3>;YZEro<NR3>[;<wfm>:
WFID <Qstring>;NR_PT <NR1>;PT_FMT{ ENV | Y };
XUNit<QString>;XINcr <NR3>;PT_Off <NR1>;YUNit
<QString>;
YMUlt <NR3>; YOFf <NR3>;YZEro <NR3>...]
```

WFMPre:BIT_Nr

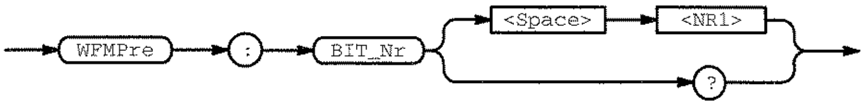
Returns the number of bits per binary waveform point for the first ordered waveform as specified by the DATa:SOUrce command. The WFMPre:BIT_Nr command is ignored on input.

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 first ordered waveform as specified by the DATa:SOUrce command.

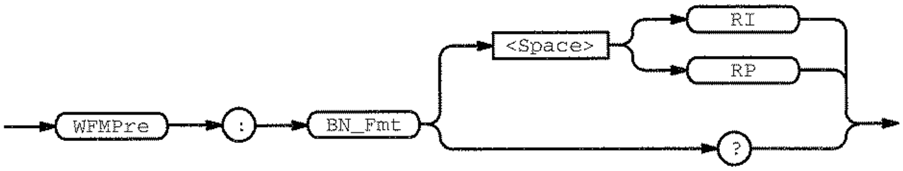
Group: Waveform

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.

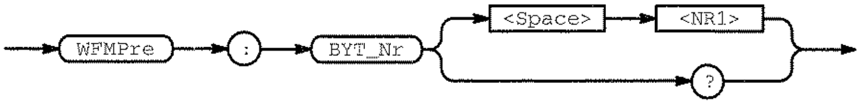
Group: Waveform

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.

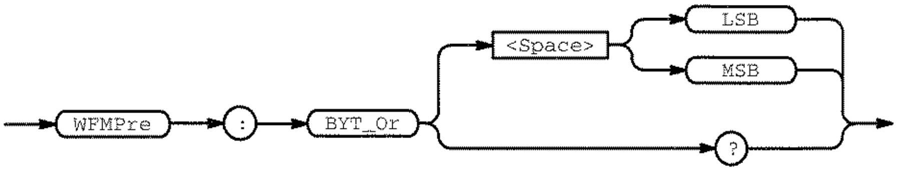
Group: Waveform

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 bye to be transmitted first.

Examples: `WFMPRE:BYT_OR MSB`
specifies that the most significant byte in the waveform data will be 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.

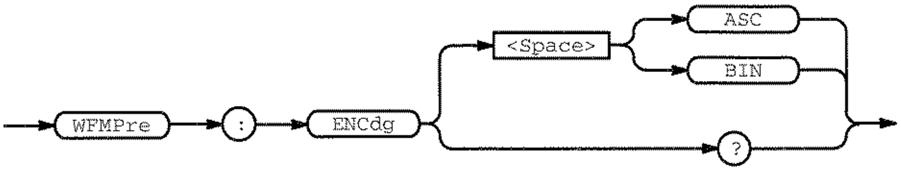
Group: Waveform

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.

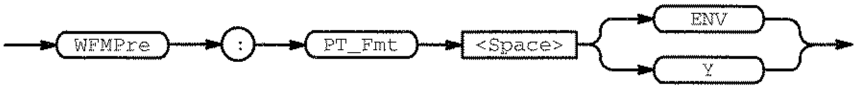
`WFMPRE:ENCDG?`
might return `BIN`, indicating that the waveform data is in binary format.

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:

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

$$Y_{n_{max}} = YZEro + YMUlt\ (y_{n_{max}} - YOFf)$$

$$Y_{n_{min}} = YZEro + YMUlt\ (y_{n_{min}} - YOFf)$$

`Y` specifies a normal waveform where one ASCII or binary data point is transmitted for each point in the waveform record. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

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

$$Y_n = YZEro + YMUlt\ (y_n - YOFf)$$

Examples: `WFMPRE:PT ENV`
sets the waveform data point format to enveloped.

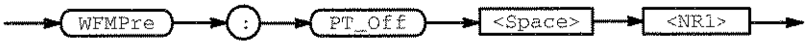
WFMPre:PT_Off (No Query Form)

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>`



Arguments: `<NR1>` = 0 to the recordlength, and is the position of the trigger point relative to DATa:STARt.

Examples: `WFMPRE:PT_OFF 1`

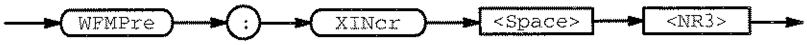
specifies that the trigger point is the first point in the waveform record.

WFMPre:XINcr (No Query Form)

Specifies the horizontal sampling interval for the reference waveform specified by the DATa:DESTination command.

Group: Waveform

Syntax: `WFMPre:XINcr <NR3>`



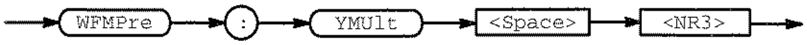
Arguments: `<NR3>` is the sampling interval, in seconds per point.

WFMPre:YMUlt (No Query Form)

Specifies the vertical scale factor for the reference waveform specified by the DATa:DESTination command.

Group: Waveform

Syntax: `WFMPre:YMUlt <NR3>`



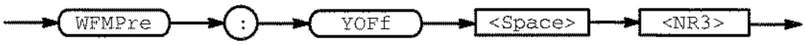
Arguments: `<NR3>` is the vertical scale factor, in YUNits (usually volts) per division.

WFMPre:YOFf (No Query Form)

Specifies the offset of the vertical component for the reference waveform specified by the DATa:DESTination command.

Group: Waveform

Syntax: `WFMPre:YOFf <NR3>`



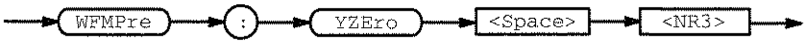
Arguments: `<NR3>` is the vertical offset in digitizing levels.

WFMPre:YZEro (No Query Form)

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

Group: Waveform

Syntax: `WFMPre:YZEro <NR3>`



Arguments: `<NR3>` is of the offset, in YUNits (usually volts).

Table 2-27 lists additional WFMPre commands that are included for compatibility purposes.

NOTE

These commands do not support a query form and all information is ignored.

Table 2-27: Additional WFMPre Commands

Command	Argument	Description
`WFMPre:CRVchk`	`{CHKSM0 | NONe}`	Binary curve error check
`WFMPre:NR_PT`	`<NR1>`	Number of waveform points
`WFMPre:WFId`	`<QString>`	Waveform identifier
`WFMPre:XUNit`	`<QString>`	Horizontal units
`WFMPre:XMUlt`	`<NR3>`	Horizontal (X-axis) scale factor
`WFMPre:XOFf`	`<NR3>`	Horizontal (X-axis) offset
`WFMPre:XZEro`	`<NR3>`	Horizontal (X-axis) origin offset
`WFMPre:YUNit`	`<QString>`	Vertical units
`WFMPre:ZMUlt`	`<NR3>`	Z-axis scale factor
`WFMPre:ZOFf`	`<NR3>`	Z-axis offset
`WFMPre:ZUNit`	`<QString>`	Z-axis units
`WFMPre:ZZEro`	`<NR3>`	Z-axis origin offset

NOTE

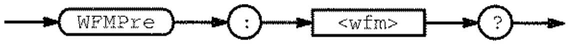
When returning WFMPRE:<wfm> information from the oscilloscope, <wfm> specifies the waveform source (CH<x>, MATH<x>, or REF<x>). The source must also be set using the DAta:SOUrce command. When sending WFMPRE:<wfm> information to the scope, the <wfm> specification is ignored and the reference location specified by DATa:DESTination is used instead.

WFMPre:<wfm>? (Query Only)

Returns the waveform formatting data for first ordered waveform as specified by the DATa:SOUrce command. Channel and math waveforms must be displayed before they can be queried. Querying an invalid reference waveform generates an execution error.

Group: Waveform

Syntax: `WFMPre:<wfm>?`



Returns: The format of the response is:

```
<wfm>:WFID <Qstring>;NR_PT <NR1>;PT_FMT { ENV | Y };
XUNit <QString>;XINcr <NR3>;PT_Off <NR1>;YUNit
<QString>;YMUlt <NR3>;YOFf <NR3>;YZEro <NR3>
[;<wfm>:WFID <Qstring>;NR_PT <NR1>;
PT_FMT { ENV | Y };XUNit <QString>;XINcr <NR3>;
PT_Off <NR1>;YUNit <QString>;YMUlt <NR3>;YOFf <NR3>;
YZEro <NR3>...]
```

WFMPre:<wfm>:NR_Pt

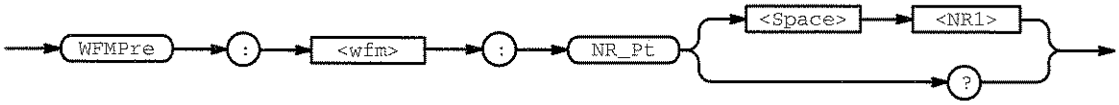
Sets or queries the number of points that are in the transmitted waveform record. This value is ignored on input.

Related Commands: DATa:DESTination

Group: Waveform

Syntax: `WFMPre:<wfm>:NR_Pt <NR1>`

`WFMPre:<wfm>:NR_Pt?`



Arguments: `<NR1>` is the number of data points. If DATa:WIDth is 2 then there are twice as many bytes.

`<NR1>` = 0 means that the waveform record is of an unspecified length.

Examples: `WFMPRE:CH1:NR_Pt?`
might return `5000` 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 first ordered waveform as selected by the DATa:SOUrce command. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

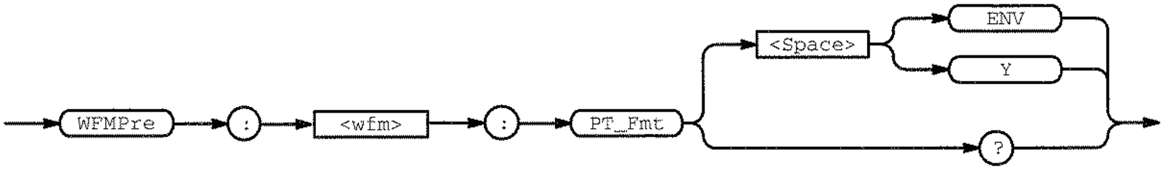
Group: Waveform

Related Commands: DATa:DESTination

Syntax:

```
WFMPre:<wfm>:PT_Fmt { ENV | Y }

WFMPre:<wfm>:PT_Fmt?
```



Arguments: ENV specifies that the waveform is transmitted as maximum and minimum point pairs. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

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

$$Y_{n_{max}} = YZEro + YMUlt\ (y_{n_{max}} - YOFf)$$

$$Y_{n_{min}} = YZEro + YMUlt\ (y_{n_{min}} - YOFf)$$

Y specifies a normal waveform where one ASCII or binary data point is transmitted for each point in the waveform record. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

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

$$Y_n = YZEro + YMUlt\ (y_n - YOFf)$$

Examples:

```
WFMPRE:MATH1:PT_FMT?
```

might return ENV, indicating that the MATH1 waveform data format is enveloped.

WFMPre:<wfm>:PT_Off

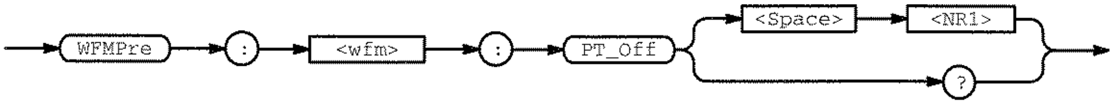
Returns the trigger point within the waveform record. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

Group: Waveform

Syntax:

```
WFMPre:<wfm>:PT_Off <NR1>

WFMPre:<wfm>:PT_Off?
```



Arguments: <NR1> = 0 to the recordlength, and is the position of the trigger point relative to DATa:STARt when queried.

Examples: `WFMPRE:CH1:PT_OFF?`
returns 0 indicating the trigger position within the waveform record.

WFMPre:<wfm>:WFId

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

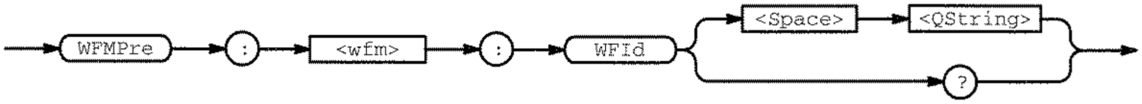
The WFMPre:<wfm>:WFId command is ignored on input.

Group: Waveform

Syntax:

```
WFMPre:<wfm>:WFId <QString>

WFMPre:<wfm>:WFId?
```



Arguments: <QString> is the waveform identifier string.

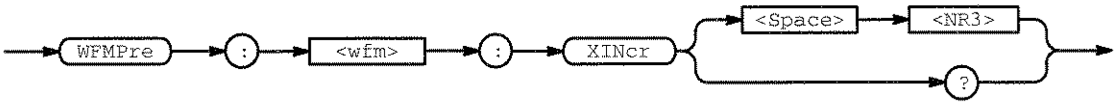
WFMPre:<wfm>:XINcr

Sets or queries the horizontal sampling interval. 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

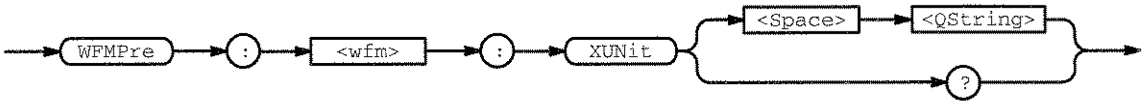
Returns the horizontal (X-axis) units of the waveform data at the time of creation.

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?`
might return `"s"`, indicating that the horizontal units for channel 1 are seconds.

WFMPre:<wfm>:YMUlt

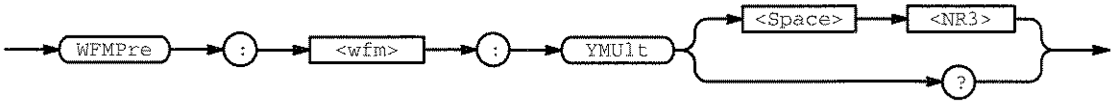
Sets or queries the vertical scale factor, in YUNit per unscaled data point value. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

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

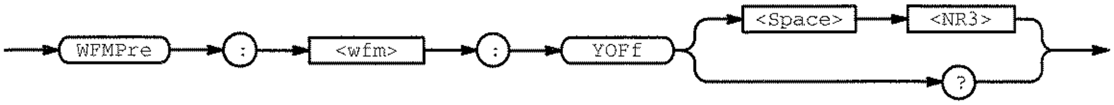
Sets or queries the vertical position of the waveform. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

Group: Waveform

Syntax:

```
WFMPre:<wfm>:YOFf <NR3>

WFMPre:<wfm>:YOFf?
```



Arguments: <NR3> is the position in digitizing levels.

WFMPre:<wfm>:YUNit

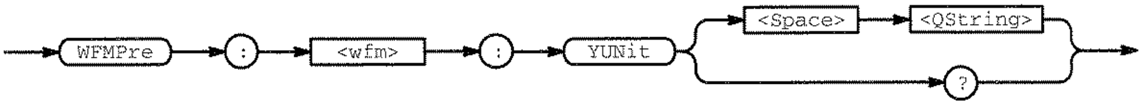
Returns the vertical (Y-axis) units of the waveform data at the time of creation.

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 volts^2, 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.

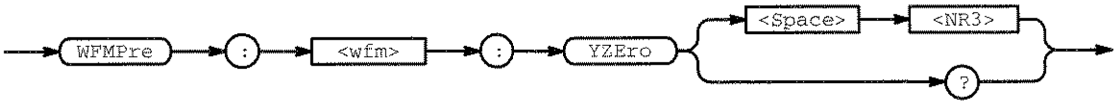
WFMPre:<wfm>:YZEro

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

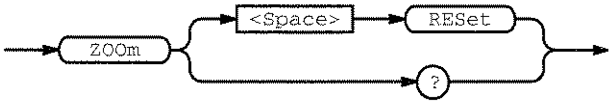
Resets the display to its normal state, and resets all Zoom parameters to their factory default settings. The ZOOm query returns the current vertical and horizontal positioning and scaling of the display. This command is equivalent to selecting **Reset Zoom Factors** in the Zoom menu.

Group: Zoom

Syntax:

```
ZOOm RESet

ZOOm?
```



Arguments: `RESet` sets the horizontal and vertical positions to zero, and the horizontal and vertical scale to one.

Examples: `ZOOM?`
might return `:ZOOM:STATE 0;HORIZONTAL:SCALE 1.00E+0;POSITION 500.0E-3;LOCK LIVE;:ZOOM:VERTICAL:SCALE 1.0E+0;POSITION 0.0E+0.`

ZOOm:HORizontal:LOCk

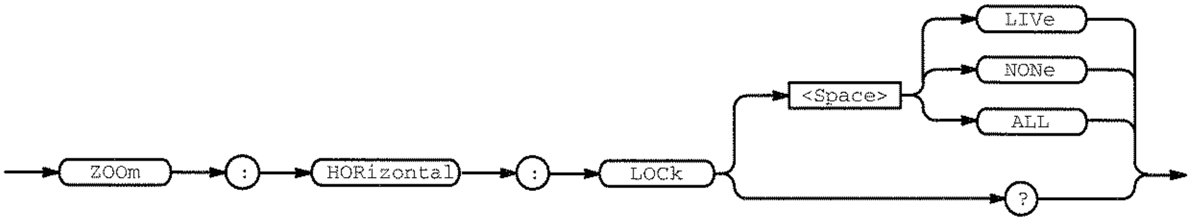
Specifies the waveforms that the horizontal zoom parameters affect. This is equivalent to setting **Horizontal Lock** in the Zoom side menu.

Group: Zoom

Syntax:

```
ZOOm:HORizontal:LOCk { LIVe | NONe | ALL }

ZOOm:HORizontal:LOCk?
```



Arguments: `LIVe` specifies that all live (CH<x>) waveforms will be horizontally positioned and scaled together.

`NONe` specifies that only the selected waveform is positioned and scaled using the horizontal zoom parameters.

`ALL` specifies that all (CH<x>, Ref<x>, Math<x>) waveforms will be horizontally positioned and scaled together.

Examples: `ZOOM:HORIZONTAL:LOCK LIVE`
specifies that all live waveforms are positioned and scaled together.

`ZOOM:HORIZONTAL:LOCK?`
returns either `LOCK` or `NONE`.

ZOOm:HORizontal:POSition

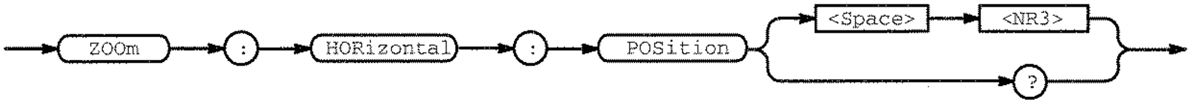
Sets or queries the horizontal position of waveforms. If ZOOm:HORizontal:LOCk is set to LIVe then all waveforms are affected otherwise only the selected waveform is affected.

Group: Zoom

Syntax:

```
ZOOm:HORizontal:POSition <NR3>
```

```
ZOOm:HORizontal:POSition?
```



Arguments: `<NR3>` is from 0 to 100, and is the percent of the waveform that is to the left of screen center.

Examples: `ZOOM:HORIZONTAL:POSITION 50`
centers the waveform on the display.

ZOOm:HORizontal:SCAle

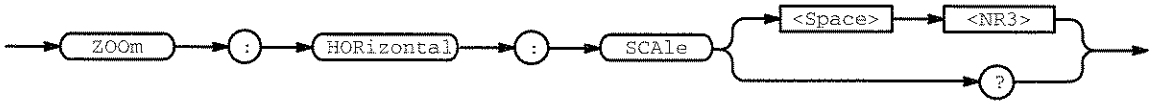
Sets or queries the horizontal expansion factor. This command is equivalent to using the front-panel **Horizontal Scale** knob when Zoom is on.

Group: Zoom

Syntax:

```
ZOOm:HORizontal:SCAle <NR3>
```

```
ZOOm:HORizontal:SCAle?
```



Arguments: `<NR3>` is the amount of expansion in the horizontal direction.

Examples: `ZOOM:HORIZONTAL:SCALE?`
might return `1.00E+0` as the horizontal scale factor.

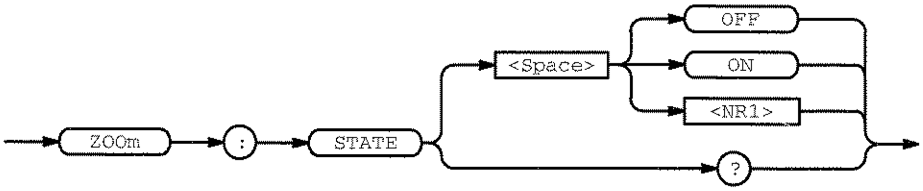
ZOOm:STATE

Turns Zoom mode on and off. When Zoom mode is on, the horizontal and vertical position and scale commands affect the waveform display not the acquisition. This is the only way to position and scale math and reference waveforms. This command is equivalent to turning **Zoom** on and off in the Zoom side menu.

Group: Zoom

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

`ZOOm:STATE?`



Arguments: `OFF` or `<NR1>` = 0 turns Zoom mode off.

`ON` or `<NR1>` ≠ 0 turns Zoom mode on.

Examples: `ZOOM:STATE ON`
enables the Zoom feature.

`ZOOM:STATE?`
returns either `0` or `1` depending on the state of Zoom mode.

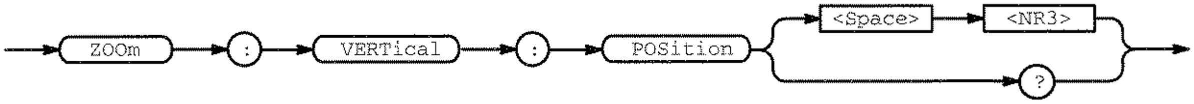
ZOOm:VERTical:POSition

Sets or queries the vertical position of 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

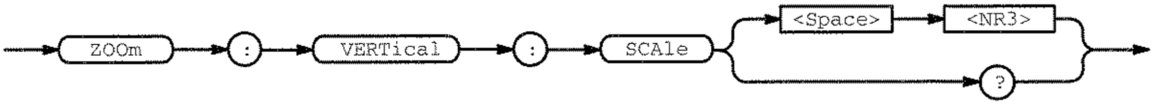
Sets or queries the vertical expansion and compression factor.

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 digitizing oscilloscope provides a status and event reporting system for the GPIB and RS-232-C interfaces. This system informs you of certain significant events that occur within the digitizing oscilloscope.

The digitizing oscilloscope status handling system consists of five 8-bit registers and two queues. This section describes these registers and components. It also explains how the event handling system operates.

Registers

The registers in the event handling system fall into two functional groups:

- Status Registers contain information about the status of the digitizing oscilloscope. They include the Standard Event Status Register (SESR) and the Status Byte Register (SBR).
- Enable Registers determine whether selected types of events are reported to the Status Registers and the Event Queue. They include the Device Event Status Enable Register (DESER), the Event Status Enable Register (ESER), and the Service Request Enable Register (SRER).

Status Registers

The Standard Event Status Register (SESR) and the Status Byte Register (SBR) record certain types of events that may occur while the digitizing 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 digitizing 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 NO TAG.) Reading the status registers tells you what types of events have occurred.

The Standard Event Status Register (SESR)—The SESR, shown in Figure NO TAG, records eight types of events that can occur within the digitizing oscilloscope. Use the *ESR? query to read the SESR register. Reading the register clears the bits of the register so that the register can accumulate information about new events.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Figure 3-1: The Standard Event Status Register (SESR)

Table 3-1: SESR Bit Functions

Bit	Function
7 (MSB)	**PON** (Power On). Shows that the digitizing oscilloscope was powered on. The completion of the diagnostic tests also sets this bit.
6	**URQ** (User Request). Shows that an Application menu button was pressed.
5	**CME** (Command Error). Shows that an error occurred while the digitizing oscilloscope was parsing a command or query. Command error messages are listed in Table NO TAG on page NO TAG.
4	**EXE** (Execution Error). Shows that an error occurred while the digitizing oscilloscope was executing a command or query. Execution error messages are listed in Table NO TAG on page NO TAG.
3	**DDE** (Device Error). Shows that a device error occurred. Device error messages are listed in Table NO TAG on page NO TAG.
2	**QYE** (Query Error). Shows that either an attempt was made to read the Output Queue when no data was present or pending, or that data in the Output Queue was lost.
1	**RQC** (Request Control). Not used.
0 (LSB)	**OPC** (Operation Complete). Shows that the operation is complete. This bit is set when all pending operations complete following a `*OPC` command.

The Status Byte Register (SBR)—shown in Figure NO TAG, records whether output is available in the Output Queue, whether the digitizing oscilloscope requests service, and whether the SESR has recorded any events.

Use a Serial Poll or the *STB? query 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.

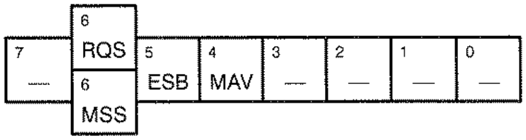


Figure 3-2: The Status Byte Register (SBR)

Table 3-2: SBR Bit Functions

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

Enable Registers

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.

Various commands set the bits in the Enable Registers. The Enable Registers and the commands used to set them are described below.

The Device Event Status Enable Register (DESER)—is shown in Figure NO TAG. 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.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Figure 3-3: The Device Event Status Enable Register (DESER)

The Event Status Enable Register (ESER)—is shown in Figure NO TAG. 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. Use the *ESE? query to read it.

7	6	5	4	3	2	1	0
PON	URQ	CME	EXE	DDE	QYE	RQC	OPC

Figure 3-4: The Event Status Enable Register (ESER)

The Service Request Enable Register (SRER)—is shown in Figure NO TAG. It controls which bits in the SBR generate a Service Request 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 or the MSS bit changes back to a zero.

7	6	5	4	3	2	1	0
—	—	ESB	MAV	—	—	—	—

Figure 3-5: The Service Request Enable Register (SRER)

The Enable Registers and the *PSC Command

The *PSC command controls the Enable Registers contents 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 non-volatile memory through a power cycle.

NOTE

*To enable the PON (Power On) event to generate a Service Request, 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.*

Queues

The digitizing oscilloscope status and event reporting system contains two queues: the Output Queue and the Event Queue.

The Output Queue

The digitizing 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.

WARNING

When a controller sends a query, an <EOM>, and a second query, the digitizing scope 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 of 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."

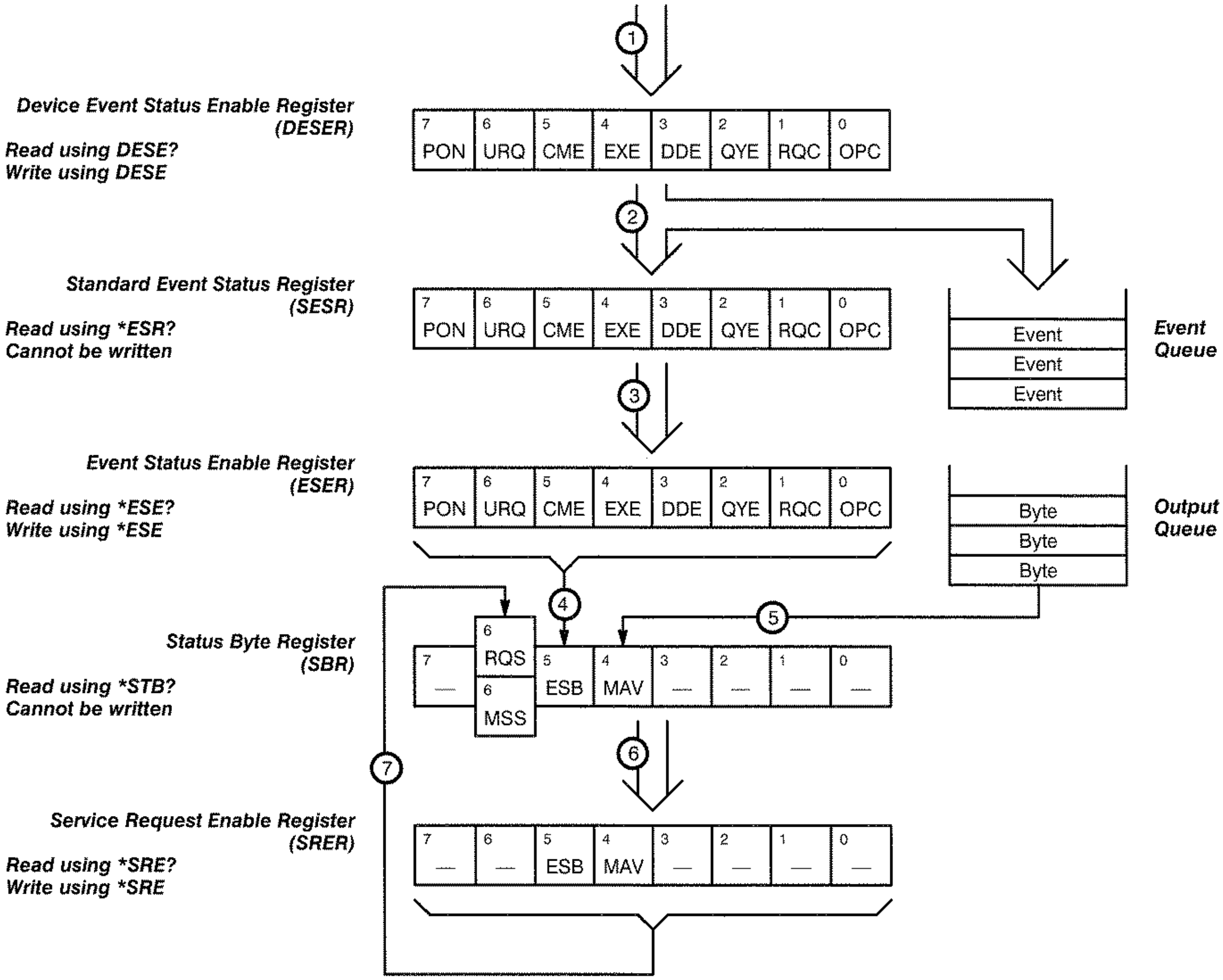
Read the Event Queue with the EVENT? query (which returns only the event number), with the EVMSG? query (which returns the event number and a text description of the event), or with the ALLEV? query (which returns all the event numbers along with a description of the event). Reading an event removes it from the queue.

Before reading an event from the Event Queue, you must use the *ESR? query to read the summary of the event from the SESR. This makes the events summarized by the *ESR? read available to the EVENT? and EVMSG? queries, 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 NO TAG, on page NO TAG, shows how to use the status and event handling system. In the explanation that follows, numbers in parentheses refer to numbers in Figure NO TAG.



Device Event Status Enable Register (DESER)
Read using DESE?
Write using DESE
Standard Event Status Register (SESR)
Read using *ESR?
Cannot be written
Event Status Enable Register (ESER)
Read using *ESE?
Write using *ESE
Status Byte Register (SBR)
Read using *STB?
Cannot be written
Service Request Enable Register (SRER)
Read using *SRE?
Write using *SRE
1
7 PON 6 URQ 5 CME 4 EXE 3 DDE 2 QYE 1 RQC 0 OPC
2
7 PON 6 URQ 5 CME 4 EXE 3 DDE 2 QYE 1 RQC 0 OPC
Event
Event
Event
Event Queue
3
7 PON 6 URQ 5 CME 4 EXE 3 DDE 2 QYE 1 RQC 0 OPC
Byte
Byte
Byte
Output Queue
4
5
7 — 6 RQS 6 MSS 5 ESB 4 MAV 3 — 2 — 1 — 0 —
6
7
7 — 6 — 5 ESB 4 MAV 3 — 2 — 1 — 0 —

Figure 3-6: Status and Event Handling Process

When an event occurs, a signal is sent to the DESER (1). If that type of event is enabled in the DESER (that is, if the bit for that event type is set to 1), the appropriate bit in the SESR is set to one and the event is recorded in the Event Queue (2). If the corresponding bit in the ESER is also enabled (3), then the ESB bit in the SBR is set to one (4).

When output is sent to the Output Queue, the MAV bit in the SBR is set to one (5).

When a bit in the SBR is set to one and the corresponding bit in the SRER is enabled (6), the MSS bit in the SBR is set to one and a service request is generated (7).

Synchronization Methods

Although most GPIB commands are completed almost immediately after being received by the digitizing 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 digitizing oscilloscope will continue processing other commands. This means that some operations will not be completed in the order that they were sent.

Sometimes the result of an operation depends on the result of an earlier operation. A first operation must complete before the next one gets processed. The digitizing oscilloscope's status and event reporting system provide ways to do this.

For example, a typical application might involve acquiring a single-sequence waveform then taking a measurement on the acquired waveform. You could use the following command sequence to do this:

```
/** Set up single-sequence acquisition **/
SELECT:CH1 ON
HORIZONTAL:RECORDLENGTH 500
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/** Acquire waveform data **/
ACQUIRE:STATE ON
/** Set up the measurement parameters **/
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/** Take amplitude measurement on acquired data **/
MEASUREMENT:IMMED:VALUE?
```

The acquisition of the waveform requires extended processing time. It may not finish before the digitizing oscilloscope takes an amplitude measurement (See Figure NO TAG). This can result in an incorrect amplitude value.

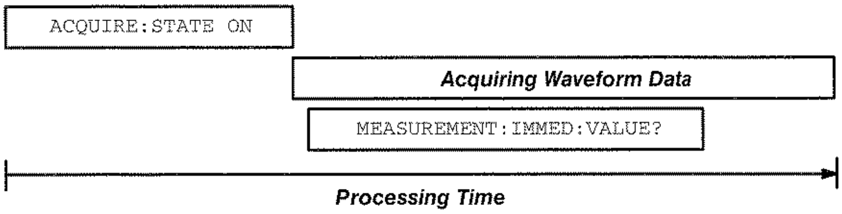


Figure 3-7: Command Processing Without Using Synchronization

To ensure the digitizing oscilloscope completes waveform acquisition before taking the measurement on the acquired data, you can synchronize the program. Figure NO TAG shows the desired processing sequence.

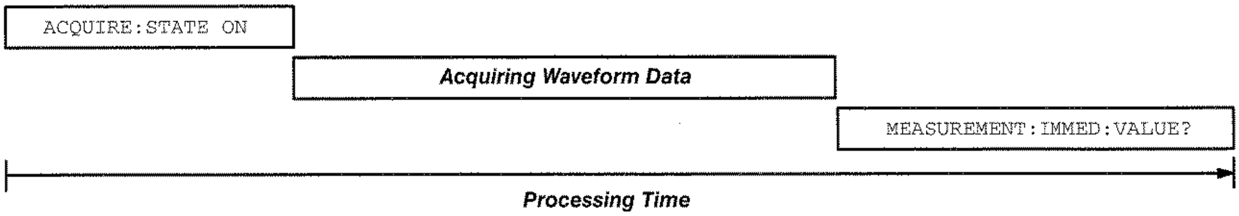


Figure 3-8: Processing Sequence With Synchronization

You can use four commands to synchronize the operation of the digitizing 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
HORIZONTAL:RECORDLENGTH 500
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 way to achieve synchronization, it is also the most costly. The processing time of the digitizing 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 digitizing oscilloscope's input buffer, but the commands will not be processed by the digitizing oscilloscope until all operations in process are complete. If the input buffer becomes full, the controller will be unable to write more commands to the buffer. This can cause a time-out.

Using the BUSY Query

The BUSY? query allows you to find out whether the digitizing oscilloscope is busy processing a command that has an extended processing time such as single-sequence acquisition.

The same command sequence using the BUSY? query for synchronization looks like this:

/* Set up single-sequence acquisition */

```
SELECT:CH1 ON
HORIZONTAL:RECORDLENGTH 500
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. The BUSY? query helps you avoid time-outs caused by writing too many commands to the input buffer. The controller is still tied up, though, and the repeated BUSY? query will result 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 achieve synchronization by using this command with either a serial poll or service request handler.

Serial Poll Method—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) will be enabled and the Event Status Bit (ESB) in the Status Byte Register will be 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
HORIZONTAL:RECORDLENGTH 500
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—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. You can 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 will be generated.

The same command sequence using the *OPC command for synchronization looks like this:

```
/* Set up single-sequence acquisition */
SELECT:CH1 ON
HORIZONTAL:RECORDLENGTH 500
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

The *OPC? query places a 1 in the Output Queue once an operation is complete. A timeout could occur if you try to read the output queue before there is any data in it.

The same command sequence using the *OPC? query for synchronization looks like this:

```
/* Set up single-sequence acquisition */
SELECT:CH1 ON
HORIZONTAL:RECORDLENGTH 500
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 NO TAG through NO TAG list all the programming interface messages the digitizing oscilloscope generates in response to commands and queries.

For most messages, a secondary message from the digitizing 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 NO TAG shows the messages when the system has no events or status to report. These have no associated SESR bit.

Table 3-3: No Event Messages

Code	Message
0	No events to report – queue empty
1	No events to report – new events pending *ESR?

Table NO TAG shows the error messages generated by improper command syntax. Check that the command is properly formed and that it follows the rules in the Command Syntax chapter starting on page 2-1.

Table 3-4: Command Error Messages—CME Bit 5

Code	Message
100	Command error
102	Syntax error
103	Invalid separator
104	Data type error
105	GET not allowed
108	Parameter not allowed
110	Command header error
111	Header separator error
112	Program mnemonic too long
113	Undefined header

Table NO TAG lists the execution errors that are detected during execution of a command. In these error messages, you should read “macro” as “alias.”

Table 3-5: Execution Error Messages—EXE Bit 4

Code	Message
200	Execution error
201	Invalid while in local
210	Trigger error
211	Trigger ignored
212	Arm ignored
220	Parameter error
221	Settings conflict
222	Data out of range
223	Too much data
224	Illegal parameter value
230	Data corrupt or stale
240	Hardware error
241	Hardware missing
242	Hardware configuration error
243	Hardware I/O device error
260	Expression error
261	Math error in expression
2200	Measurement error, Measurement system error
2201	Measurement error, Zero period
2202	Measurement error, No period found
2203	Measurement error, No period, second waveform
2204	Measurement error, Low signal amplitude
2205	Measurement error, Low amplitude, second waveform
2206	Measurement error, Invalid gate
2207	Measurement error, Measurement overflow
2208	Measurement error, Waveform does not cross Mid Ref
2209	Measurement error, No second Mid Ref crossing

Table 3-5: Execution Error Messages—EXE Bit 4 (Cont.)

Code	Message
2210	Measurement error, No Mid Ref crossing, second waveform
2211	Measurement error, No backwards Mid Ref crossing
2212	Measurement error, No negative crossing
2213	Measurement error, No positive crossing
2214	Measurement error, No crossing
2215	Measurement error, No crossing, second waveform
2216	Measurement error, No crossing, target waveform
2217	Measurement error, Constant waveform
2218	Measurement error, Unused
2219	Measurement error, No valid edge – No arm sample
2220	Measurement error, No valid edge – No arm cross
2221	Measurement error, No valid edge – No trigger cross
2222	Measurement error, No valid edge – No second cross
2223	Measurement error, waveform mismatch
2224	Measurement error, WAIT calculating
2225	Measurement error, No waveform to measure
2226	Null Waveform
2227	Positive and Negative Clipping
2228	Measurement error, Positive Clipping
2229	Measurement error, Negative Clipping
2230	Measurement error, High Ref < Low Ref
2235	Math error, Invalid math description
2236	Math error, Reference waveform is invalid
2237	Math error, Out of acquisition memory
2240	Invalid password
2241	Waveform request is invalid
2242	Data start and stop > record length
2243	Waveform requested is not a data source
2244	Waveform requested is not turned on

Table 3-5: Execution Error Messages—EXE Bit 4 (Cont.)

Code	Message
2245	Saveref error, Selected channel is turned off
2246	Saveref error, Selected channel data invalid
2247	Saveref error, Out of reference memory
2248	Saveref error, Source reference data invalid
2249	Reference deletion error, Waveform in use for math
2260	Calibration error
2270	Alias error
2271	Alias syntax error
2272	Alias execution error
2273	Illegal alias label
2274	Alias parameter error
2275	Alias definition too long
2276	Alias expansion error
2277	Alias redefinition not allowed
2278	Alias header not found
2279	Alias label too long
2280	Alias table full
2285	TekSecure® Pass
2286	TekSecure® Fail
2290	Limit error, reference in use
2291	Limit error, reference data invalid
2292	Limit error, out of reference memory
2293	Limit error, selected channel is turned off
2301	Cursor error, off-screen

Table NO TAG lists the device errors that can occur during digitizing oscilloscope operation. These errors may indicate that the oscilloscope needs repair.

Table 3-6: Device Error Messages—DDE Bit 3

Code	Message
300	Device-specific error
310	System error
311	Memory error
312	PUD memory lost
313	Calibration memory lost
314	Save/recall memory lost
315	Configuration memory lost
350	Queue overflow (does not set DDE bit)

Table NO TAG lists the system event messages. These messages are generated whenever certain system conditions occur.

Table 3-7: System Event Messages

Code	Message
400	Query event
401	Power on (PON bit 7 set)
402	Operation complete (OPC bit 0 set)
403	User request (URQ bit 6 set)
404	Power fail (DDE bit 3 set)
405	Request control
410	Query INTERRUPTED (QYE bit 2 set)
420	Query UNTERMINATED (QYE bit 2 set)
430	Query DEADLOCKED (QYE bit 2 set)
440	Query UNTERMINATED after indefinite response (QYE bit 2 set)
450	Right menu button #1 pushed (URQ bit 6 set)
451	Right menu button #2 pushed (URQ bit 6 set)
452	Right menu button #3 pushed (URQ bit 6 set)
453	Right menu button #4 pushed (URQ bit 6 set)

Table 3-7: System Event Messages (Cont.)

Code	Message
454	Right menu button #5 pushed (URQ bit 6 set)
460	Bottom menu button #1 pushed (URQ bit 6 set)
461	Bottom menu button #2 pushed (URQ bit 6 set)
462	Bottom menu button #3 pushed (URQ bit 6 set)
463	Bottom menu button #4 pushed (URQ bit 6 set)
464	Bottom menu button #5 pushed (URQ bit 6 set)
465	Bottom menu button #6 pushed (URQ bit 6 set)
466	Bottom menu button #7 pushed (URQ bit 6 set)

Table NO TAG 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

Code	Message
500	Execution warning
510	String data too long, truncated
525	Parameter underrange
526	Parameter overrange
527	Parameter rounded
528	Parameter out of range
530	Data stop > stop, Values swapped internally
531	Data stop > record length, Curve truncated
532	Curve data too long, curve truncated
540	Measurement warning
541	Measurement warning, Low signal amplitude
542	Measurement warning, Unstable histogram
543	Measurement warning, Low resolution
544	Measurement warning, Uncertain edge
545	Measurement warning, Invalid in minmax
546	Measurement warning, Need 3 edges

Table 3-8: Execution Warning Messages—EXE Bit 4 (Cont.)

Code	Message
547	Measurement warning, Clipping positive/negative
548	Measurement warning, Clipping positive
549	Measurement warning, Clipping negative

Table NO TAG shows internal errors that indicate an internal fault in the digitizing oscilloscope.

Table 3-9: Internal Warning Messages

Code	Message
600	Internal warning
620	Internal warning, Bad thermistor
630	Internal warning, 50 Ω overload

Programming Examples

The example programs illustrate methods you can use to control the digitizing oscilloscope from the GPIB interface. The diskettes that come with this manual contain listings for these programs written in Microsoft QuickBASIC 4.5 and Microsoft QuickC 2.5.

The programs run on a PC-compatible system equipped with a Tektronix (National Instruments) GPIB board and associated drivers. For example, the programs will work with a Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB package (See Figure 4-1).

All the example programs assume that the GPIB system recognizes the digitizing oscilloscope as DEV1 and the PC (controller) as GPIB0. You can use the `IBCONF.EXE` program to assign these names.

The example software includes:

- `MEAS`: automatically measures waveform parameters.
- `COMM`: shows communication between controller and oscilloscope.
- `GETWFM`: reads a waveform from an oscilloscope and stores it in a file.
- `CURSOR`: uses cursors to measure waveform parameters.
- `TL`: a talker-listener program.

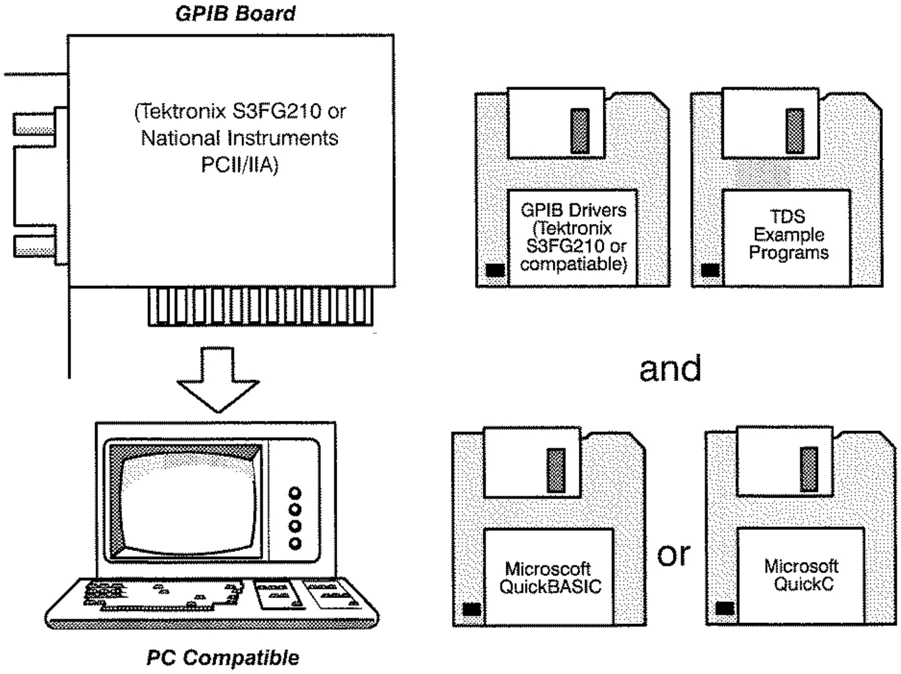


Figure 4-1: Equipment Needed to Run the Example Programs

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 programs you compile in the Sources directory work with the Tektronix S3FG210 (National Instruments GPIB-PCII-IIA) GPIB system. It may take extra steps or changes to get them to work with older Tektronix GURU and other GPIB systems.

Compiling And Linking Your Example Quick-C Programs

To make an executable for any example, perform the following:

□ **Step 1:** Install QuickC. Select the SMALL memory model. Be sure to set up your path so DOS can access the QuickC directory.

□ **Step 2:** Install the Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB board and drivers. Remember to identify the GPIB device as DEV1. You can use the `IBCONF.EXE` program to do this.

□ **Step 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, you might type:

```
mkdir examples

cd examples

copy B:\quick-c\*.* .
```

□ **Step 4:** For this installation, you will also want to copy `DECL.H` and `MCIB.OBJ` from your Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB drivers directory to this directory. For example, if the GPIB drivers are in the `gpib-pc` directory and you are in the example programs directory, you would type:

```
copy \gpib-pc\decl.h .

copy \gpib-pc\mcib.obj .
```

□ **Step 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 `CURSOR.C`, type: `nmake cursor.mak`

To compile and link `TL.C`, type: `nmake tl.mak`

□ **Step 6:** Run the program by simply typing the program name.

To run `meas`, type: `meas`

To run `comm`, type: `comm`

To run `getwfm`, type: `getwfm`

To run `cursor`, type: `cursor`

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:

□ **Step 1:** Install QuickBASIC.

□ **Step 2:** Install the Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB board and drivers. Remember to reboot your PC to initialize the GPIB drivers.

□ **Step 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, you might type:

```
mkdir examples

cd examples

copy b:\q-basic\*.* .
```

□ **Step 4:** For this installation, you will also want to copy `QBDECL.BAS` and `QBIB.OBJ` from your Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB drivers directory to the directory your example programs are in. For example, if the GPIB drivers are in the `gpib-pc` directory and you are in the example programs directory, you would type:

```
copy \gpib-pc\qbdecl.bas .

copy \gpib-pc\qbib.obj .
```

□ **Step 5:** Perform the following two steps for example programs:

1) Compile the program by 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 `CURSOR.BAS` , type: `bc /o cursor.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) by 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 `CURSOR.OBJ`, type: `link cursor.obj+qbib.obj;`

To link `TL.OBJ`, type: `link tl.obj+qbib.obj;`

`GPIBIO.BAS` is a collection of input/output routines used by the other programs and is included for proper file compilation.

□ **Step 6:** Run the program by simply typing the program name.

To run `meas`, type: `meas`

To run `comm`, type: `comm`

To run `getwfm`, type: `getwfm`

To run `cursor`, type: `cursor`

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 reenable front-panel operation, do one of the following: cycle power on the digitizing 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.*

Appendix A: Character Charts

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

Table A-1: The TDS Character Set

	0	1	2	3	4	5	6	7
0	NUL 0 0	∿ 12 16	space 5 32	0 10 48	@ 14 64	P 10 80	‘ 5 96	p 11 112
1	ˇ 10 1	Ω 14 17	! 5 33	1 10 49	A 12 65	Q 13 81	a 8 97	q 10 113
2	¿ 7 2	△ 15 18	” 7 34	2 10 50	B 10 66	R 10 82	b 11 98	r 7 114
3	Ç 8 3	ᴮW 11 19	# 10 35	3 10 51	C 10 67	S 9 83	c 8 99	s 8 115
4	¨ 10 4	_/‾ 12 20	$ 10 36	4 10 52	D 12 68	T 10 84	d 10 100	t 7 116
5	` 10 5	‾_ 12 21	% 12 37	5 10 53	E 9 69	U 12 85	e 9 101	u 11 117
6	ᵯ 12 6	µ 12 22	& 12 38	6 10 54	F 9 70	V 11 86	f 6 102	v 10 118
7	‘ 5 7	˘ 10 23	’ 5 39	7 10 55	G 11 71	W 15 87	g 10 103	w 14 119
8	¡ 5 8	— 16 24	(6 40	8 10 56	H 13 72	X 10 88	h 11 104	x 9 120
9	HT 0 9	＿ 16 25	) 6 41	9 10 57	I 6 73	Y 10 89	i 5 105	y 10 121
A	LF 0 10	∞ 12 26	* 8 42	: 6 58	J 7 74	Z 10 90	j 5 106	z 8 122
B	´ 10 11	ESC 0 27	+ 11 43	; 6 59	K 10 75	[6 91	k 10 107	{ 6 123
C	± 11 12	x̄ 9 28	, 6 44	< 11 60	L 8 76	\\ 9 92	l 5 108	| 6 124
D	CR 0 13	≠ 11 29	— 11 45	= 11 61	M 15 77	] 6 93	m 15 109	} 6 125
E	– 10 14	~ 10 30	. 6 46	> 11 62	N 13 78	^ 11 94	n 11 110	~ 11 126
F	● 7 15	° 10 31	/ 9 47	? 7 63	O 13 79	_ 11 95	o 10 111	| 3 127

Table A-2: The ASCII & GPIB Code Chart

	0	1	2	3	4	5	6	7
0	0 NUL 0 0	20 DLE 10 16	40 0 SP 20 32	60 16 0 30 48	100 0 @ 40 64	120 16 P 50 80	140 0 ‘ 60 96	160 16 p 70 112
1	1 GTL SOH 1 1	21 LLO DC1 11 17	41 1 ! 21 33	61 17 1 31 49	101 1 A 41 65	121 17 Q 51 81	141 1 a 61 97	161 17 q 71 113
2	2 STX 2 2	22 DC2 12 18	42 2 ” 22 34	62 18 2 32 50	102 2 B 42 66	122 18 R 52 82	142 2 b 62 98	162 18 r 72 114
3	3 ETX 3 3	23 DC3 13 19	43 3 # 23 35	63 19 3 33 51	103 3 C 43 67	123 19 S 53 83	143 3 c 63 99	163 19 s 73 115
4	4 SDC EOT 4 4	24 DCL DC4 14 20	44 4 $ 24 36	64 20 4 34 52	104 4 D 44 68	124 20 T 54 84	144 4 d 64 100	164 20 t 74 116
5	5 PPC ENQ 5 5	25 PPU NAK 15 21	45 5 % 25 37	65 21 5 35 53	105 5 E 45 69	125 21 U 55 85	145 5 e 65 101	165 21 u 75 117
6	6 ACK 6 6	26 SYN 16 22	46 6 & 26 38	66 22 6 36 54	106 6 F 46 70	126 22 V 56 86	146 6 f 66 102	166 22 v 76 118
7	7 BEL 7 7	27 ETB 17 23	47 7 ’ 27 39	67 23 7 37 55	107 7 G 47 71	127 23 W 57 87	147 7 g 67 103	167 23 w 77 119
8	10 GET BS 8 8	30 SPE CAN 18 24	50 8 (28 40	70 24 8 38 56	110 8 H 48 72	130 24 X 58 88	150 8 h 68 104	170 24 x 78 120
9	11 TCT HT 9 9	31 SPD EM 19 25	51 9) 29 41	71 25 9 39 57	111 9 I 49 73	131 25 Y 59 89	151 9 i 69 105	171 25 y 79 121
A	12 LF A 10	32 SUB 1A 26	52 10 * 2A 42	72 26 : 3A 58	112 10 J 4A 74	132 26 Z 5A 90	152 10 j 6A 106	172 26 z 7A 122
B	13 VT B 11	33 ESC 1B 27	53 11 + 2B 43	73 27 ; 3B 59	113 11 K 4B 75	133 27 [5B 91	153 11 k 6B 107	173 27 { 7B 123
C	14 FF C 12	34 FS 1C 28	54 12 , 2C 44	74 28 < 3C 60	114 12 L 4C 76	134 28 \ 5C 92	154 12 l 6C 108	174 28 | 7C 124
D	15 CR D 13	35 GS 1D 29	55 13 – 2D 45	75 29 = 3D 61	115 13 M 4D 77	135 29] 5D 93	155 13 m 6D 109	175 29 } 7D 125
E	16 SO E 14	36 RS 1E 30	56 14 . 2E 46	76 30 > 3E 62	116 14 N 4E 78	136 30 ^ 5E 94	156 14 n 6E 110	176 30 ~ 7E 126
F	17 SI F 15	37 US 1F 31	57 15 / 2F 47	77 UNL ? 3F 63	117 15 O 4F 79	137 UNT _ 5F 95	157 15 o 6F 111	177 DEL (RUBOUT) 7F 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY

octal 25 PPU GPIB code

NAK ASCII character

hex 15 21 decimal

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
*CLS
*DDT
*ESE
*ESR
*IDN
*LRN
*OPC
*PSC
*PUD
*RCL
*RST
*SAV
*SRE
*STB
*TRG
*TST
*WAI
ABOrt
ABSolute
AC
ACCept
ACQuire
ACQUISition
ACTivate
ALIas
ALL
ALLEv
ALLOcate
ALWays
AMPlitude
AND
APPMenu
AREA
ASC
ASCIi
AT
AUTO
AUTOSet
AUXiliary
AVErage
BACkwards
BAud
BANdwidth
BAUd
BELl
BIN
BIT_Nr
BMP
BMP
BN_Fmt
BOTTOM1
BOTTOM2
BOTTOM3
BOTTOM4
BOTTOM5
BOTTOM6
BOTTOM7
BOX
BURst
BUSY
BY
BYT_Nr
BYT_Or
CATalog
CARea
CENtronics
CENtronics
CHKsm0
CH1
CH2
CH3
CH4
CH3
CH4
CLAss
CLEar
CLEARMenu
CLEARSpool
CLOCK
CLEARSNapshot
CLEARSpool
CLOCk
CMEan
COMpare
CONTRast
CONTROl
COUNt
COUPling
CPU
CRMs
CROSSHair
CRVchk
CURSor
CURSOR1
CURSOR2
CURVe
CUSTom
CYCLE
DATa
DATE
DC
DEFIne
DELay
DELAYEd
DELEte
DELTa
DESE
DESKJet
DESTination
DIAg
DIRection
DISplay
DOTs
DPU411
DPU412
ECL
EDGE
EDGE1
EDGE2
EIther
ENCdg
ENV
ENVelope
EPSColor
EPSImage
EPSColor
EPSMono
EPSMonoEPSOn
EVEN
EVENT
EVENTS
EVENTSTime
EVMsg
EVQty
EXECute
EXERciser
FACtory
FAIl
FALSe
FIELD1
FIELD2
FIELDEither
FIFty
FILTer
FIRst
FLAg
FORCe
FORMat
FORWards
FPAnel
FRAme
FREE
FREQuency
FULl
FUNCtion
GATing
GLItch
GND
GPIb
GRAticule
GRId
HALt
HARDCopy
HARDFlagging
HARDFlagging
HBArHDELTA
HDR
HDELTA
HEADer
HERtz
HFRej
HIGH
HIGHLimit
HIRes
HIStogram
HOLdoff
HORizontal
HPOS
HPGl
HPOS1
HPOS2
HUNdred
ID
IMMed
INDependent
IMPedance
INDependent
INFInite
INFPersist
INIT
INPut
INTENSIFied
INTENSITy
INTERLeafINTERLAce
INVert
LABel
LANdscape
LASERJet
LAYout
LESSLimit
LESSThan
LEVel
LFRej
LIMit
LINE
LINES
LINEAr
LIVe
LOCk
LOG
LONG
LOGIc
LONG
LOW
LOWLimit
LSB
MAIn
MATH1
MATH2
MATH3
MAXimum
MEAN
MEAS1
MEAS2
MEAS3
MEAS4
MEASUrement
MEG
MESSage
METHod
MID
MID2
MINImum
MINMax
MODe
MORELimit
MOREThan
MSB

NAMe
NANd
NDUTy
NEGAtive
NEWpass
NOISErej
NONe
NOR
NORMal
NOVershoot
NR_PNTSc
NUMACq
NUMAVg
NUMEnv
NWIdth
ODD
ODD
OFF
OFFSet
ON
ONCe
OR
OPTion
OUTside
OVERAll
PAIred
PARIty
PAL
PAIred
PARity
PARTial
PASSWord
PATtern
PCX
PDUTy
PEAKdetect
PERCent
PERIod
PERSistence
PHAse
PK2pk
POLarity
PORT
PORTRait
POSition
POSITION1
POSITION2
POSITIVe
POVershoot
PRObe
PT_Fmt
PT_Off
PULse
PWIdth
RATE1
RATE2
RATE3
RATE4
RECAll
RECOrdlength
REF1
REF2
REF3
REF4
REFLevel
REFSelect
REJect
REM
REPEt
RESet
RESUlt
RI
RIBinary
RIGHT1
RIGHT2
RIGHT3
RIGHT4
RIGHT5
RISE
RMS
RP
RPBinary
RS232
RUN
RUNSAfter
RUNSTop
RUNT
SAMple
SAVe
SCAle
SCAN
SECAm
SECdiv
SECOnds
SELect
SELFdiag
SEQuence
SET
SETLevel
SETUp
SHORT
SHOW
SINX
SLOpe
SNAp
SOFTFlagging
SNAPShot
SOFTFlagging
SOUrce
SOURCE1
SOURCE2
SRIbinary
SRPbinary
STARt
STATE
STOP
STOPAfter
STOPBits
STOPBits
STORe
STYle
SYNc
SYStem
TARget
TEMPLate
TEXt
THInkjet
TIFf
TIME
THReshold
TIFf
TIMe
TITLe
TOLerance
TRACk
TRIGBar
TRACk
TRIGAfter
TRIGBar
TRIGger
TRIGT
TRUe
TTL
TWEnty
TYPe
UNIts
UNLock
VALue
VARpersist
VBArs
VDELTAVDELTA
VECtors
VERBose
VERTical
VIDeo
VOLts
WAVEform
WAVFrm
WFId
WFMPre
WHEn
WIDth
WIThin
XINcr
XMUlt
XOFf
XUNit
XY
XYZ
XZEro
Y
YMUlt
YOFf
YT
YUNit
YZ
YZEro
ZMUlt
ZOFf
ZOOm
ZUNit
ZZEro

Appendix C: Interface Specifications

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

GPIB Function Subsets

The digitizing oscilloscope supports many GPIB function subsets, as listed below. Some of the listings describe subsets that the digitizing oscilloscope does not support.

- SH1 (Source Handshake). The digitizing oscilloscope can transmit multiline messages across the GPIB.
- AH1 (Acceptor Handshake). The digitizing oscilloscope can receive multiline messages across the GPIB.
- T5 (Talker). The digitizing oscilloscope becomes a talker when its talk address is sent 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 another device's talk address is sent with ATN asserted. The digitizing oscilloscope has talk-only capability for hardcopy operation.
- L4 (Listener). The digitizing oscilloscope becomes a listener when its listen address is sent with the ATN (Attention) line asserted. The digitizing oscilloscope does not have listen-only capability.
- SR1 (Service Request). The digitizing oscilloscope asserts an SRQ (Service Request) line to notify the controller when it requires service.
- RL1 (Remote/Local). The digitizing oscilloscope responds to both the GTL (Go To Local) and LLO (Local Lock Out) interface messages.
- PP0 (Parallel Poll). The digitizing oscilloscope has no parallel poll capability. It does not respond to the following interface messages: PPC, PPD, PPE, and PPU. The digitizing 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 digitizing 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 digitizing oscilloscope responds to the GET (Group Execute Trigger) interface message.
- C0 (Controller). The digitizing oscilloscope cannot control other devices.
- E2 (Electrical). The digitizing oscilloscope uses tristate buffers to provide optimal high-speed data transfer.

Interface Messages

Table A-3 shows the standard interface messages that are supported by the digitizing oscilloscope.

Table A-3: Digitizing Oscilloscope Standard Interface Messages

Message	GPIB
DCL	Yes
GET	Yes
GTL	Yes
LLO	Yes
PPC	No
PPD	No
PPE	No
PPU	No
SDC	Yes
SPD	Yes
SPE	Yes
TCT	No
UNL	Yes
UNT	Yes
Listen Addresses	Yes
Talk Addresses	Yes

Appendix D: Factory Initialization Settings

The factory initialization settings provide a known state for the digitizing oscilloscope.

Settings

Factory initialization sets values as shown in Table A-4.

Table A-4: Factory Initialization Defaults

Control	Changed by Factory Init to
Acquire mode	Sample
Acquire repetitive signal (TDS 420/460/520/540 only)	ON (Enable ET)
Acquire stop after	RUN/STOP button only
Acquire # of averages	16
Acquire # of envelopes	10
Channel selection	Channel 1 on, all others off
Cursor H Bar 1 position	10% of graticule height (−3.2 divs from the center)
Cursor H Bar 2 position	90% of the graticule height (+3.2 divs from the center)
Cursor V Bar 1 position	10% of the record length
Cursor V Bar 2 position	90% of the record length
Cursor function	Off
Cursor mode	Independent
Cursor time units	Seconds
Delayed edge trigger coupling	DC
Delayed edge trigger level	0 V
Delayed edge trigger slope	Rising
Delayed edge trigger source	Channel 1
Delay trigger average #	16
Delay trigger envelope #	10

Table A-4: Factory Initialization Defaults (Cont.)

Control	Changed by Factory Init to
Delay time, delayed runs after main	TDS 420/460: 10 ns TDS 520/540/620/640: 16.0 ns
Delay time, delayed triggerable after main	TDS 420/460: 60 ns TDS 520/540/620/640: 16.0 ns
Delay events, triggerable after main	2
Delayed, delay by ...	Delay by Time
Delayed, time base mode	Delayed Runs After Main
Display format	YT
Display graticule type	Full
Display intensity – contrast	150%
Display intensity – text	60%
Display intensity – waveform	75%
Display intensity – overall	85%
Display interpolation filter	Sin(x)/x
Display style	Vectors
Display trigger “T”	On
Display variable persistence	500 ms
Edge trigger coupling	DC
Horizontal – delay trigger position	50%
Horizontal – delay trigger record length	500 points (10 divs)
Horizontal – delay trigger time/div.	50 µs
Horizontal – main trigger position	50%
Horizontal – main trigger record length	500 points (10 divs)
Horizontal – main trigger time/div.	500 µs
Horizontal – time base	Main only
Logic pattern trigger Ch4 (Ax2) input	X (don’t care)
Logic state trigger Ch4 (Ax2) input	Rising edge

Table A-4: Factory Initialization Defaults (Cont.)

Control	Changed by Factory Init to
Logic trigger input (pattern and state)	Channel 1 = H (high), Channels 2 & 3 (Ax1) = X (don't care)
Logic trigger threshold (all channels) (pattern and state)	1.4 V (when 10X probe attached)
Logic trigger class	Pattern
Logic trigger logic (pattern and state)	AND
Logic trigger triggers when ... (pattern and state)	Goes TRUE
Main trigger type	Edge
Math function (single wfm)	Invert (Inv) for math3
Math operator (dual wfm)	+ for math1, − for math2
Math source 1 (single and dual)	Channel 1 (Ch1)
Math source 2	Channel 2 (Ch2)
Math type	Dual Wfm Math for math1 and math2, single for math3
Measure Delay to	Channel 1 (Ch1)
Measure Delay edges	Both rising and forward searching
Measure High-Low Setup	Histogram
Measure High Ref	90% and 0 V (units)
Measure Gating	Off
Measure Low Ref	10% and 0 V (units)
Measure Mid Ref	50% and 0 V (units)
Measure Mid2 Ref	50% and 0 V (units)
Message Window coordinates	74, 84, 475, 135
Pattern trigger Ch4/Ax2 input	X (don't care)
Pulse glitch trigger polarity	Positive
Pulse runt high threshold	2.0 V
Pulse runt low threshold	0.0 V
Pulse runt trigger polarity	Positive
Pulse trigger class	Glitch

Table A-4: Factory Initialization Defaults (Cont.)

Control	Changed by Factory Init to
Pulse trigger filter state	Accept glitch
Pulse trigger glitch width	2.0 ns
Pulse trigger level	0.8 V
Pulse trigger source (Glitch, runt, and width)	Channel 1 (Ch1)
Pulse width trigger when ...	Within limits
Pulse width upper limit	2.0 ns
Pulse width lower limit	2.0 ns
Pulse width trigger polarity	Positive
Saved setups	No change
Saved waveforms	No change
Trig Bar (TDS 820)	Short
Vertical bandwidth (all channels)	Full
Vertical coupling (all channels)	DC
Vertical impedance (termination) (all channels)	1 MΩ
Main trigger holdoff	0%
Edge trigger level	0.0 V
Main trigger mode	Auto
Edge trigger slope	Rising
Edge trigger source	Channel 1
Vertical offset (all channels)	0 V
Vertical position (all channels)	0 divs.
Vertical volts/div. (all channels)	100 mV/div.
Zoom horizontal (all channels)	1.0X
Zoom horizontal lock	All
Zoom horizontal position (all channels)	50% = .5 (the middle of the display)
Zoom state	Off
Zoom vertical (all channels)	1.0X

Table A-4: Factory Initialization Defaults (Cont.)

Control	Changed by Factory Init to
Zoom vertical position (all channels)	0 divs.

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 can be 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.

TEKSecure

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

Index