Programmer Manual

Tektronix

TLS 216
Logic Scope

070-8835-00

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

First Printing: September 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.

TLS 216 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.

P6240 WARRANTY

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product.

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

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

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

Table of Contents

Getting Started

Syntax and Commands

Status and Events

Examples

Appendices

Glossary and Index

List of Figures

List of Tables

Preface

This programmer manual covers the TLS 216 Logic Scope. It provides information on operating your logic scope using the General Purpose Interface Bus (GPIB) interface.

Section 1 *Getting Started* describes how to connect the logic scope to a controller.

Section 2 *Syntax and Commands* describes the structure and content of the messages your program sends to the logic scope. Section 2 also describes what each command does and provides examples of how you might use them. The subsection *Command Groups* lists commands by functional area. *Command Descriptions* describes the commands in alphabetical order.

Section 3 *Status and Events* describes how to use service requests (SRQ’s) and various event messages in your programs. This section also lists the status and error messages the logic scope provides.

Section 4 *Programming Examples* describes some example logic scope programs and how to compile them. The 3.5” and 5.25” disks that come with this manual contain an executable and a Microsoft QuickC 2.5 source code version of each program.

Related Manuals

The following documents are related to the use or service of the logic scope.

- The *TLS 216 User Manual* (Tektronix part number 070-8834-00) describes using a computer to control the logic scope through the GPIB interface.
- The *TLS 216 Reference* (Tektronix part number 070-8833-00) provides a quick overview of how to operate your logic scope.
- The *TLS 216 Performance Verification and Specifications Technical Reference* (Tektronix part number 070-8832-00) explains how to verify the performance of the logic scope.
- The *TLS 216 Service Manual* (Tektronix part number 070-8831-00) provides information for maintaining and servicing your logic scope to the module level.

Getting Started

You can write computer programs that remotely set the TLS 216 Logic Scope front panel controls or take measurements and read those measurements for further analysis or storage.

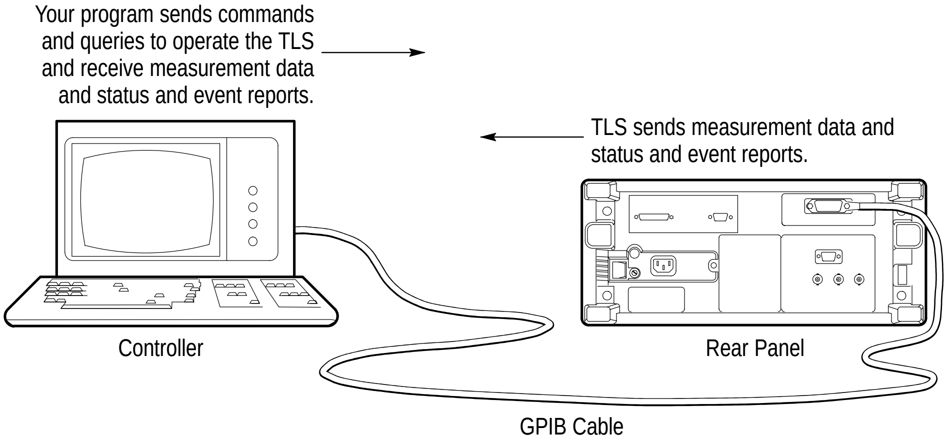

Your program sends commands and queries to operate the TLS and receive measurement data and status and event reports.
TLS sends measurement data and status and event reports.
Controller
Rear Panel
GPIB Cable

Figure 1–1: Interaction between the Controller and the Logic Scope

This section describes how to connect the logic scope to a remote system and configure it for remote communication using the appropriate front panel controls.

Setting Up Remote Communications

Even the best instrument control program will not do much if you do not properly connect the instrument to the controller. To connect to the controller, perform the following steps:

1. Locate the 24-pin GPIB connector on the rear panel of the logic scope, shown in Figure 1–2. This connector has a D-type shell and conforms to IEEE Std 488.1-1987.

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

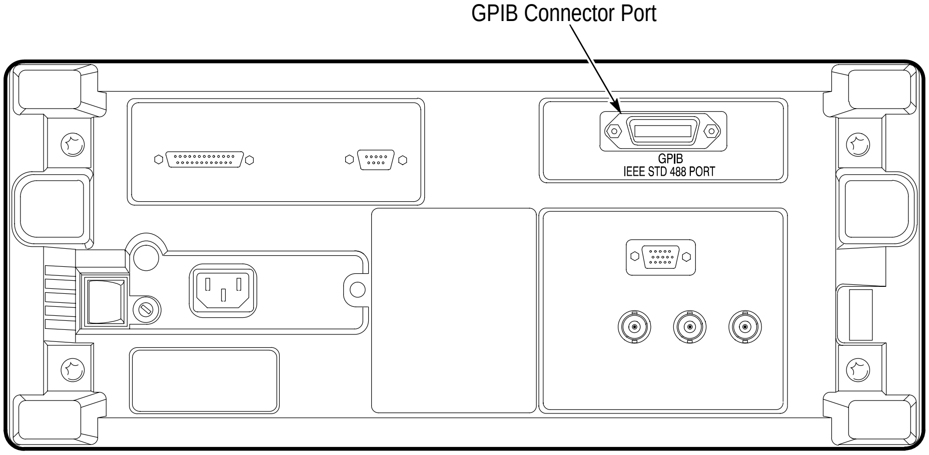


Figure 1–2: GPIB Connector Location

If needed, you can stack GPIB connectors as shown in Figure 1–3.

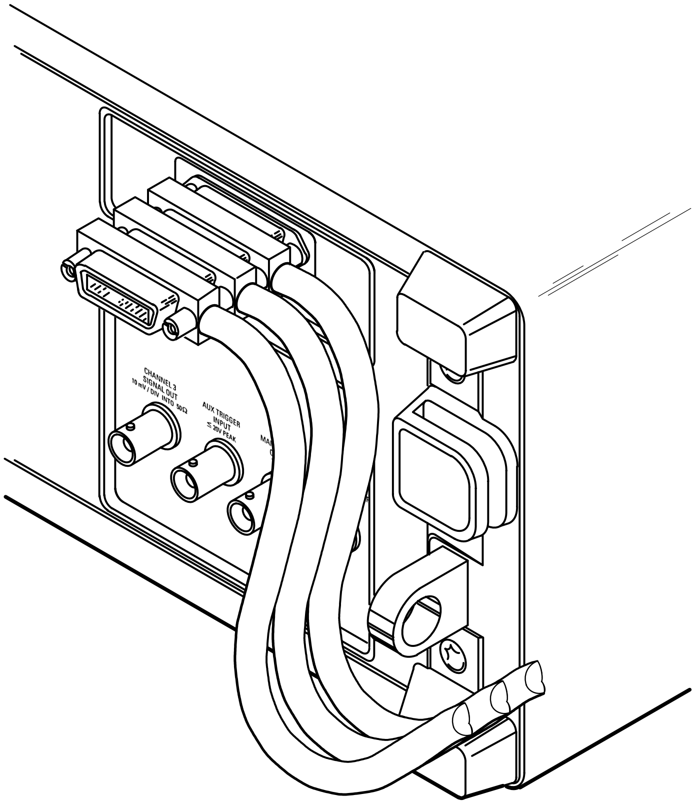


Figure 1–3: How to Stack GPIB Connectors

GPIB Requirements

Observe these rules when you use your logic scope 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 1–4. Do not use loop or parallel configurations.

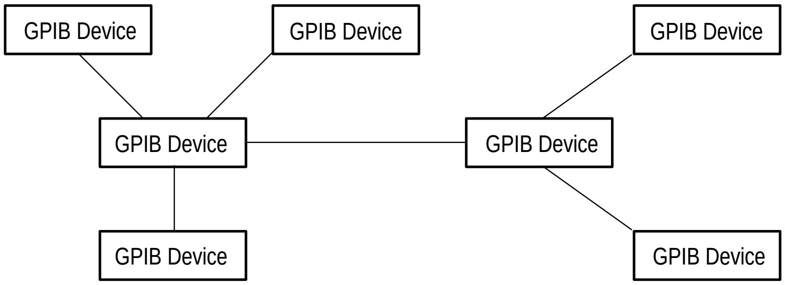


Figure 1–4: Typical GPIB Network Configurations

Appendix C: Interface Specifications, gives more information on the GPIB configuration of the logic scope.

Setting the GPIB Parameters

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

1. Press the UTILITY (SHIFT DISPLAY) button to display the Utility menu.
2. Press the System button in the main menu until it highlights the I/O selection in the pop-up menu. (See Figure 1–5.)

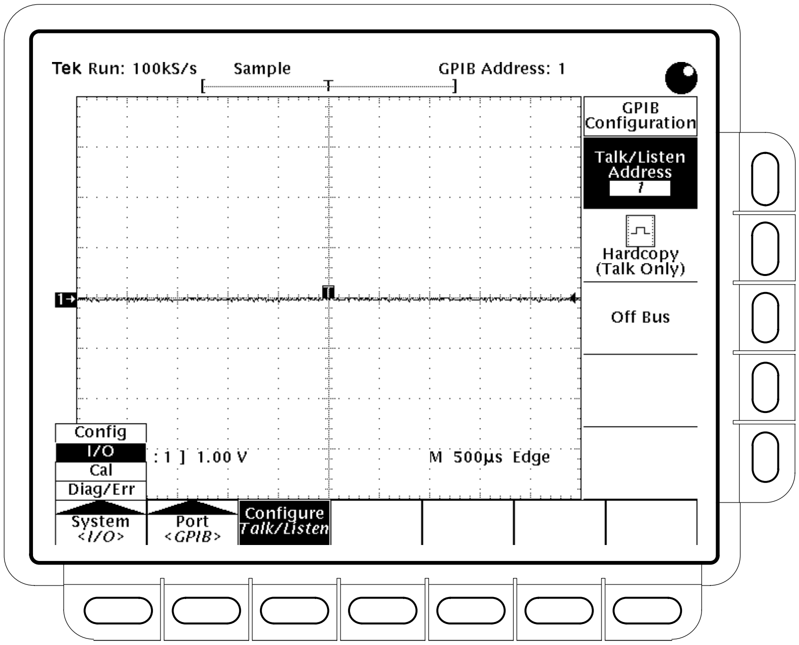


Figure 1–5: Selecting the I/O System in the Main Menu

3. Press the Port button in the main menu until it highlights the GPIB selection in the pop-up menu. (See Figure 1–6.)

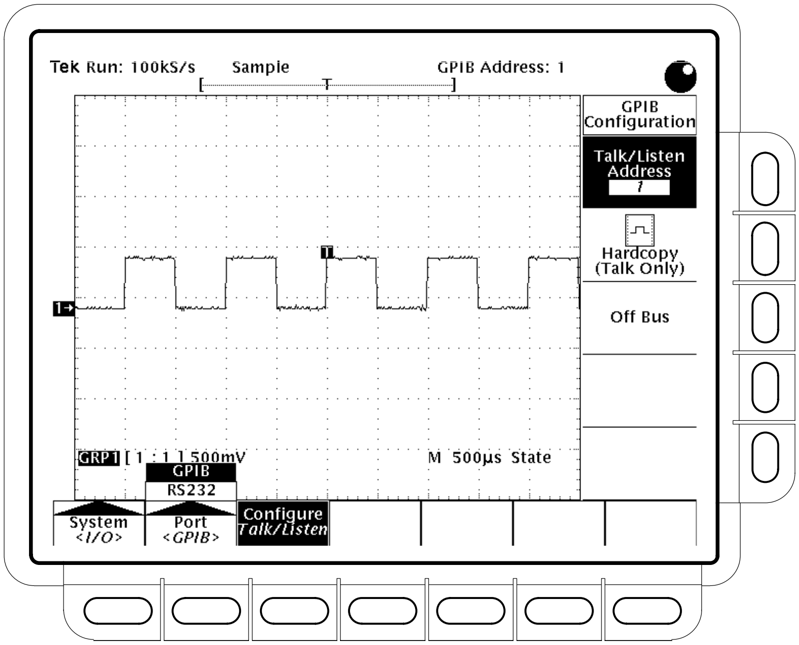


Figure 1–6: Selecting the GPIB Address in the GPIB Configuration Side Menu

4. Press the Configure button in the main menu to display the GPIB Configuration side menu. (See Figure 1–6.)

Other Communication Setups

The procedure just listed configures the logic scope for bidirectional communication with your controller. In certain situations you may want to disable communications or send hard copies directly to a non-488.2 hard copy device.

- To isolate the logic scope from the bus, press the Off Bus side menu button.

  This setting disables all communication with the controller.

- To enter a special mode of operation to communicate directly with non-488.2 hard copy devices, press the Hardcopy side menu button.

  This setting places the logic scope in Talk Only mode and directs it to send hard copy information when you press the HARDCOPY button only (and to accept a HARDCOPY ABORT command).

Command Syntax

You can control the TLS 216 Logic Scope 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 logic scope uses to process them. The next section, entitled *Commands,* lists the commands and queries themselves.

You transmit commands to the logic scope 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 logic scope to perform a specific action. Queries cause the logic scope 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 logic scope 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.

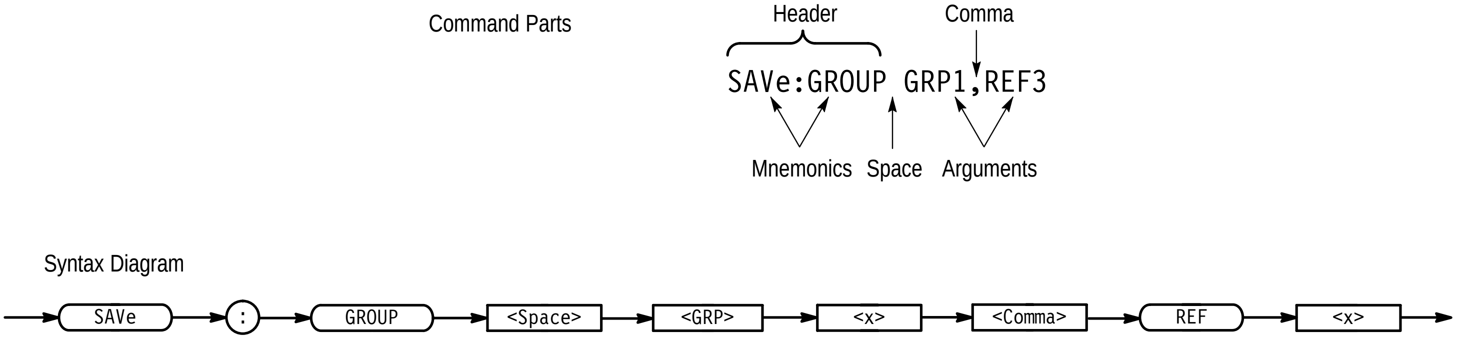


Figure 2–1: Command Message Elements

Commands

Commands cause the logic scope 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 logic scope 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 logic scope 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 Logic Scope

You can clear the Output Queue and reset the logic scope 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 logic scope ignores commands consisting of any combination of white space characters and line feeds.

Abbreviating Commands

You can abbreviate many logic scope 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 logic scope 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 instrument concatenates all the responses 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. You may concatenate set commands and queries in the same message. For example,

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

   is a valid message that sets the acquisition mode to normal, then queries the number of acquisitions for averaging, and the acquisition state. The instrument executes concatenated commands and queries 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 logic scope 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>.

Some header mnemonics specify one of a range of mnemonics. For example, a channel mnemonic can be CH1, CH2, CH3, or CH4. You use these mnemonics in the command just as you do any other mnemonic. In the command descriptions, this list of choices is abbreviated 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 can specify the channel to use as a mnemonic in the header.

Symbol	Meaning
`CH<x>`	A channel specifier; <x> is either 1, 2, 3, or 4.
`CH<x>`	A channel specifier; <x> is one of 1 through 16.

Group Mnemonics

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

Symbol	Meaning
`GRP<x>`	A group specifier; <x> is one of 1 through 8.

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 part of a group or not (for instance, when specifying a trigger source). Specify such a waveform as follows:

Symbol	Meaning
`<wfm>`	Can be `CH<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 logic scope commands require numeric arguments. The syntax shows the format that the logic scope returns in response to a query. It is also the preferred format when sending the command to the logic scope though it may accept any of the formats. This manual represents these arguments as follows:

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

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

Quoted String Arguments

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

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

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

Follow these rules when you use quoted strings:

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 embedded 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 logic scope commands use a block argument form:

Symbol	Meaning
`<NZDig>`	A nonzero 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.

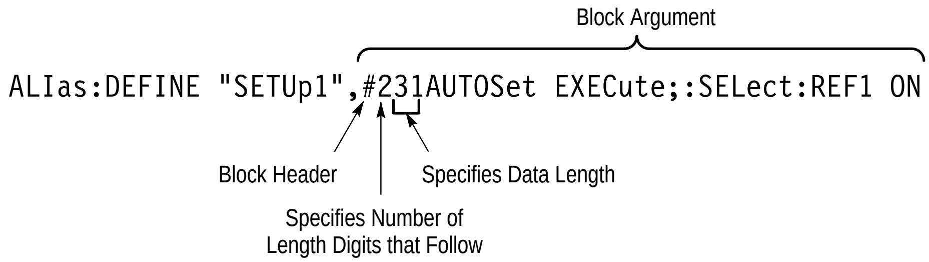

Block Argument
ALIas:DEFINE "SETUp1",#231AUTOSet EXECute;:SELect:REF1 ON
Block Header
Specifies Data Length
Specifies Number of Length Digits that Follow

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.

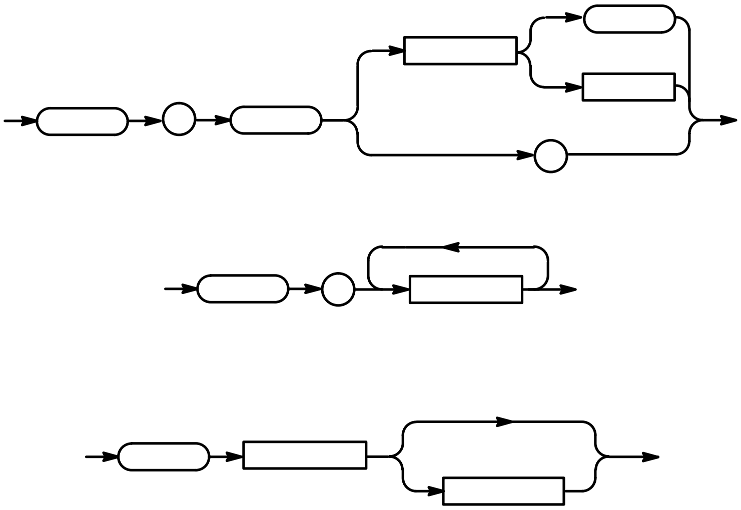


Figure 2–3: Typical Syntax Diagrams

Command Groups

This section lists the TLS 216 Logic Scope commands in two ways. It first lists commands by functional groups. It then lists them alphabetically. (See Figure 2–4.) The functional group list, with brief command descriptions, starts below. The alphabetical list, with more detail on each command, starts on page 2–41.

The logic scope GPIB interface conforms to Tektronix standard codes and formats and IEEE Std 488.2-1987 except where noted.

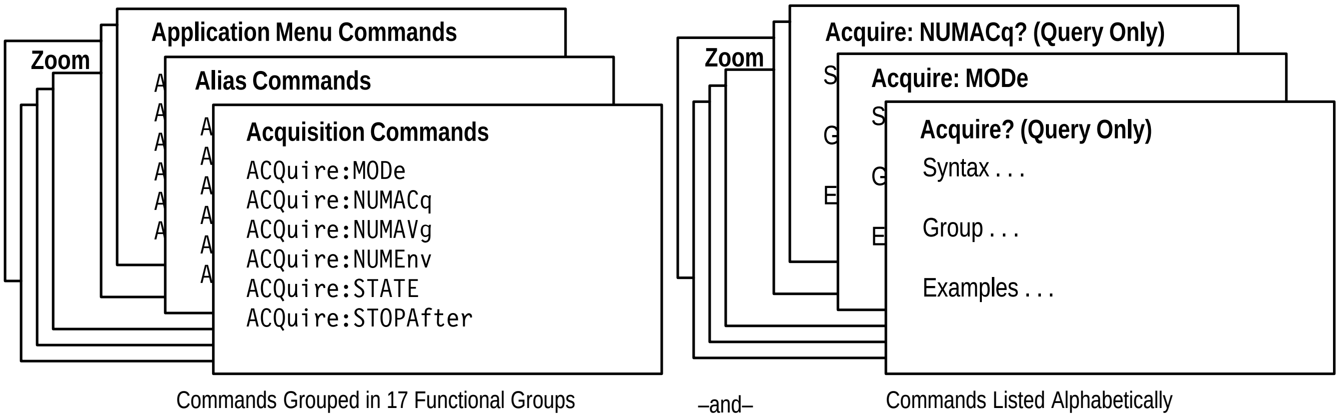


Figure 2–4: Functional List and an Alphabetical List of Commands

Acquisition Commands

Acquisition commands affect waveform acquisition. These commands control mode, averaging, enveloping, and single-waveform acquisition. (Persistence controls are in the section *Display Commands* on page 2–17.)

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

Table 2–4: Acquisition Commands (Cont.)

Header	Description
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 logic scope displays an Application menu and a user presses a front-panel button, the logic scope 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 built-in self-calibration and diagnostic routines of the logic scope. 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?`	Return 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:HBArs:UNIts`	Set H bar units
`CURSor:MODe`	Set cursor tracking mode
`CURSor:PAIred`	Position paired cursors; return paired cursor settings
`CURSor:PAIred:HDELTA?`	Return horizontal distance between 1st and 2nd paired cursors.
`CURSor:PAIred:HPOS1?`	Return horizontal position of 1st paired cursor
`CURSor:PAIred:HPOS2?`	Return 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?`	Select active paired cursor
`CURSor:PAIred:UNITS?`	Return base units only
`CURSor:PAIred:VDELTA?`	Return vertical distance between 1st and 2nd paired cursors
`CURSor:VBArs`	Position vertical bar cursors
`Cursor:VBArs:BINARY<x>?`	Return a 16 character string indicating logic levels at the cursor position
`CURSor:VBArs:DELTa?`	Return 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, the instrument acquires and transmits waveforms but does not display them. The update rate is much faster when waveform display is off.

Table 2–9: Display Commands

Header	Description
`CLEARMenu`	Clear menus from display
`DISplay?`	Return display settings
`DISplay:CLOCk`	Control the display of the date/time stamp
`DISplay:COLOr?`	Return display color settings
`DISplay:COLOr:CONTRast`	Control the collision contrast
`DISplay:COLOr:MAP?`	Return color for a group or reference group
`DISplay:COLOr:MAP:<item>?`	Return color for a specific group or reference group
`DISplay:COLOr:MAP:<item>:TO`	Set color for group or reference group to specific color index
`DISplay:COLOr:PALEtte?`	Return palette settings
`DISplay:COLOr:PALEtte:PERSistence`	Set current persistence palette to a preset persistence palette
`DISplay:COLOr:PALEtte:REGular`	Set current palette to a preset palette
`DISplay:COLOr:PALEtte:RESETAll`	Reset all palettes to their factory default settings
`DISplay:COLOr:PALEtte:<palette name>:RESET`	Reset a selected palette to its factory default settings
`DISplay:COLOr:PALEtte:<palette name>:<item name>`	Set the color of a selected item on a selected palette
`DISplay:FILTer`	Displayed data interpolation
`DISplay:FORMat`	Set display type to voltage versus time; returns YT only
`DISplay:GRAticule`	Graticule style
`DISplay:INTENSITy?`	Return intensity settings
`DISplay:INTENSITy:TEXt`	Set text brightness
`DISplay:INTENSITy:WAVEform`	Set waveform brightness
`DISplay:PERSistence`	Set variable persistence decay time

Table 2–9: Display Commands (Cont.)

Header	Description
DISplay:STYle	Set waveform dots, vector, infinite or variable persistence
DISplay:TRIGT	Control display of the trigger indicator on screen
DISplay:TRIGBar	Control display of the trigger bars 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

File System Commands

File system commands help you use the built-in 3.5″ floppy disk drive.

Table 2–10: File System Commands

Header	Description
FILESystem?	Return file system settings
FILESystem:COPy	Copy file to new file
FILESystem:CWD	Set directory path
FILESystem:DELEte	Delete named file
FILESystem:DELWarn	Set front panel delete warning
FILESystem:DIR	Return files and directories in CWD
FILESystem:FORMat	Format named drive
FILESystem:FREESpace	Return free space on current drive
FILESystem:MKDir	Make new directory
FILESystem:OVERWrite	Set file-overwrite protection
FILESystem:PRInt	Print file to port
FILESystem:REName	Assign new name to file
FILESystem:RMDir	Delete named directory

Hardcopy Commands

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

Table 2–11: Hardcopy Commands

Header	Description
`HARDCopy`	Start or terminate hardcopy
`HARDCopy:FILEName`	Select file to send hard copy data to
`HARDCopy:FORMat`	Set hardcopy output format
`HARDCopy:LAYout`	Set hardcopy orientation
`HARDCopy:PALEtte`	Select palette to use when making hard copy
`HARDCopy:PORT`	Set hardcopy port for output

Horizontal Commands

Horizontal commands control the time bases of the logic scope. 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 oscilloscopes.

Table 2–12: Horizontal Commands

Header	Description
`HORizontal?`	Return horizontal settings
`HORizontal:DELay?`	Return delay time base settings
`HORizontal:DELay:MODe`	Set delay time base mode
`HORizontal:DELay:SCAle`	Set delay time base time/division
`HORizontal:DELay:SECdiv`	Same as HORizontal:DELay:SCAle
`HORizontal:DELay:TIMe`	Set delay time
`HORizontal:DELay:TIMe?`	Return delay time parameters
`HORizontal:DELay:TIMe: RUNSAfter`	Set time to wait in Delay-runs-after-main mode
`HORizontal:DELay:TIMe: TRIGAfter`	Set time to wait in Delay-runs-after-trigger mode
`HORizontal:FITtoscreen`	Set up Waveform Compress

Table 2–12: Horizontal Commands (Cont.)

Header	Description
HORizontal:MAIn?	Return main time/division
HORizontal:MAIn:SCAle	Set main time base time/division
HORizontal:MAIn:SECdiv	Same as HORizontal:MAIn:SCAle
HORizontal:MODe	Turn delay time base on or off
HORizontal:POSition	Set portion of waveform to display
HORizontal:RECOrdlength	Set 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	Set main time base trigger position

Measurement Commands

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

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.

Groups vs. Channels

Although the logic scope displays waveforms as groups, it takes automated measurements on the selected channel within the group. Keep in mind, however, that the selected channel must be part of a group and that group must be selected. The instrument can only make measurements on references if the reference is displayed. See *Vertical Commands* on page 2–34 for defining waveform groups and which channels they contain.

Immediate Measurements

In addition to the four displayed measurements, the measurement commands let you specify a fifth measurement, IMMed. An immediate measurement has no front-panel equivalent. The instrument never displays immediate measurements. 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.

Table 2–13: Measurement Commands

Header	Description
MEASUrement?	Return all measurement parameters
MEASUrement:CLEARSNapshot	Remove measurement snapshot
MEASUrement:GATING	Set or query measurement gating
MEASUrement:IMMed?	Return immediate measurement parameters.
MEASUrement:IMMed:CLOCk?	Return immediate measurement clock parameters
MEASUrement:IMMed:CLOCk: DIREction	Set the clock source direction for a group measurement
MEASUrement:IMMed:CLOCk: EDGE	Set the clock source edge for a group measurement
MEASUrement:IMMed:CLOCk: SOUrce	Set the clock source for a group measurement
MEASUrement:IMMed:DELay?	Return immediate delay measurement settings
MEASUrement:IMMed:DELay: DIREction	Select search direction for delay measurements
MEASUrement:IMMed:DELay: EDGE1	Select waveform edge for delay measurements
MEASUrement:IMMed:DELay: EDGE2	Select waveform edge for delay measurements
MEASUrement:IMMed:SOURCE1	Set the measurement source (channel)
MEASUrement:IMMed:SOURCE2	Set the second (channel) measurement source (delay or “to” channel)
MEASUrement:IMMed:TYPe	Identify 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>:CLOCk?	Return current clock parameters for the measurement specified by MEAS<x>
MEASUrement:MEAS<x>:CLOCk: DIRection	Set the clock source direction for a group measurement
MEASUrement:MEAS<x>:CLOCk: EDGE	Set the clock source edge for a group measurement
MEASUrement:MEAS<x>:CLOCk: SOUrce	Set the clock source for a group measurement
MEASUrement:MEAS<x>:DELay?	Return delay measurement parameters
MEASUrement:MEAS<x>:DELay: DIRection	Set search direction for delay measurements

Table 2–13: Measurement Commands (Cont.)

Header	Description
MEASUrement:MEAS<x>:DELay:EDGE1	Select waveform edge for delay measurements
MEASUrement:MEAS<x>:DELay:EDGE2	Select waveform edge for delay measurements
MEASUrement:MEAS<x>:SOURCE1	Set the measurement source (channel)
MEASUrement:MEAS<x>:SOURCE2	Set the second (channel) measurement source (delay or “to” channel)
MEASUrement:MEAS<x>:STATE	Turn measurement display on or off
MEASUrement:MEAS<x>:TYPe	The measurement to be taken
MEASUrement:MEAS<x>:UNIts?	Return units to use for measurement
MEASUrement:MEAS<x>:VALue?	Return measurement result
MEASUrement:METHod	Select method for calculating reference levels
MEASUrement:REFLevel?	Return reference levels
MEASUrement:REFLevel:ABSolute?	Return the absolute reference level settings.
MEASUrement:REFLevel:ABSolute:HIGH	Set top level for rise time (90% level)
MEASUrement:REFLevel:ABSolute:LOW	Set low level for rise time (10% level)
MEASUrement:REFLevel:ABSolute:MID	Set mid level for measurements
MEASUrement:REFLevel:ABSolute:MID2	Set mid level for delay measurements
MEASUrement:REFLevel:METHod	Select method to assign HIGH and LOW levels: either % or absolute volts
MEASUrement:REFLevel:PERCent?	Return the Percent Settings reference level
MEASUrement:REFLevel:PERCent:HIGH	Set top level for rise time (90% level)
MEASUrement:REFLevel:PERCent:LOW	Set low level for rise time (10% level)
MEASUrement:REFLevel:PERCent:MID	Set mid level for measurements
MEASUrement:REFLevel:PERCent:MID2	Set Mid level for delay measurements
MEASUrement:SNAPShot	Display 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
ID?	Return instrument identification and firmware information
*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
SET?	Same as *LRN?
TEKSecure	Initialize waveforms and setups
TIMe	Set time
*TRG	Perform Group Execute Trigger (GET)
*TST?	Self-test
UNLock	Unlock front panel (local lockout)
VERBose	Return full command name or minimum spellings with query

RS232 Commands

RS232 commands allow you to use the serial communications port.

Table 2–15: RS232 Commands

Header	Description
`RS232?`	Return RS232 parameters
`RS232:BAUd`	Set baud rate
`RS232:HARDFlagging`	Set hard flagging
`RS232:PARity`	Set parity
`RS232:SOFTFlagging`	Set soft flagging
`RS232:STOPBits`	Set # of stop bits

Save and Recall Commands

Save and Recall commands allow you to store and retrieve internal waveform groups and control setups.

- When you "save a group," the logic scope saves all the waveforms in the specified group to the specified reference group (one of four) or to a mass storage file in the logic scope file system.
- When you then "recall a group," the logic scope places a copy of the waveform group stored in the file system media into the specified reference group (one of four).
- When you "display a saved group," you turn on the reference group that contains it using the SELect:<grp> command (see *Vertical Commands* on page 2–34). To display a group saved to the file system, you must first recall it to a reference group (using the method described previously), and then use the SELect:<grp> command to turn on the group as was just described. You can only perform measurements and waveform transfers on displayed reference groups.
- When you "save a setting," you save all the settings of the logic scope.
- When you then "recall a setting," the logic scope restores itself to the state it was in when you last saved that setting.

You may substitute the header mnemonic WAVEFORM for GROUP (as an alias) in all Save and Recall commands to follow.

Table 2–16: Save and Recall Commands

Header	Description
ALLOcate?	Return number of allocated and nonallocated data points
ALLOcate:GROUP?	Return number of allocated data points: Length, First Channel, Last Channel
ALLOcate:GROUP:FREE?	Return number of nonallocated data points
ALLOcate:GROUP:REF<z> <length>, <firstchan>, <lastchan>?	Specify the number of allocated data points and the channels to be used for a new reference
DELEte:GROUP	Delete stored waveform group
DELEte:SETUp	Delete stored setup
*RCL	Recall setting
RECAll:GROUP	Recall saved waveform group
RECAll:SETUp	Recall saved instrument setting
*SAV	Save setting
SAVe:GROUP	Save waveform group
SAVe:SETUp	Save instrument setting
SAVe:WAVEFORM	Save designated channel waveform

Status and Error Commands

Table 2–17 lists the status and error commands the logic scope supports. These commands let you determine the status of the logic scope, and control events.

Several commands and queries used with the logic scope 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 2–17: 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

Table 2–17: Status and Error Commands (Cont.)

Header	Description
EVENT?	Return event code
EVMsg?	Return event code and message
EVQty?	Return number of events in queue
*OPC	Operation complete
*PSC	Power-on status clear
*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 logic scope triggering. There are two triggers, main and delayed. Where appropriate, the command set has parallel constructions for each trigger.

You can set the main and delayed triggers to one of five types: edge, pulse, pattern, state, and sequence. Edge triggering is the most familiar type.

Edge triggering lets you display a waveform at or near the point where the signal passes through a voltage level of your choosing.

Pulse triggering lets the logic scope trigger whenever it detects a pulse of a certain width.

Pattern triggering lets you logically combine signals from one or more channels into a pattern. The logic scope triggers when the pattern meets boolean and timing conditions.

Sequence triggering lets you logically combine signals from one or more channels into a pattern. The logic scope triggers when a sequence of two patterns meet their boolean and timing conditions.

State triggering lets you logically combine signals from one or more channels and provides a signal from a single channel to use as a clock. The logic scope triggers when signals meet boolean conditions at the time they are clocked.

Table 2–18: Trigger Commands

Header	Description
`TRIGger`	Force trigger event; return parameters
`TRIGger:DELay`	Set the delay trigger level to 50%
`TRIGger:DELay:BY`	Set delay by time or events
`TRIGger:DELay:EDGE?`	Return parameters set for delay edge trigger
`TRIGger:DELay:EDGE:SLOpe`	Set slope for the delay edge trigger
`TRIGger:DELay:EDGE:SOUrce`	Set source for delay trigger source
`TRIGger:DELay:EVENTS?`	Return delay trigger event parameters
`TRIGger:DELay:EVENTS:COUNt`	Set events count for delay by events
`TRIGger:DELay:PATtern?`	Return all delayed pattern trigger settings
`TRIGger:DELay:PATtern:AUXiliary`	Identify the logic level expected for Auxiliary Trigger in the delay trigger pattern: H (High), L (Low), or X (Don't Care)
`TRIGger:DELay:PATtern:CH<x>`	Identify the logic level expected for Ch x in the delay pattern trigger: H (High), L (Low), or X (Don't Care)
`TRIGger:DELay:PATtern:INPut`	Select the decimal numbers which, when converted to binary, comprise the delay trigger pattern and mask
`TRIGger:DELay:PATtern:LIMIT?`	Return the delay time-qualified trigger pattern parameters.
`TRIGger:DELay:PATtern:LIMIT: WHEN`	Identify the triggering criteria for the time-qualified delay pattern trigger: Less Than, More Than, Time Out
`TRIGger:DELay:PATtern:LIMIT: WIDth`	Set pattern duration for delay pattern trigger
`TRIGger:DELay:PATtern:LOGIC`	Select the logic applied to the delay trigger pattern: AND, OR, NAND, NOR
`TRIGger:DELay:PATtern:QUALifier`	Select the qualifier for the delay pattern trigger: Limit or Range
`TRIGger:DELay:PATtern:RANge?`	Return the main pattern trigger range parameters.
`TRIGger:DELay:PATtern:RANge: HIGHLimit`	Set the upper time limit of the range for the range-qualified delay pattern trigger
`TRIGger:DELay:PATtern:RANge: LOWLimit`	Set the lower time limit for the range for the range-qualified delay pattern trigger
`TRIGger:DELay:PATtern:RANge: WHEN`	Select the triggering criteria for the range-qualified delay pattern trigger: In Range or Out of Range
`TRIGger:DELay:PULse?`	Return pulse trigger parameters

Table 2–18: Trigger Commands (Cont.)

Header	Description
TRIGger:DELay:PULse:CLAss	Select the pulse trigger class: Width Or Glitch
TRIGger:DELay:PULse:GLItch?	Return glitch trigger parameters
TRIGger:DELay:PULse:GLItch: POLarity	Select glitch polarity: Positive Or Negative
TRIGger:DELay:PULse:GLItch: WHEN	Sets < T, or > ,T time out
TRIGger:DELay:PULse:GLItch: WIDth	Set glitch trigger to discriminate between glitch and valid pulse
TRIGger:DELay:PULse:SOUrce	Identify the pulse trigger channel
TRIGger:DELay:PULse:WIDTH?	Return pulse width settings
TRIGger:DELay:PULse:WIDth: HIGHLimit	Set pulse trigger maximum pulse width
TRIGger:DELay:PULse:WIDth: LOWLimit	Set pulse trigger minimum pulse width
TRIGger:DELay:PULse:WIDth: POLarity	Set pulse trigger: Plus Or Minus
TRIGger:DELay:PULse:WIDth: WHEn	Set pulse trigger when pulse detected or when not detected
TRIGger:DELay:SEQuence?	Return delay sequence settings
TRIGger:DELay:SEQuence:END?	Return delay sequence end settings
TRIGger:DELay:SEQuence:END: AUXiliary	Identify the logic level expected at the Auxiliary Trigger input End Event for delay sequence trigger: H (High), L (Low), or X (Don't Care)
TRIGger:DELay:SEQuence:END: CH<x>	Identity the logic level expected at the Ch x End Event for delay sequence trigger: H (High), L (Low), or X (Don't Care)
TRIGger:DELay:SEQuence:END: INPut	Set the decimal numbers which, when converted to binary, comprise the Stop Event pattern and mask to the delay sequence trigger
TRIGger:DELay:SEQuence:END: LOGic	Set the logic function applied to the main sequence trigger inputs for the end event: AND, OR, NAND, NOR
TRIGger:DELay:SEQuence:LIMIT?	Return delay sequence limit settings
TRIGger:DELay:SEQuence:LIMIT: WHEN	Select the triggering criteria for the time-qualified delay sequence trigger: Less Than, More Than, or Not in Time

Table 2–18: Trigger Commands (Cont.)

Header	Description
TRIGger:DELay:SEQuence:LIMIT:WIDTH	Set the time duration from time qualified delay sequence trigger
TRIGger:DELay:SEQuence:QUAlifier	Select the qualifier for the delay sequence trigger: Limit or Range
TRIGger:DELay:SEQuence:RANge?	Return delay sequence range settings
TRIGger:DELay:SEQuence:RANge:HIGHLimit	Set the upper time limit of the range for the range-qualified delay sequence trigger
TRIGger:DELay:SEQuence:RANge:LOWLimit	Set the lower time limit for the range for the range-qualified delay sequence trigger
TRIGger:DELay:SEQuence:RANge:WHEN	Set the triggering criteria for the range-qualified delay sequence trigger: In Range or Out of Range
TRIGger:DELay:SEQuence:STARt?	Return delay sequence start settings
TRIGger:DELay:SEQuence:STARt:AUXiliary	Set the logic level expected at the Auxiliary input to the Start Event for delay sequence trigger: H (High), L (Low), or X (Don't Care)
TRIGger:DELay:SEQuence:STARt:CH<x>	Set the logic level expected at the Ch x to the Start Event for delay sequence trigger: H (High), L (Low), or X (Don't Care)
TRIGger:DELay:SEQuence:STARt:INPut	Set the decimal number which, when converted to binary, is the Start Event pattern to the delay sequence trigger
TRIGger:DELay:SEQuence:STARt:LOGIC	Select the logic function applied to the delay sequence trigger inputs for the Start Event: AND, OR, NAND, NOR
TRIGger:DELay:STATE?	Return all trigger delay state parameters
TRIGger:DELay:STATE:AUXiliary	Set the level expected at the Auxiliary Trigger input to the delay state trigger: H (High), L (Low), or X (Don't Care)
TRIGger:DELay:STATE:CH<x>	Set the level expected at the Ch x input to the delay state trigger: H (High), L (Low), or X (Don't Care)
TRIGger:DELay:STATE:CLOCk	Identify the channel providing the clock to the delay state trigger
TRIGger:DELay:STATE:INPut	Set the decimal numbers which, when converted to binary, comprise the state expected at the inputs to the delay state trigger and masks
TRIGger:DELay:STATE:LOGIC	Select the logic function applied to the delay state trigger inputs: AND, OR, NAND, NOR

Table 2–18: Trigger Commands (Cont.)

Header	Description
TRIGger:DELay:STATE:SLOpe	Select the clock edge that clocks the delay state trigger: rising or falling
TRIGger:DELay:TIMe	Set time for delay by time
TRIGger:DELay:TYPe	Select the type of delay trigger: Edge, Pulse, Pattern, State, or Sequence
TRIGger:MAin	Set the main trigger level to 50%
TRIGger:MAin:EDGE?	Return parameters set for main edge trigger
TRIGger:MAin:EDGE:SLOpe	Set the slope for the main edge trigger
TRIGger:MAin:EDGE:SOUrce	Set the source for main trigger source
TRIGger:MAIn:HOLDoff?	Return main trigger holdoff value
TRIGger:MAin:HOLdoff:VALue	Set main trigger holdoff value
TRIGger:MAin:LEVel	Set triggering level for main trigger (used with type edge triggers only)
TRIGger:MAIn:MODe	Set main trigger mode
TRIGger:MAin:PATtern?	Return all pattern settings
TRIGger:MAin:PATtern:AUXiliary	Select the logic level expected for Auxiliary Trigger in the main trigger pattern: H (High), L (Low), or X (Don't Care)
TRIGger:MAin:PATtern:CH<x>	Select the logic level expected for Ch x in the main pattern trigger: H (High), L (Low), or X (Don't Care)
TRIGger:MAin:PATtern:INPut	Set the decimal numbers which, when converted to binary, comprise the main trigger pattern and mask
TRIGger:MAin:PATtern:LIMit?	Return all pattern limit parameters
TRIGger:MAin:PATtern:LIMit: WHEN	Select the triggering criteria for the time-qualified main pattern trigger: True for Less Than, True for More Than, or Not in Time
TRIGger:MAin:PATtern:LIMit: WIDth	Set the width (in time) of the main trigger pattern
TRIGger:MAin:PATtern:LOGIC	Select the logic applied to the main trigger pattern: AND, OR, NAND, NOR
TRIGger:MAin:PATtern:QUAlifier	Select the qualifier for the main pattern trigger: Limit or Range
TRIGger:MAin:PATtern:RANge?	Return all pattern range parameters
TRIGger:MAin:PATtern:RANge: HIGHLimit	Set the upper time limit of the range for the range-qualified main pattern trigger

Table 2–18: Trigger Commands (Cont.)

Header	Description
`TRIGger:MAin:PATtern:RANge:LOWLimit`	Set the lower time limit for the range for the range-qualified main pattern trigger
`TRIGger:MAin:PATtern:RANge:WHEN`	Select the triggering criteria for the range-qualified main pattern trigger: In Range or Out of Range
`TRIGger:MAin:PULse?`	Return pulse trigger parameters
`TRIGger:MAin:PULse:CLAss`	Select the pulse trigger class: Width Or Glitch
`TRIGger:MAin:PULse:GLItch?`	Return glitch trigger parameters
`TRIGger:MAin:PULse:GLItch:POLarity`	Set glitch polarity: positive or negative
`TRIGger:MAin:PULse:GLItch:WHEN`	Select glitch triggering criteria: TRUE < Time, TRUE > Time, or Time Out
`TRIGger:MAin:PULse:GLItch:WIDth`	Set glitch trigger width
`TRIGger:MAin:PULse:SOUrce`	Select pulse trigger channel
`TRIGger:MAin:PULse:WIDth?`	Return all pulse width parameters
`TRIGger:MAin:PULse:WIDth:HIGHLimit`	Set pulse trigger maximum pulse width
`TRIGger:MAin:PULse:WIDth:LOWLimit`	Set pulse trigger minimum pulse width
`TRIGger:MAin:PULse:WIDth:POLarity`	Select pulse trigger positive or negative
`TRIGger:MAin:PULse:WIDth:WHEn`	Select pulse trigger detection mode
`TRIGger:MAin:SEQuence?`	Set time limit for the time-qualified main sequence trigger
`TRIGger:MAin:SEQuence:END?`	Return all main sequence end parameters
`TRIGger:MAin:SEQuence:END:AUXiliary`	Select the logic level expected at the Auxiliary Trigger input End Event for main sequence trigger: H (High), L (Low), or X (Don't Care)
`TRIGger:MAin:SEQuence:END:CH<x>`	Select the logic level expected at the Ch x End Event for main sequence trigger: H (High), L (Low), or X (Don't Care)
`TRIGger:MAin:SEQuence:END:INPut`	Set the decimal number which, when converted to binary, is the Stop Event pattern to the main sequence trigger
`TRIGger:MAin:SEQuence:END:LOGIC`	Select the logic function applied to the main sequence trigger inputs for the Start Event: AND, OR, NAND, NOR

Table 2–18: Trigger Commands (Cont.)

Header	Description
TRIGger:MAin:SEQuence:LIMIT?	Return all main sequence limit parameters
TRIGger:MAin:SEQuence:LIMIT:WHEN	Set the triggering criteria for the time-qualified main sequence trigger: Less Than, More Than, or Not in Time
TRIGger:MAin:SEQuence:LIMIT:WIDth	Select the time limit for the Trigger When definitions of Less Than, More Than, or Time Out
TRIGger:MAin:SEQuence:QUAlifier	Select the qualifier for the main sequence trigger: Limit or Range
TRIGger:MAin:SEQuence:RANge?	Return all main sequence Range parameters
TRIGger:MAin:SEQuence:RANge:HIGHLimit	Set the upper time limit of the range for the range-qualified main sequence trigger
TRIGger:MAin:SEQuence:RANge:LOWLimit	Set the lower time limit for the range for the range-qualified main sequence trigger
TRIGger:MAin:SEQuence:RANge:WHEN	Select the triggering criteria for the range-qualified main sequence trigger: In Range Or Out Of Range
TRIGger:MAin:SEQuence:STARt?	Return all main sequence Start Trigger parameters
TRIGger:MAin:SEQuence:STARt:AUXiliary	Select the logic level expected at the Auxiliary input to the Start Event for main sequence trigger: H (High), L (Low), or X (Don’t Care)
TRIGger:MAin:SEQuence:STARt:CH<x>	Set the logic level expected at the Ch x to the Start Event for main sequence trigger: H (High), L (Low), or X (Don’t Care)
TRIGger:MAin:SEQuence:STARt:INPut	Set the decimal number which, when converted to binary, is the Start Event pattern to the main sequence trigger
TRIGger:MAin:SEQuence:STARt:LOGIC	Select the logic function applied to the main sequence trigger inputs for the Start Event: AND or OR
TRIGger:MAin:STATE?	Return all main state trigger parameters
TRIGger:MAin:STATE:AUXiliary	Set the level expected at the Auxiliary Trigger input to the main state trigger: H (High), L (Low), or X (Don’t Care)
TRIGger:MAin:STATE:CH<x>	Set the level expected at the Ch x input to the main state trigger: H (High), L (Low), or X (Don’t Care)
TRIGger:MAin:STATE:CLOCk	Select the channel providing the clock to the main state trigger

Table 2–18: Trigger Commands (Cont.)

Header	Description
`TRIGger:MAin:STATE:INPut`	Set the decimal number which, when converted to binary, is the state expected at the inputs to the main state trigger
`TRIGger:MAin:STATE:LOGIC`	Select the logic function applied to the main state trigger inputs: AND, OR, NAND, NOR
`TRIGger:MAin:STATE:SLOpe`	Select the clock edge that clocks the main state trigger: Rising Or Falling
`TRIGger:MAin:TYPe`	Set main trigger type: Edge, Pulse, Pattern, State, or Sequence
`TRIGger:STATE?`	Return trigger system status
`TRIGger:THRESHold?`	Return the trigger thresholds of all channels (not effective for Edge triggering)
`TRIGger:THRESHold:CH<x>`	Select the channel trigger threshold for +ECL, –ECL, TTL, CMOS, or value
`TRIGger:THRESHold:Group<x>`	Set the group trigger threshold for +ECL, –ECL, TTL, CMOS, or value
`TRIGger:THRESHold:ALL`	Set all trigger thresholds (groups and channels) for +ECL, –ECL, TTL, CMOS, or value

Vertical Commands

Vertical commands control the display of groups, of their channels and waveforms, and or reference waveforms.

Groups vs. Channels

The logic scope stores and displays waveforms in groups, which contain one or more channels each of which can contain one waveform. Use vertical commands to vertically position, offset, scale, and otherwise manipulate groups of waveforms.

You *must* include a channel in a group to acquire (sample and digitize) its waveform; channels not included in any group (or included in a group that is off) can only be used as trigger sources.

Table 2–19: Vertical Commands

Header	Description
`CH<x>?`	Return all channel vertical parameters
`CH<x>:BANdwidth?`	Return specified channel bandwidth
`CH<x>:COUPling?`	Return specified channel coupling mode
`CH<x>:IMPedance?`	Return specified channel impedance
`CH<x>:OFFSet?`	Return specified channel offset
`CH<x>:POSition?`	Return specified channel position
`CH<x>:SCAle?`	Return specified channel scale
`GROUPSET`	Set channel contents for *all* groups
`GROUP<x>?`	Return group parameters
`GROUP<x>:BANdwidth`	Set group bandwidth
`GROUP<x>:DELETE`	Delete group
`GROUP<x>:DISplay`	Set display mode for group
`GROUP<x>:FIRst`	Set first channel in group
`GROUP<x>:HEIght`	Change bus and timing group waveform heights
`GROUP<x>:LAST`	Set last channel in the group
`GROUP<x>:OFFSet`	Set group offset
`GROUP<x>:POSition`	Set group position
`GROUP<x>:SCAle`	Set or return the group volts/div
`GROUP<x>:SELect`	Set selected channel in group
`GROUP<x>:VHIgh`	Set the upper threshold voltage for determining Bus Mode displays

Table 2–19: Vertical Commands (Cont.)

Header	Description
GROUP<x>:VLOw	Set the lower threshold voltage for determining Bus mode displays
GROUP<x>:VOLts	Same as GROUP<x>:SCAle
REF<x>?	Return vertical parameters
REF<x>:BANdwidth?	Return group bandwidth
REF<x>:DISplay	Set display mode for the group
REF<x>:FIRst?	Return first channel in the group
REF<x>:HEIght	Set bus and timing group waveform heights
REF<x>:LAST?	Return last channel in the group
REF<x>:OFFSet?	Return group offset
REF<x>:POSition	Set group position
REF<x>:SCAle?	Return group volts/div
REF<x>:SELect	Set selected channel in the group
REF<x>:VHIgh	Set the upper threshold voltage for determining Bus mode displays
REF<x>:VLOw	Set the lower threshold voltage for determining Bus mode displays
REF<x>:VOLts?	Return Group volts/div
SELect?	Return selected waveform group
SELect:GROUP<x>	Set selected waveform group
SELect:CONTROl	Set front panel group selector
SELect:REF<x>	Set selected reference group

Waveform Group Commands

Waveform commands let you transfer waveform data points to and from the logic scope. 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 or ENVelope mode uses one 8-bit byte per waveform data point. Data acquired in AVErage mode uses two 8-bit bytes per point. For more information on the acquisition modes see the ACQuire: MODe command on page 2–41.

The DATa:WIDth command lets you specify the number of bytes per data point when transferring data to and from the logic scope. 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 logic scope can transfer waveform data in either ASCII or binary format. You specify the format with the DATa:ENCdg command.

ASCII Data. ASCII data is represented by 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. 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 logic scope, 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 logic scope 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 logic scope, 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 logic scope. You can transfer out multiple waveforms at one time by specifying more than one source.

You may transfer waveforms one at a time into the logic scope. Waveforms sent to the logic scope are always stored in one of the 16 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. Waveforms must be selected for display to be transferred.

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 commands starting on page 2–322 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 Logic Scope

You can transfer waveforms from the logic scope to an external controller using the following sequence:

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

Transferring Waveform Data to the Logic Scope

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

1. Specify waveform reference memory using DATa:DESTination.
2. Specify the memory size for the reference location (entered in Step 1) using the ALLOcate:WAVEFORM:REF<x> command.
3. Specify the waveform data format using DATa:ENCdg.
4. Specify the number of bytes per data point using DATa:WIDth.
5. Specify first data point in the waveform record using DATa:STARt.
6. Set the Ref Display type to other than Off.
7. Transfer waveform preamble information using WFMPRe:<wfm>.
8. Transfer waveform data to the logic scope using CURVe.

Table 2–20: Waveform Commands

Header	Description
CURVe	Transfer waveform data
DATa	Set waveform data format and location
DATa:DESTination	Set destination to the indicated channel
DATa:ENCdg	Select waveform data encoding method
DATa:SOUrce	Select source of CURVe? data
DATa:STARt	Set starting point in waveform transfer
DATa:STOP	Set ending point in waveform transfer
DATa:TARget	Set the destination to the indicated channel
DATa:WIDth	Set byte width of waveform points
WAVFrm?	Return waveform preamble and data
WFMPre?	Return waveform format data
WFMPre:BIT_Nr	Set preamble bit width of waveform points
WFMPre:BN_Fmt	Set preamble binary encoding type
WFMPre:BYT_Nr	Set preamble byte width of waveform points
WFMPre:BYT_Or	Set preamble byte order of waveform points
WFMPre:CRVchk	Set preamble checksum of waveform points
WFMPre:ENCdg	Set preamble encoding method
WFMPre:NR_Pt	Set number of points in the curve
WFMPre:PT_Fmt	Set format of curve points
WFMPre:PT_Off	Set trigger position
WFMPre:WFId	Set curve identifier
WFMPre:XINcr	Set horizontal sampling interval
WFMPre:XMUlt	Set horizontal scale factor
WFMPre:XOFf	Set horizontal offset
WFMPre:XUNit	Set horizontal units
WFMPre:XZEro	Set horizontal origin offset
WFMPre:YMUlt	Set vertical scale factor
WFMPre:YOFf	Set vertical offset
WFMPre:YUNit	Set vertical units
WFMPre:YZEro	Set offset voltage
WFMPre:ZMUlt	Set Z-axis scale factor
WFMPre:ZOFf	Set Z-axis offset
WFMPre:ZUNit	Set Z-axis units

Table 2–20: Waveform Commands (Cont.)

Header	Description
WFMPre:ZZEro	Set Z-axis origin offset
WFMPre:<wfm>?	Set number of points in the curve
WFMPre:<wfm>:NR_Pt	Set number of points in the curve
WFMPre:<wfm>:PT_Fmt	Set format of curve points
WFMPre:<wfm>:PT_Off	Set trigger position
WFMPre:<wfm>:WFId	Set curve identifier
WFMPre:<wfm>:XINcr	Set horizontal sampling interval
WFMPre:<wfm>:XUNit	Set horizontal units
WFMPre:<wfm>:YMUlt	Set vertical scale factor
WFMPre:<wfm>:YOFf	Set vertical offset
WFMPre:<wfm>:YUNit	Set vertical units
WFMPre:<wfm>:YZEro	Set 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 2–21: Zoom Commands

Header	Description
ZOOm	Reset zoom parameters to defaults
ZOOm:HORizontal?	Return horizontal zoom parameters
ZOOm:HORizontal:LOCk	Set horizontal zoom lock
ZOOm:HORizontal:POSition	Set horizontal zoom position
ZOOm:HORizontal:SCAle	Set horizontal zoom scale
ZOOm:STATE	Set turn zoom mode On Or Off
ZOOm:VERTical?	Set return vertical zoom parameters
ZOOm:VERTical:POSition	Set vertical zoom position
ZOOm:VERTical:SCAle	Set 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.

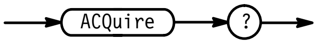
This manual fully spells out headers, mnemonics, and arguments with the minimal spelling shown in upper case. For example, to use the abbreviated form of the ACQuire:MODe command just type ACQ:MOD.

ACQuire? (Query Only)

Returns all current acquisition parameters.

Group Acquisition

Syntax `ACQuire?`



Examples `ACQUIRE?`

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

ACQuire:MODe

Sets or queries the acquisition mode of the logic scope. 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 logic scope 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 the ACQuire:MODe command, determines how the final value of the acquisition interval is generated from the many data samples.

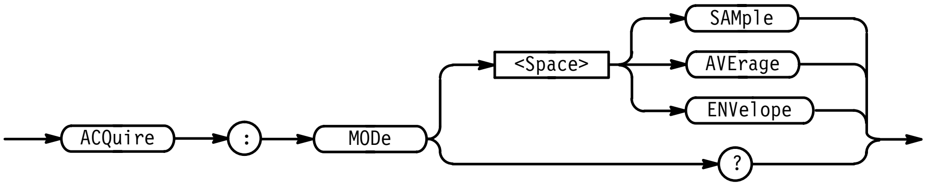
Group Acquisition

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

Syntax

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

```
ACQuire:MODe?
```



Arguments SAMple specifies that the displayed data point value is simply the first sampled value 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.

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 High and Low 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. The instrument resets this value to zero when any Acquisition, Horizontal, or Vertical arguments that affect the waveform are modified. The maximum number of acquisitions that may be counted is 2^{30}–1. Counting stops when this number is reached. This is the same value displayed in the upper left corner of the screen.

Group

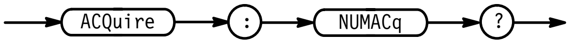
Acquisition

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 or queries the number of waveform acquisitions that make up an averaged waveform. Using this command is equivalent to setting the Average count in the Acquisition Mode side menu.

Group

Acquisition

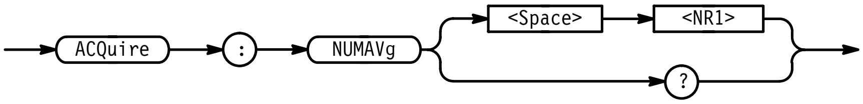
Related Commands

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 there are 75 acquisitions specified for averaging.

ACQuire:NUMEnv

Sets or queries the number of waveform acquisitions comprising an envelope waveform. Using this command is equivalent to setting the Envelope count in the Acquisition Mode side menu.

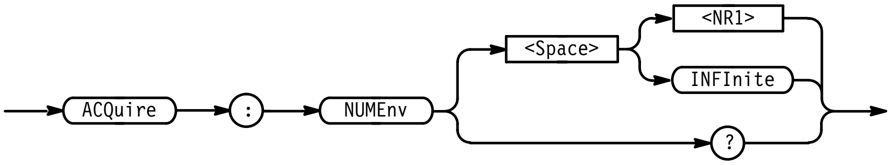
Group Acquisition

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 the number of envelopes to infinity, the logic scope 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:STATE

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

Group

Acquisition

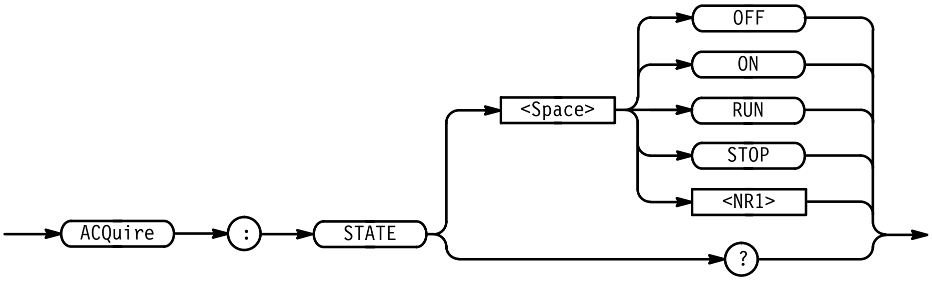
Related Commands

ACQuire:NUMACq? ACQuire:STOPAfter

Syntax

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

```
ACQuire:STATE?
```



Arguments

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

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

Examples

`ACQUIRE:STATE:RUN`
starts acquisition of waveform data and resets the number of acquisitions count (NUMACQ) to zero.

`ACQUIRE:STATE?`
returns either `0` or `1`, depending on whether the acquisition system is running.

ACQuire:STOPAfter

Tells the logic scope when to stop taking acquisitions. Using this command is equivalent to setting Stop After in the Acquire menu.

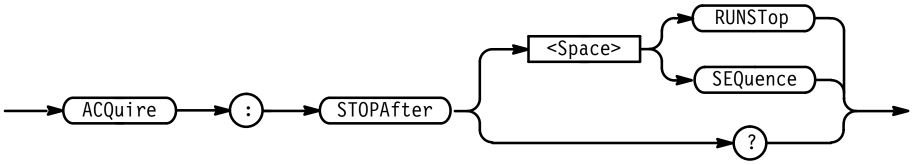
Group Acquisition

Related Commands ACQuire:MODe, ACQuire:STATE

Syntax

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

```
ACQuire:STOPAfter?
```



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

`SEQuence` specifies "single sequence" operation, where the logic scope 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 logic scope will stop after digitizing a waveform from a single trigger event. However, if the acquisition mode is set to average 100 waveforms, then the logic scope 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 logic scope is in single sequence mode.

NOTE*. If you set the acquisition system to single sequence, envelope mode, and the number of envelopes to infinity, the logic scope 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

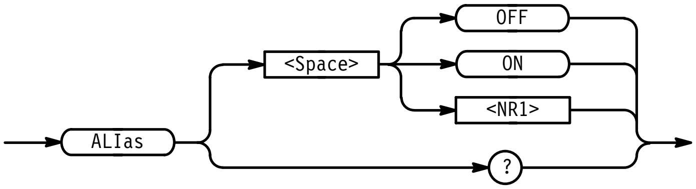
Turns command aliases on or off. This command is identical to the ALIas:STATE command.

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, execution error 102 will be generated: “syntax error, illegal use of alias name”.

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

Examples

```
ALIAS ON
```

turns the alias feature on.

```
ALIAS?
```

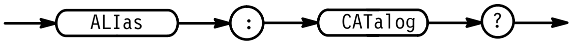
returns 1 when the aliases are on.

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

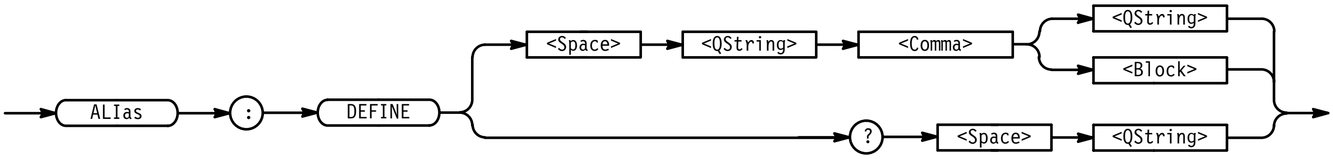
Assigns a sequence of program messages to an alias label. These messages are then substituted for the alias whenever it is received as a command or query provided ALIas:STATE has been turned ON. The ALIas:DEFINE? query returns the definition of a selected alias.

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 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 logic scope.

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

ALIas:DELEte (No Query Form)

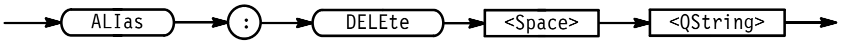
Removes a specified alias. This command is identical to ALIas:DELEte:NAMe.

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)

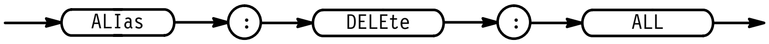
Deletes all existing aliases.

Group

Alias

Syntax

`ALIas:DELEte:ALL`



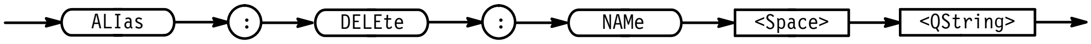
Examples `ALIAS:DELETE:ALL`
deletes all aliases.

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 be removed. 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.

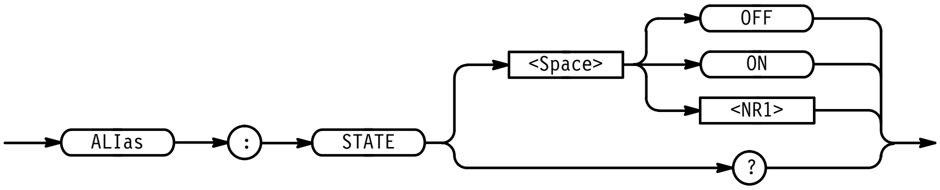
ALIas:STATE

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 the instrument receives a defined alias, it substitutes the specified command sequence and executes it.

Examples

ALIAS:STATE OFF
turns the command alias feature off.

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

ALLEv? (Query Only)

Causes the logic scope to return all events and their messages, and removes the returned events from the Event Queue. Commas separate the messages. Use the *ESR? query to enable the events. For a complete discussion of the use of these registers, see page 3–1. This command is similar to repeatedly sending the *EVMsg? query.

Group

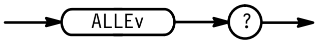
Status and Error

Related Commands

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

Syntax

```
ALLEv?
```



Returns

The event code and message in the following format:

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

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

<Command> is the command that caused the error and may be returned when a command error is detected by the logic scope. 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

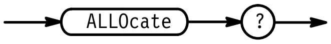
ALLEV?
might return the string :ALLEV 2225,"Measurement error, No waveform to measure; ",420,"Query UNTERMINATED;.

ALLOcate? (Query Only)

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

Group Save and Recall

Syntax `ALLOcate?`



Examples `ALLOCATE?`

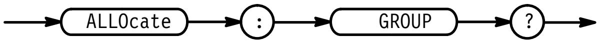
might return `:ALLOCATE:GROUP:REF1 20000;REF2 0;REF3 0;REF4 0`, indicating that all 2,000 data points are allocated to reference memory location 1.

ALLOcate:GROUP? (Query Only)

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

Group Save and Recall

Syntax `ALLOcate:GROUP?`



Examples `ALLOCATE?`

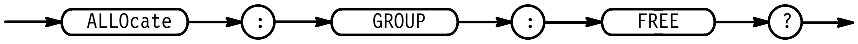
might return `:ALLOCATE:GROUP:REF1 500;REF2 500;REF3 500;REF4 0;`, indicating that 500 data points are allocated to each of the first three reference memory locations.

ALLOcate:GROUP:FREE? (Query Only)

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

Group Save and Recall

Syntax `ALLOcate:GROUP:FREE?`



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

Examples `ALLOCATE:GROUP:FREE?`

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

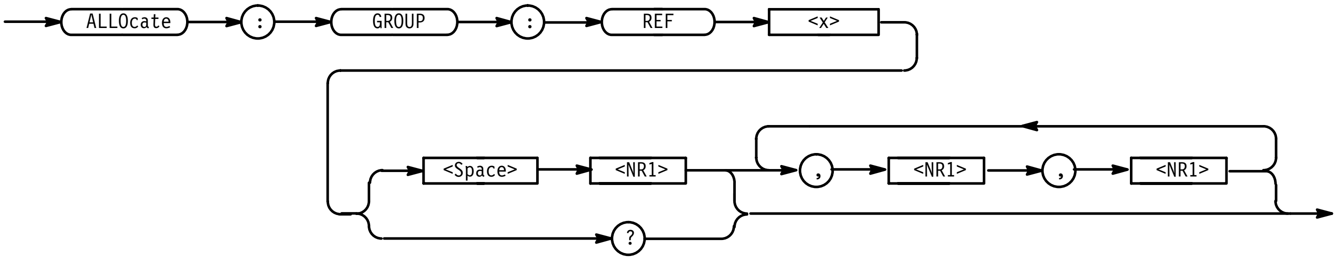
ALLOcate:GROUP:REF<x>

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:GROUP:REF<x> <NR1>[,<NR1>,<NR1>]`

`ALLOcate:GROUP:REF<x>?`



Arguments `<NR1>` $\neq$ 0 specifies the number of data points: 500, 1000, or 2000. 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, 1002 points will allocate 2000 points of data for the reference. The memory size of the four reference locations combined cannot exceed 80,000 data points.

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

The second `<NR1>` specifies the first channel, the third specifies the last channel. If `<NR1>` is not specified, channel 1, and channel 1 are assumed.

Examples

`ALLOCATE:GROUP:REF2 1000`
reserves 1,000 data points for REF2, specifying CH1 for first and last channels.

`ALLOCATE:GROUP:REF4 2000, 10, 14`
reserves 2000 data points for reference group 4. The first channel is 10, the last channel is 14.

`ALLOCATE:GROUP:REF1?`
might return `500`.

APPMenu

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

Group

Application Menu

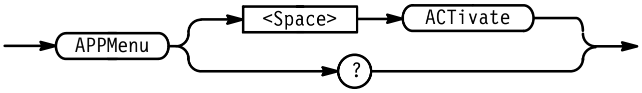
Related Commands

CLEARMenu, *ESR, EVENT?

Syntax

```
APPMenu ACTivate
```

```
APPMenu?
```



Arguments

`ACTivate` displays the Application menu. Use the CLEARMenu command to deactivate the Application menu.

With the Application menu activated, each time a front-panel menu button is pressed an event is generated indicating its identity. See page 3–18 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 3–1 for a complete discussion of the use of these registers.

Examples

`APPMENU ACTIVATE`
displays the application menu.

APPMenu:LABel

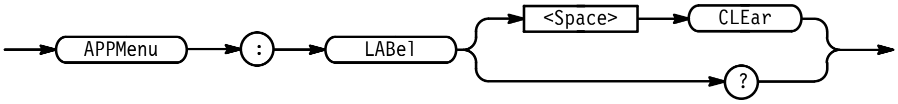
Removes all user-defined Application menu button labels from the display. The APPMenu:LABel? query returns all current label settings.

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?`
might return `:APPMENU:LABEL:BOTTOM1 "";BOTTOM2 "";BOTTOM3 "";BOTTOM4 "";BOTTOM5 "";BOTTOM6 "";BOTTOM7 "";RIGHT1 "";RIGHT2 "";RIGHT3 "";RIGHT4 "";RIGHT5 ""`.

APPMenu:LABel:BOTTOM<x>

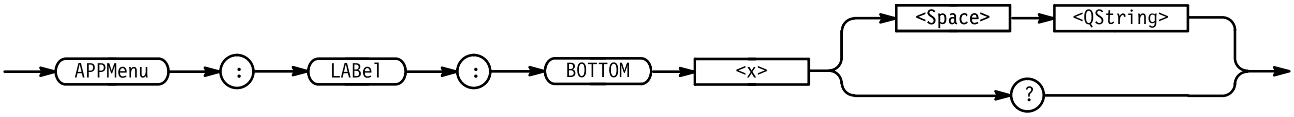
Defines a label for the main menu button specified by <x>. Main menu buttons are located along the bottom of the display and are numbered from 1 to 7, starting with the left-most button.

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 logic scope 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 embedded characters 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–2.

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:BOTTOM3 "SETUP"?`
might return `B3`.

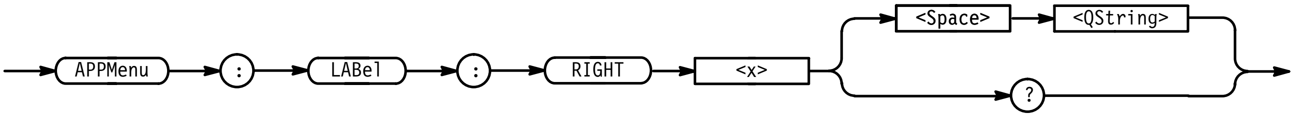
APPMenu:LABel:RIGHT<x>

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 logic scope 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 2–56 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–2.

Examples `APPMENU:LABEL:RIGHT1 "TEST ON"`
displays the label “TEST ON” next to the top button in the side menu.

`APPMENU:LABEL:RIGHT5`
might return `ABORT` if that is the label on the fifth button from the top of the side menu.

APPMenu:TITLe

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

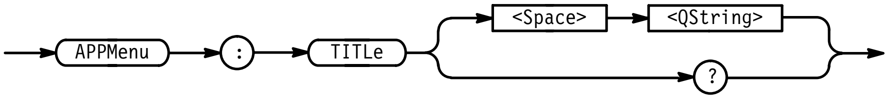
Group Application Menu

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 logic scope Character Chart in Appendix A. The maximum length of the title is 1000 characters. The APPMenu:LABel:BOTTOM<x> command on page 2–56 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–2.

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 logic scope to adjust its vertical, horizontal, and trigger controls to provide a stable display of the selected group. Using this command is equivalent to pressing the front-panel AUTOSET button. For a detailed description of the Autoset function, see *Autosetting the Logic Scope* in Section 3 of the User Manual.

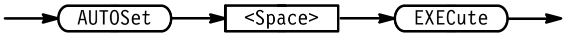
***NOTE**. The AUTOSet command does not return control to the instrument controller until the autoset operation is complete.*

Group

Miscellaneous

Syntax

```
AUTOSet EXECute
```



Arguments `EXECute` autosets the displayed waveform.

BELl (No Query Form)

Activates the audio indicator (beep) of the logic scope.

Group Miscellaneous

Syntax `BELl`

Examples `BELl`
rings the bell.

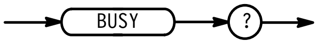
BUSY? (Query Only)

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

Group Status and Error

Related Commands *OPC, *WAI

Syntax `BUSY?`



Returns <NR1> = 0 means that the logic scope is not busy processing a command whose execution time is extensive. These commands are listed in Table 2–22.

<NR1> = 1 means that the logic scope is busy processing one of the commands listed in Table 2–22.

Table 2–22: 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 logic scope to perform an internal self-calibration and return its calibration status.

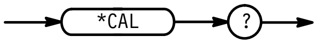
> ***NOTE**. The self-calibration takes approximately 10 minutes to complete. No other commands will be executed during this period.*

Group

Calibration and Diagnostic

Syntax

```
*CAL?
```



Returns

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

<NR1> $\neq$ 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.

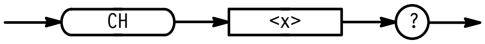
CH<x>? (Query Only)

Returns all the vertical parameters for the specified channel. This query is only made available to maintain compatibility with Tektronix TDS model instruments.

The logic scope only displays channels in user-defined groups. See the query description GROUP<x>? on page 2–126 of this section to query the vertical parameters for any *group* of input channels.

Group Vertical

Syntax CH<x>?



Examples CH1?

might return :CH1:SCALE 10.0E–3;POSITION 0.0E+0;
OFFSET 0.0E+0;COUPLING DC;IMPEDANCE 75;BANDWIDTH FULL.

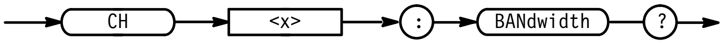
CH<x>:BANdwidth? (Query Only)

Returns the specified channel bandwidth setting. This query is only made available to maintain compatibility with Tektronix TDS model instruments.

The logic scope only displays channels in user-defined groups. See the command description GROUP<x>:BANdwidth on page 2–127 of in this section to set bandwidth for any *group* of input channels.

Group Vertical

Syntax CH<x>:BANdwidth?



Returns FULL for the maximum bandwidth setting.

20 for the 20 MHz bandwidth limit setting.

Examples `CH1:BANDWIDTH?`

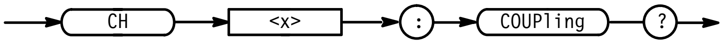
might return `FULL`, which indicates there is no bandwidth limiting on channel 1.

CH<x>:COUPling? (Query Only)

Returns the specified channel input coupling mode. The logic scope only provides DC coupling; therefore this query always returns DC. This query is only made available to maintain compatibility with Tektronix TDS model instruments.

Group Vertical

Syntax `CH<x>:COUPling?`



Returns `DC`

Examples `CH3:COUPLING?`

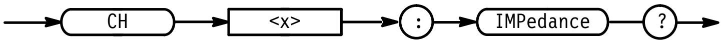
always returns `DC`, indicating that channel 3 is set to DC coupling, the only coupling mode available.

CH<x>:IMPedance? (Query Only)

Returns the specified channel input impedance setting. The logic scope only provides 75 Ω input coupling; therefore this query always returns 75 Ω. This query is only available to maintain compatibility with Tektronix TDS model instruments.

Group Vertical

Syntax `CH<x>:IMPedance?`



Examples `CH3:IMPEDANCE?`

always returns `75`, indicating channel 3 is set to 75 Ω impedance, the only impedance available.

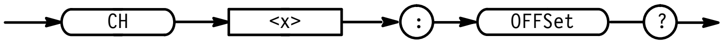
CH<x>:OFFSet? (Query Only)

Returns 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 query is only made available to maintain compatibility with Tektronix TDS model instruments.

The logic scope only offsets channels when included in a user-defined groups. See the command description GROUP<x>:OFFSet on page 2–131 of this section to set offset for any *group* of input channels.

Group Vertical

Syntax `CH<x>:OFFSet?`



Returns `<NR3>`

Examples `CH1:OFFSET?`

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

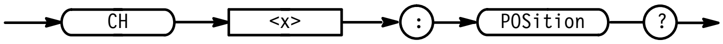
CH<x>:POSition? (Query Only)

Returns the specified channel position. The position value is applied to the signal before digitization. This query is only made available to maintain compatibility with Tektronix TDS model instruments.

The logic scope only positions a channel when it is included in a user-defined group. See the command description GROUP<x>:POSition on page 2–131 of this section to set the vertical positioning for any *group* of input channels.

Group Vertical

Syntax `CH<x>:POSition?`



Returns `<NR3>`

Examples `CH1:POSITION?`
might return `-1.3E+0`, indicating that the current position of channel 1 is –1.3 divisions.

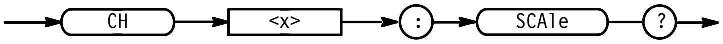
CH<x>:SCAle? (Query Only)

Returns the specified channel scale. This is equivalent to adjusting the front-panel Vertical SCALE knob. This query is only available to maintain compatibility with Tektronix TDS model instruments.

The logic scope only scales a channel when it is included in a user-defined group. See the command description GROUP<x>:SCAle in this section to set the scale for any *group* of input channels.

Group Vertical

Syntax `CH<x>:SCAle?`



Returns `<NR3>`

Examples `CH2:SCALE?`
might return `1.00E+0`, indicating that the current Volts per division setting of channel 2 is 1 Volt.

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 logic scope status data structures.

Group

Status and Error

Related Commands

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

Syntax

```
*CLS
```

The *CLS command clears the following:

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

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

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

CURSor? (Query Only)

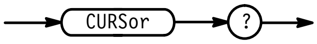
Returns all current cursor settings.

Group

Cursor

Syntax

```
CURSor?
```



Examples CURSOR?

might return :CURSOR:FUNCTION OFF;MODE INDEPENDENT;VBARS:BINARY1 "---------------0";BINARY2 "---------------0";UNITS SECONDS; POSITION1 500.0E-6;POSITION2 4.50E-3;SELECT CURSOR1;:CURSOR: HBARS:UNITS BASE;POSITION1 3.20E+0;POSITION2 -3.20E+0;SELECT CURSOR1;:CURSOR:PAIRED:UNITS BASE;POSITION1 500.0E-6;POSITION2 4.50E-3;SELECT CURSOR1 as the current cursor settings.

CURSor:FUNCtion

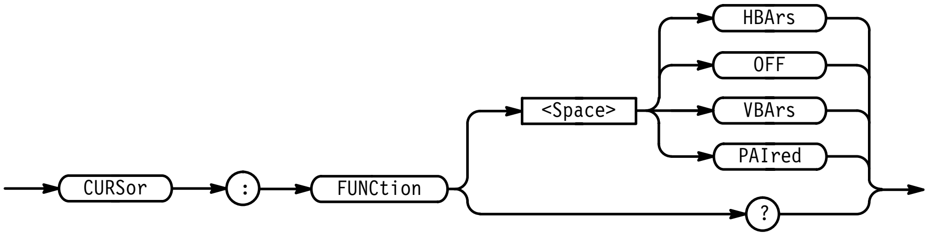
Selects and displays the cursor type. Cursors are attached to the selected waveform. This command is equivalent to setting Function in the Cursor menu.

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.

***NOTE**. HBArs and PAIred settings are not available if the selected group has a display mode of Timing or Bus.*

Examples

`CURSOR:FUNCTION VBARS`
selects vertical bar type cursors.

`CURSOR:FUNCTION?`
might return `HBARS` for horizontal cursors.

CURSor:HBArs? (Query Only)

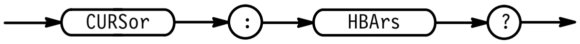
Returns current settings for the horizontal bar cursors.

Group

Cursor

Syntax

`CURSor:HBArs?`



Examples

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

CURSor:HBArs:DELTa? (Query Only)

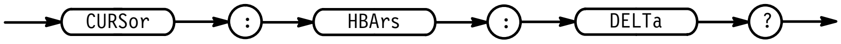
Returns the voltage difference between the two horizontal bar cursors.

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>

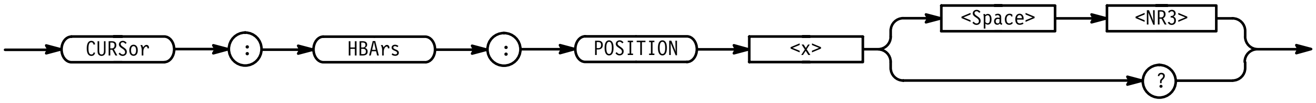
Positions or queries the horizontal bar cursor.

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

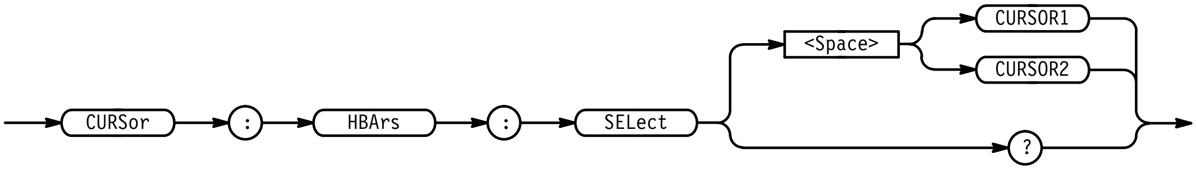
Selects or queries which horizontal bar cursor is active for front-panel control. The active cursor will be displayed as a solid horizontal line and can be moved using the front-panel general purpose knob when the Cursor menu is active. The unselected cursor will be displayed as a dashed horizontal line.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

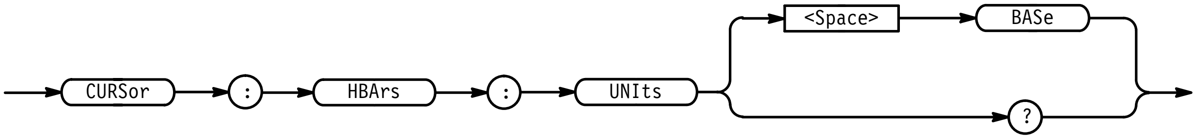
CURSor:HBArs:UNIts

Sets or queries the units for the horizontal bar cursors. The only units the logic scope provides is base (volts). This command is only made available to maintain compatibility with Tektronix TDS model instruments, some of which provide other unit types.

Group Cursor

Syntax CURSor:HBArs:UNIts BASe

CURSor:HBArs:UNIts?



Arguments BASe specifies base as the unit of measure.

Examples CURSOR:HBARS:UNITS BASE
sets the units for the horizontal bar cursors to base.

CURSOR:HBARS:UNITS?
always returns BASe.

CURSor:MODe

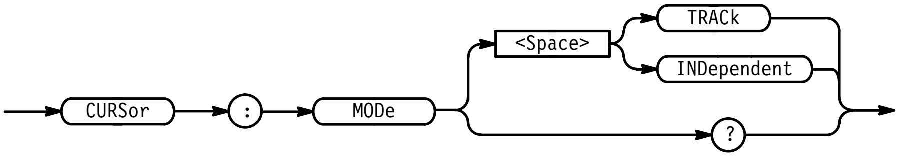
Selects whether the two cursors move together in unison or separately from each other.

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, and is the default mode.

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

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

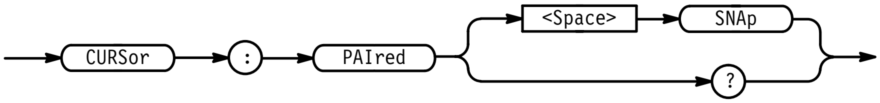
CURSor:PAIred

Sets or queries the paired cursor settings.

Group Cursor

Syntax `CURSor:PAIred SNAp`

`CURSor:PAIred?`



Arguments

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

Examples

CURSOR:PAIRED SNAP
specifies that the cursors positions are the same as the current DATA:STARt and DATA:STOP values.

CURSOR:PAIRED?
might return :CURSOR:PAIRED:UNITS SECONDS;POSITION1 1.00E–6;POSITION2 9.00E–6;SELECT CURSOR2.

CURSor:PAIred:HDELTA? (Query Only)

Queries the H bar (voltage) distance between the first and second paired cursor (the absolute value of the first cursor's vertical position minus the second cursor's vertical position).

Group

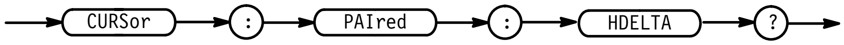
Cursor

Related Commands

CURSor:FUNCtion

Syntax

CURSor:PAIred:HDELTA?



Returns

<NR3>

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

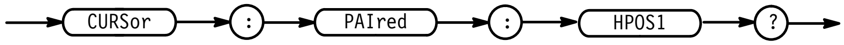
Cursor

Related Commands

CURSor:FUNCtion

Syntax

`CURSor:PAIred:HPOS1?`



Returns

`<NR3>`

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

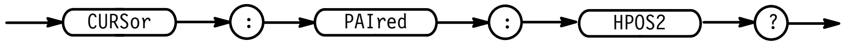
Cursor

Related Commands

CURSor:FUNCtion

Syntax

`CURSor:PAIred:HPOS2?`



Returns

`<NR3>`

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. The position returned is the time relative to the trigger position.

Group

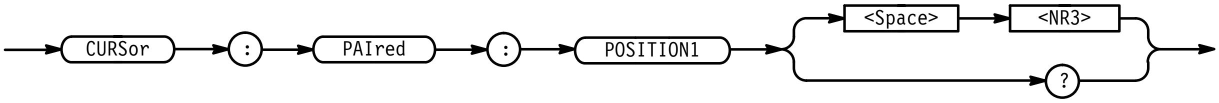
Cursor

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 after the trigger position.

`CURSOR:POSITION1?`
might return `1.00E-6`, indicating that the position of the first paired cursor is at 1 μs after the trigger position.

CURSor:PAIred:POSITION2

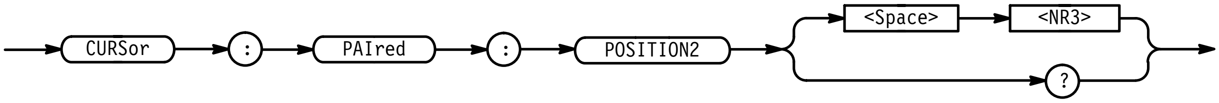
Sets or queries the vertical bar (time) position of the second paired cursor. The position returned is the time relative to the trigger position.

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 90.0E-6`
sets the second paired cursor position to 90 μs after the trigger position.

`CURSOR:POSITION2?`
might return `1.00E-6`, indicating that the position of the second paired cursor is 1 μs after the trigger position.

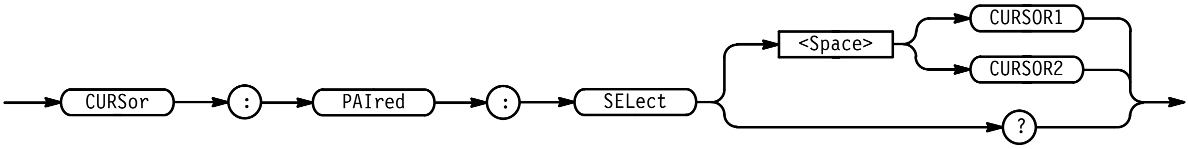
CURSor:PAIred:SELect

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

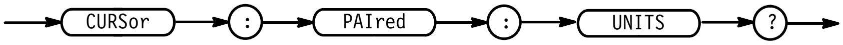
`CURSOR:PAIRED:SELECT?`
returns `CURSOR1` when the first paired cursor is the active cursor.

CURSor:PAIred:UNITS? (Query Only)

Returns `BASE`, which indicates there are no units other than volts available for measuring amplitude with cursors. This query is only available to maintain compatibility with Tektronix TDS model instruments.

Group Cursor

Syntax `CURSor:PAIred:UNITS?`



Returns `BASE`

Examples

```
CURSOR:PAIRED:UNITS?
```

returns `BASE` to indicate that only base units (volts) are available with paired cursors.

CURSor:PAIred:VDELTA? (Query Only)

Queries the Vbar (time) distance between paired cursors. It returns the absolute value of the first cursor horozontal position minus the second cursor horizontal position.

Group

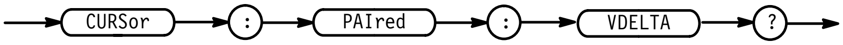
Cursor

Related Commands

CURSor:FUNCtion

Syntax

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



Returns

```
<NR3>
```

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 or returns the current vertical bar cursor settings for horizontal position, delta, cursor selection, and units.

Group

Cursor

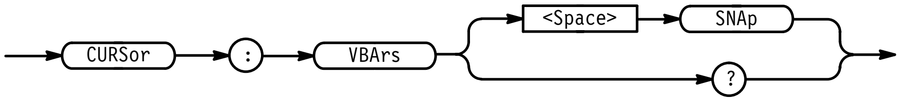
Related Commands

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?`

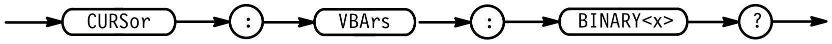
might return `:CURSOR:VBARS:UNITS SECONDS;POSITION1 1.00E–6;POSITION2 9.00E–6;SELECT CURSOR2`.

CURSor:VBArs:BINARY<x>? (Query Only)

“x” specifies cursor 1 or 2. A 16 character string is returned. 1, 0, ?, or ‘– ’ indicates channel logic level; the right-most character indicates channel 1.

Group Cursor

Syntax `CURSor:VBArs:BINARY<x>?`



Returns `<Qstring>` `<Qstring>`

Examples `CURSOR:VBARS:BINARY1?`

might return a string similar to “——————————?10” indicating that channel 1 is below the V_{LO} logic threshold, channel 2 is above the V_{HI} logic threshold, channel 3 is between thresholds, and channels 4 through 16 are not in groups.

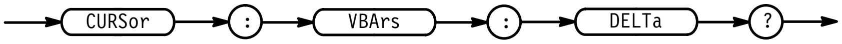
CURSor:VBArs:DELTa? (Query Only)

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>

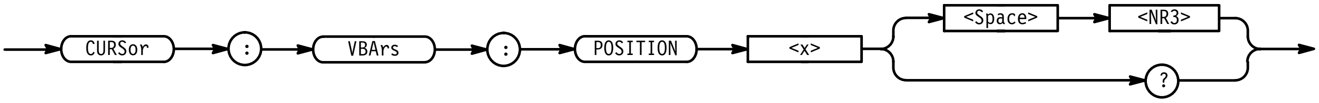
Positions a vertical bar cursor for both the vertical bar and paired cursors. Units are specified by the CURSor:VBArs:UNIts command.

Group Cursor

Related Commands CURSor:VBArs:UNIts

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

`CURSor:VBArs:POSITION<x>?`



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

Examples `CURSOR:VBARS:POSITION2 9.00E-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.

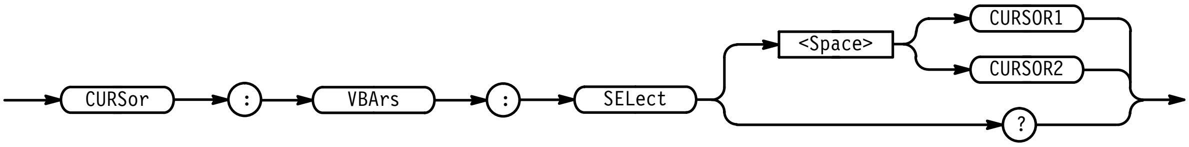
CURSor:VBArs:SELect

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 SELECT 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.

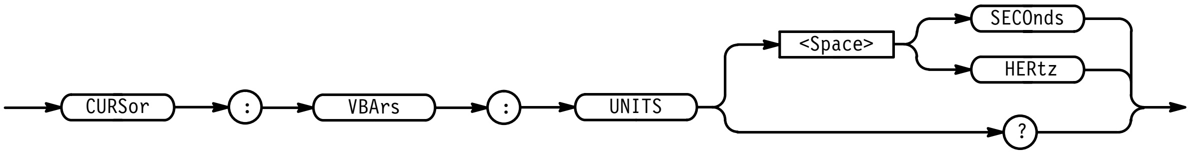
CURSor:VBArs:UNITS

Sets the vertical cursors to period or frequency. Using this command is equivalent to setting Time Units in the Cursor menu.

Group Cursor

Syntax `CURSor:VBArs:UNITS { SECOnds | HERtz }`

`CURSor:VBArs:UNITS?`



Arguments

`SECOnds` specifies that the time between cursors is measured in seconds.

`HERtz` specifies that the reciprocal of time between cursors is measured in Hertz.

Examples

`CURSor:VBArs:UNITS SECONDS`
specifies that the vertical cursors measure time in seconds.

`CURSor:VBArs:UNITS?`
might return `HERTZ` indicating the vertical cursors are set to measure frequency in Hertz.

CURVe

Transfers waveform data to and from the logic scope in binary or ASCII format. Each waveform has an associated waveform preamble that contains information such as data format and scale. Refer to the WFMPre command starting on page 2–322 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 you specify more than one source, the instrument returns a comma separated list of data blocks. 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. The instrument can transfer only one waveform at a time. The waveform will only be displayed if the reference is displayed.

***NOTE**. To transfer a reference to or from the logic scope, you must set the group containing the waveform to ANALOGALL, ANALOGSEL, BUS, or TIMING display type.*

A description of the waveform transfer process starts on page 2–36.

Group

Waveform

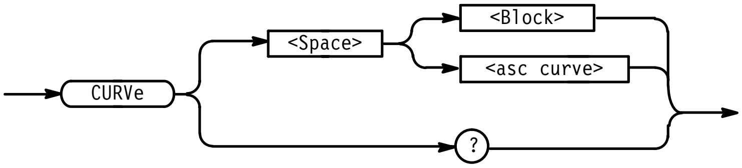
Related Commands

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 new line character at the end of the data. See the `GETGRP.C` example in the accompanying disk for more information.

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

Examples

```
CURVE?
```

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

DATa

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

Group

Waveform

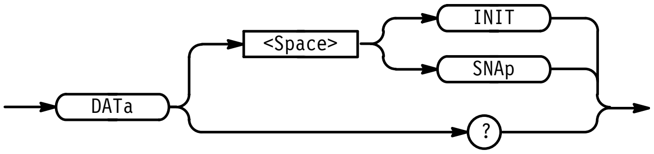
Related Commands

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

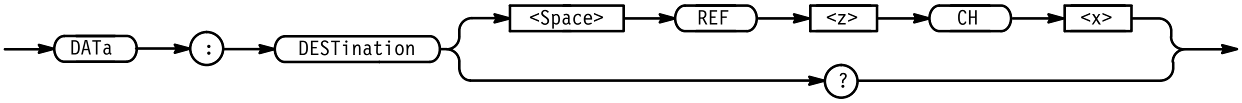
Sets or queries the reference group memory location and channel identification for storing waveform data transferred into the logic scope by the CURVe command. This command is identical to the DATa:TARget command.

NOTE*. To transfer a reference to the logic scope, the destination waveform channel must be in an ANALOGALL, ANALOGSEL, BUS, or TIMING display type.*

Group Waveform

Syntax `DATa:DESTination { REF<z>CH<x> }`

`DATa:DESTination?`



Arguments `REF<z>` (`REF1`, `REF2`, `REF3` or `REF4`) is the reference group memory location where the waveform will be stored. `CH<x>` (`CH1...16`, is the waveform ID).

Examples `DATA:DESTINATION REF3CH1`
stores incoming waveform data in reference group memory 3, as channel 1.

`DATA:DESTINATION?`
might return `REF2CH10` as the currently selected reference memory location.

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 2–23. Setting the DATa:ENCdg value causes the corresponding WFMPre values to be updated and vice versa.

Group

Waveform

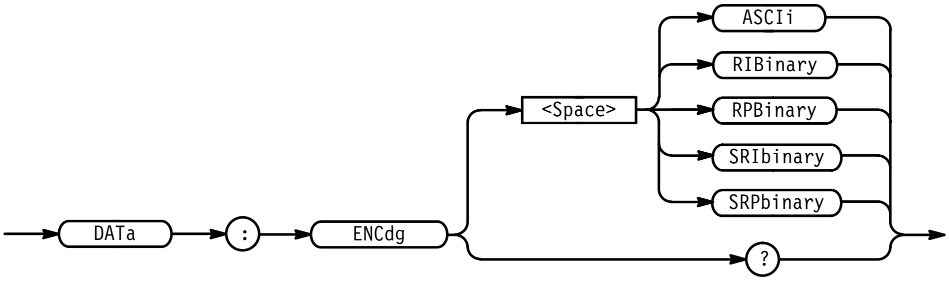
Related Commands

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 2–23: Data and WFMPre Parameter Settings

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

Examples

DATA:ENCDG RPBINARY

sets the data encoding format to be a 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 waveform data transferred from the instrument by the CURVe? query. The instrument always transfers the source data in a predefined order regardless of the order you specify using this command. The predefined order is CH1 – 16, GROUP1 – 8, REF1 – 4, REF1CH1 – 16, REF2CH1 – 16....

If GROUP<y> or REF<x> is sent as the parameter, all channels from the group will be returned. The source channels will change as the group or reference changes (the channels are not bound when the instrument receives the SOURCE command).

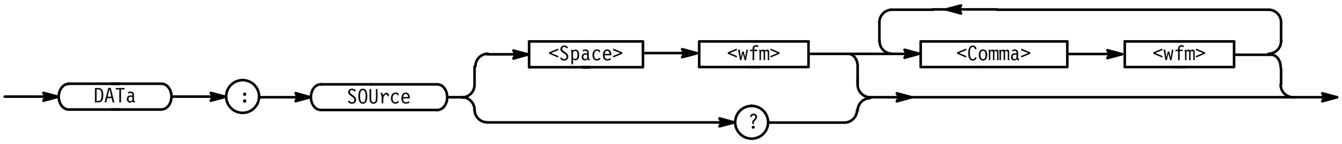
The query will return: group number of the reference, group number sent. It will not expand groups or references into component channels.

***NOTE**. To transfer a reference from the logic scope, the destination waveform channel must be in an ANALOGALL, ANALOGSEL, BUS, or TIMING display type.*

Group Waveform

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

`DATa:SOUrce?`



Arguments `<wfm>` is the location of the waveform data to be transferred from the logic scope to the controller.

Examples `DATA:SOURCE REF2, CH2, GROUP1, CH1`

specifies that the instrument will transfer four waveforms in the next CURVE? query. The order in which the instrument will transfer the data is CH1, CH2, then REF2.

`DATA:SOURCE?`

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

DATa:STARt

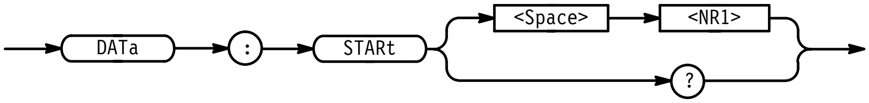
Sets or queries the starting data point for waveform transfer. This command allows for the transfer of partial waveforms to and from the logic scope.

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 the instrument transfers. 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 to be transferred when using the CURVe? query. This allows the transfer of partial waveforms to the controller.

When using the CURVe command, the logic scope stops reading data when there is no more data to read or the specified record length has been reached (the command will be ignored).

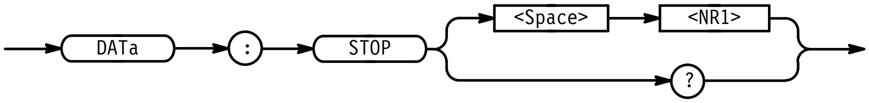
Group Waveform

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 the instrument transfers. If <NR1> is greater than the record length then the instrument will transfer data up to the record length. If both DATa:STARt and DATa:STOP are greater than the record length, an execution error will be returned. When DATa:STOP is less than DATa:STARt the values will be swapped internally for the CURVe? query.

If you wish to transfer complete waveforms only, set DATa:STARt to 1 and DATa:STOP to the maximum record length.

Examples

`DATA:STOP 1500`
specifies that the waveform transfer will stop at data point 1500.

`DATA:STOP?`
might return `1490` as the last data point to 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

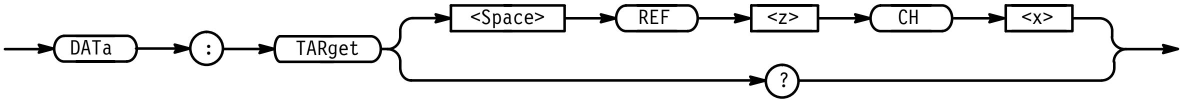
Related Commands

CURVe

Syntax

```
DATa:TARget REF<z>CH<x>
```

```
DATa:TARget?
```



DATa:WIDth

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

Group

Waveform

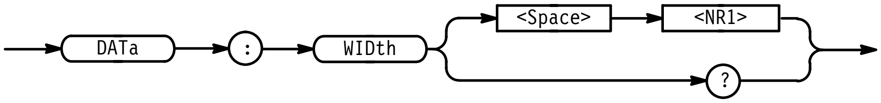
Related Commands

CURVe, WFMPre:BIT_Nr, WFMPre:BYT_Nr

Syntax

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

```
DATa:WIDth?
```



Arguments

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

<NR1> = 2 specifies there are 2 bytes (16 bits) per point. This format is useful with AVErage waveforms. If used for ENVelope 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 the logic scope displays.

Group

Miscellaneous

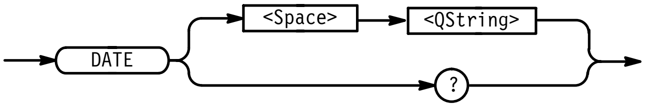
Related Commands

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 be a dash (–) after yyyy and mm.

Examples

DATE "1993-01-24"
sets the date to January 24, 1993.

DATE?
might return “1993–12–10”

*DDT

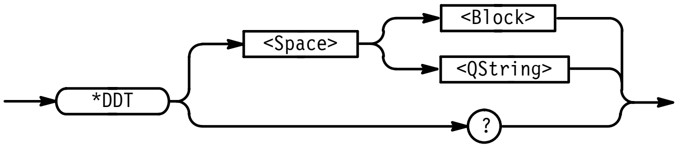
Allows the user to specify a command or a list of commands that the instrument will execute when it receives a *TRG command or the GET GPIB interface message. *DDT is a special alias *TRG uses.

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 arguments 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 instrument will start the acquisition system and ring the bell each time the controller sends a *TRG command.

`*DDT?`

might return `#10`, indicating the instrument will take no action when it receives *TRG.

DELEte:SETUp (No Query Form)

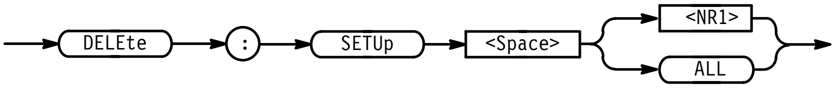
Removes stored setups from memory and initializes the location with the factory default setup.

NOTE*. You cannot recover setup information after deleting it.*

Group Save and Recall

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

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



Arguments `<NR1>` is a value from 1 to 10 specifying a setup storage location. Using an out-of-range value causes an execution error.

`ALL` specifies all stored setups.

Examples `DELETE:SETUP ALL`
removes all stored setups. The instrument initializes all ten storage locations to the factory default setup.

DELEte:GROUP (No Query Form)

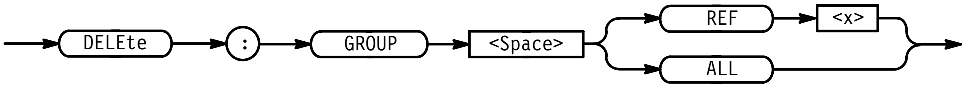
Deletes one or all of the stored reference waveform groups from memory; memory allocated to the reference location then becomes available for reallocation.

***NOTE**. The instrument does not clear waveform data from the reference location.*

Group Save and Recall

Related Commands RECAll:GROUP, SAVe:GROUP, TEKSecure

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



Arguments `REF<x>` (`REF1`, `REF2`, `REF3` or `REF`) specifies one of the reference memory locations.

`ALL` specifies all stored groups.

Examples

`DELETE:GROUP REF2`
removes the group stored at REF2.

`DELETE:GROUP ALL`
removes all groups stored in reference memory.

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 3–1.

Group

Status and Error

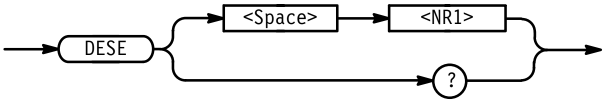
Related Commands

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

Syntax

```
DESE <NR1>

DESE?
```



Arguments

`<NR1>` is a value 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 (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 3–1.*

Examples

`DESE 209`
sets the DESER to binary 11010001, enabling the PON, URQ, EXE, and OPC bits.

`DESE?`
might return the string `:DESE 186`, indicating the DESER contains the binary value 10111010.

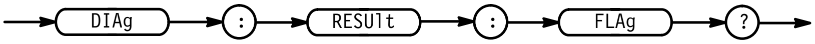
DIAg:RESUlt:FLAg? (Query Only)

Returns the pass/fail status of the last diagnostic test sequence. Use the DIAg:RESUlt:LOG? query to determine which test(s) have failed.

Group Calibration and Diagnostic

Related Commands DIAg:RESUlt:LOG?

Syntax `DIAg:RESUlt:FLAg?`



Returns `PASS` indicates all the selected diagnostic tests have passed.

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

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

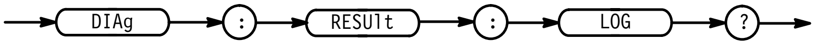
DIAg:RESUlt:LOG? (Query Only)

Returns the internal results log from the last diagnostic test sequence. The list contains all the modules and module interfaces 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)

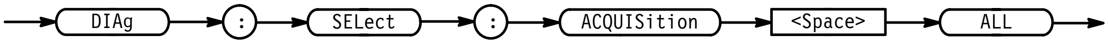
Selects the acquisition system test sequence to run when the DIAg:STATE EXECUte command is sent. This command is equivalent to setting Area in the Utility menu when System is set to Diag/Err.

Group

Calibration and Diagnostic

Syntax

```
DIAg:SELect:ACQUISition ALL
```



Arguments

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

Examples

`DIAG:SELECT:ACQUISITION ALL`
selects all functional, memory, and register tests.

DIAg:SELect:ALL (No Query Form)

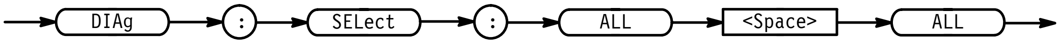
Specifies that all system test sequences will run when the DIAg:STATE EXECUte command is sent. This command is equivalent to setting Area in the Utility menu when System is set to Diag/Err.

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.

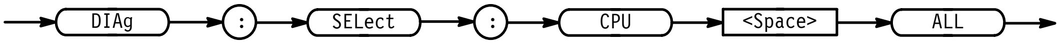
Examples DIAG:SELECT:ALL ALL
selects all functional, memory, and register tests for the acquisition, processor, and display systems test, and self diagnostics for the front panel test.

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.

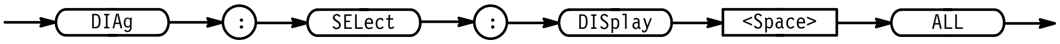
Examples DIAG:SELECT:CPU ALL
specifies that all functional, memory, and register tests be included in the processor system test.

DIAg:SELect:DISplay (No Query Form)

Selects the display system test sequence that will 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.

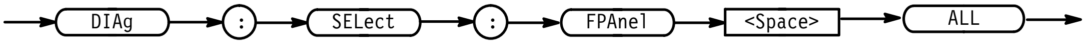
Examples `DIAG:SELECT:DISPLAY ALL`
specified that all functional, memory, and register tests be included in the display system test.

DIAg:SELect:FPAnel (No Query Form)

Selects the front-panel test sequence that will 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.

Examples `DIAG:SELECT:FPANEL ALL`
specified that all self diagnostic tests be included in the display system test.

DIAg:STATE (No Query Form)

Executes the diagnostic tests previously specified by the DIAg:SELect commands.

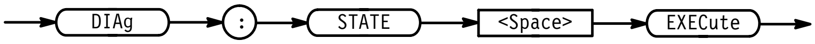
When the test sequence has completed, the instrument displays any of the modules or module interfaces that failed diagnostics on the screen and stores them in an internal log file. The pass/fail status will be returned by the DIAg:RESUlt:FLAg? query and the internal log will be returned by the DIAg:RESUlt:LOG? 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 may take 30 seconds or longer 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 logic scope will return to its pretest state. 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:

- Event Queue
- Input Queue
- 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
```

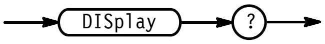
executes all selected diagnostic tests.

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`.

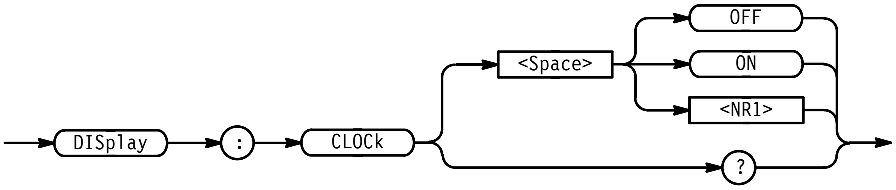
DISplay:CLOCk

Controls the display of the date and time. Using this command 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?`

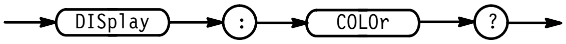
might return 1 indicating that the display shows time and date.

DISplay:COLOr? (Query Only)

Returns the display color settings.

Group Display

Syntax `DISplay:COLOr?`



Examples `DISPLAY:COLOR?`

might return

```
:DISPLAY:COLOR:PALETTE:REGULAR NORMAL;PERSISTENCE
TEMPERATURE;NORMAL:BACKGROUND 0,0,0;CI1 0,65,0;CI2 252,48,48;
CI3 167,48,75;CI4 306,35,100;CI5 44,39,72;CI6 75,33,74;TEXT
165,50,35;SCROLLTEXT 180,80,75;ZONE 145,62,99;COLLISION
132,54,80;GRATICULE 165,50,15;SCROLLBAR 130,40,45;:DISPLAY:
COLOR:PALETTE:BOLD:BACKGROUND 0,0,0;CI1 250,39,100;CI2
0,50,100;CI3 108,45,100;CI4 180,48,100;CI5 300,50,100;CI6
44,50,100;TEXT 165,50,35;SCROLLTEXT 180,80,75;ZONE 145,62,99;
COLLISION 0,100,0;GRATICULE 165,65,15;SCROLLBAR 130,40,45;
:DISPLAY:COLOR:PALETTE:HARDCOPY:BACKGROUND 0,100,0;CI1 0,0,0;
CI2 240,50,100;CI3 120,50,100;CI4 300,50,100;CI5 0,50,100;CI6
60,50,100;TEXT 0,0,0;SCROLLTEXT 0,100,0;ZONE 120,50,100;
COLLISION 120,50,100;GRATICULE 0,0,0;SCROLLBAR 0,50,0;
:DISPLAY:COLOR:PALETTE:MONO:BACKGROUND 0,0,0;CI1 252,48,100;
CI2 252,48,100;CI3 252,48,100;CI4 252,48,100;CI5 252,48,100;
CI6 252,48,100;TEXT 165,50,35;SCROLLTEXT 180,80,75;ZONE
145,62,99;COLLISION 132,54,80;GRATICULE 165,50,15;SCROLLBAR
130,40,45;:DISPLAY:CO.
```

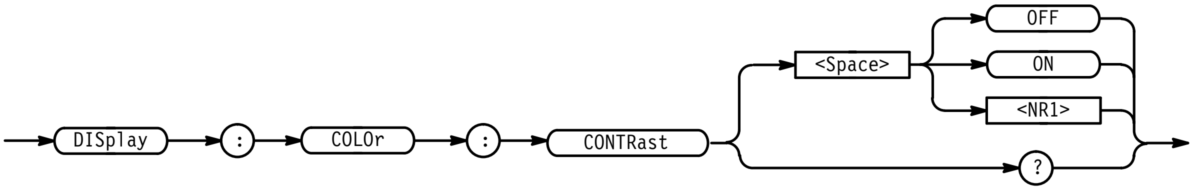
DISplay:COLOr:CONTRast

Turns on or off the collision contrast option. The logic scope will display overlapping lines in a special collision color when this item is active. Using this command is equivalent to selecting Options from the main Color menu and toggling Collision Contrast in the resulting side menu to ON or OFF.

Group Display

Syntax `DISplay:COLOr:CONTRast { OFF | ON | <NR1> }`

`DISplay:COLOr:CONTRast?`



Arguments

OFF or <NR1> = 0 turns off collision contrast.

ON or <NR1> ≠ 0 turns on collision contrast.

Examples

DISPLAY:COLOR:CONTRAST ON
turns on the contrast option.

DISPLAY:COLOR:CONTRAST?
might return 1, indicating the color contrast is on.

DISplay:COLOr:MAP? (Query Only)

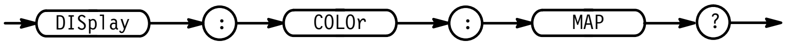
Returns the current color mappings for group and reference waveforms.

Group

Display

Syntax

```
DISplay:COLOr:MAP?
```



Examples

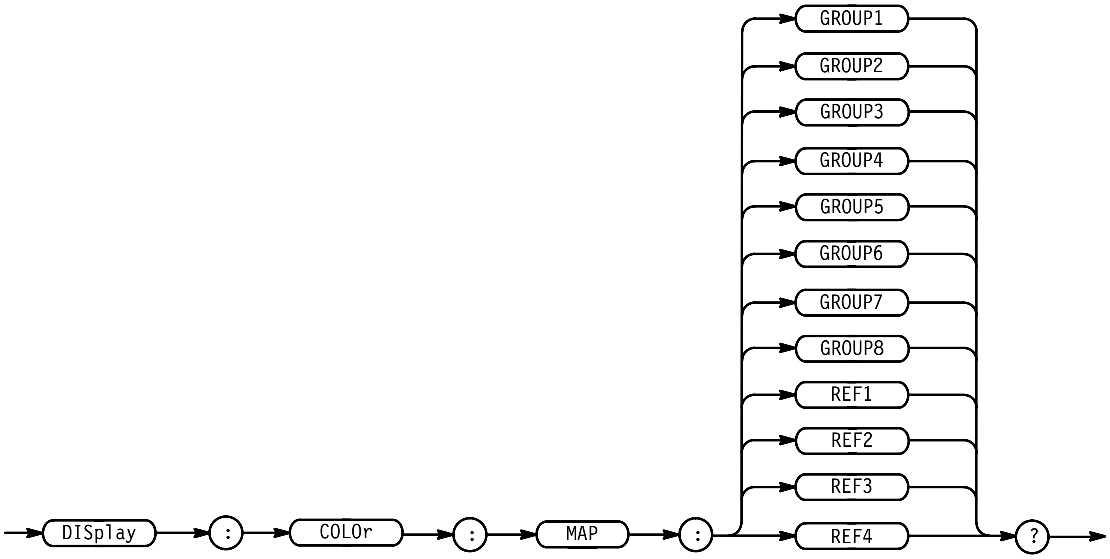
DISPLAY:COLOR:MAP?
might return :DISPLAY:COLOR:MAP:GROUP1:TO CI1;:DISPLAY:COLOR:MAP:GROUP2:TO CI2;:DISPLAY:COLOR:MAP:GROUP3:TO CI3;:DISPLAY:COLOR:MAP:GROUP4:TO CI4;:DISPLAY:COLOR:MAP:GROUP5:TO CI5;:DISPLAY:COLOR:MAP:GROUP6:TO CI6;:DISPLAY:COLOR:MAP:GROUP7:TO CI6;:DISPLAY:COLOR:MAP:GROUP8:TO CI6;:DISPLAY:COLOR:MAP:REF1:TO CI1;:DISPLAY:COLOR:MAP:REF2:TO CI2;:DISPLAY:COLOR:MAP:REF3:TO CI3;:DISPLAY:COLOR:MAP:REF4:TO CI4.

DISplay:COLOr:MAP:<item name>? (Query Only)

Returns the color index representing the color currently assigned to the item (group or reference waveform) specified. Sending this query is similar to pressing Map Group Colors or Map Reference Colors in the Color main menu, selecting a group in the side menu, and then reading the color index number under the label Color in the side menu.

Group Display

Syntax `DISplay:COLOr:MAP:{ GROUP1 | GROUP2 | GROUP3 | GROUP4 | GROUP5 | GROUP6 | GROUP7 | GROUP8 | REF1 | REF2 | REF3 | REF4 }?`



Examples `DISPLAY:COLOR:MAP GROUP3?`

might return `CI6` indicating that group 3 currently displays in the color indexed 6.

DISplay:COLOr:MAP:<item name>: TO

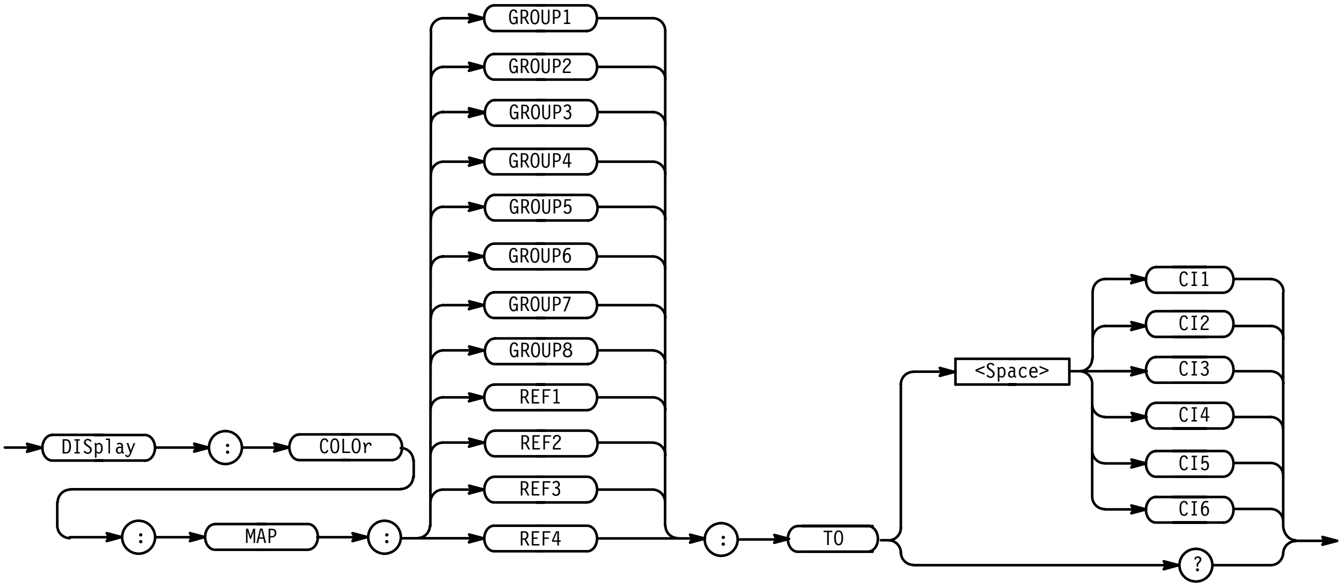
Defines the color index when setting the group or reference group to the color assigned to a specific index. Sending this command is similar to pressing Map Group Colors or Map Reference Colors on the Color main menu, pressing Color on the resulting side menu, or entering an index with the general purpose knob or the keypad.

Group Display

Syntax

```
DISplay:COLOr:MAP: { Group1 | Group2 | Group3 | Group4 | Group5 |
Group6 | Group7 | Group8 | Ref1 | Ref2 | Ref3 | Ref4 }:TO { CI1 |
CI2 | CI3 | CI4 | CI5 | CI6 }
```

```
DISplay:COLOr:MAP:Group<x>:TO?
```



Arguments

GROUP1 through GROUP8 are the eight user-definable groups for acquiring waveforms.

REF1 through REF4 are the four reference groups in which the instrument can store waveforms display as references.

CI1 through CI6 are the color indexes from 1 to 6.

Examples

DISPLAY:COLOR:MAP:GROUP1:TO CI3
sets the color of all Group1 waveforms to palette color CI3.

DISPLAY:COLOR:MAP:GROUP1:TO?
might return CI3 indicating that Group 1 is set to palette color CI3.

DISplay:COLOr:PALEtte? (Query Only)

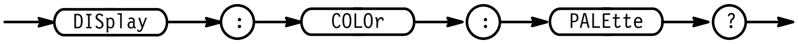
Returns color palette settings.

Group

Display

Syntax

```
DISplay:COLOr:PALEtte?
```



Examples

```
DISPLAY:COLOR:PALETTE?
```

might return

```
:DISPLAY:COLOR:PALETTE:REGULAR NORMAL;PERSISTENCE
TEMPERATURE;NORMAL:BACKGROUND 0,0,0;CI1 0,65,0;CI2 252,48,48;
CI3 167,48,75;CI4 306,35,100;CI5 44,39,72;CI6 75,33,74;TEXT
165,50,35;SCROLLTEXT 180,80,75;ZONE 145,62,99;COLLISION
132,54,80;GRATICULE 165,50,15;SCROLLBAR 130,40,45;:DISPLAY:
COLOR:PALETTE:BOLD:BACKGROUND 0,0,0;CI1 250,39,100;CI2
0,50,100;CI3 108,45,100;CI4 180,48,100;CI5 300,50,100;CI6
44,50,100;TEXT 165,50,35;SCROLLTEXT 180,80,75;ZONE 145,62,99;
COLLISION 0,100,0;GRATICULE 165,65,15;SCROLLBAR 130,40,45;
:DISPLAY:COLOR:PALETTE:HARDCOPY:BACKGROUND 0,100,0;CI1 0,0,0;
CI2 240,50,100;CI3 120,50,100;CI4 300,50,100;CI5 0,50,100;
CI6 60,50,100;TEXT 0,0,0;SCROLLTEXT 0,100,0;ZONE 120,50,100;
COLLISION 120,50,100;GRATICULE 0,0,0;SCROLLBAR 0,50,0;
:DISPLAY:COLOR:PALETTE:MONO:BACKGROUND 0,0,0;CI1 252,48,100;
CI2 252,48,100;CI3 252,48,100;CI4 252,48,100;CI5 252,48,100;
CI6 252,48,100;TEXT 165,50,35;SCROLLTEXT 180,80,75;ZONE
145,62,99;COLLISION 132,54,80;GRATICULE 165,50,15;SCROLLBAR
130,40,45.
```

DISplay:COLOr:PALEtte:PERSistence

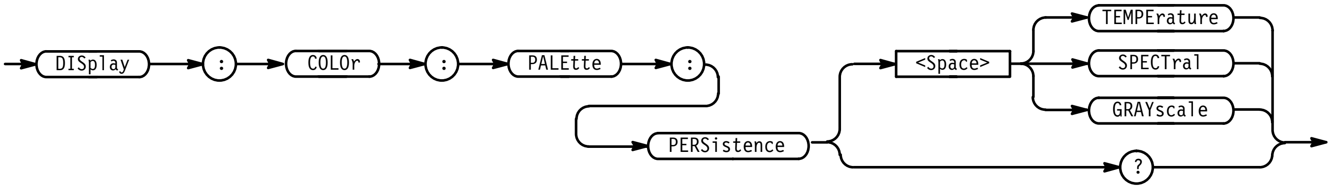
Sets the current persistence palette to one of the preset palettes. This command is equivalent to selecting Palette from the main Color menu Persistence Palettes from the resulting side menu, and Temperature (Spectral or Gray Scale) from the next side menu.

Group Display

Syntax

```
DISplay:COLOr:PALEtte:PERSistence { TEMPErature | SPECTral |
GRAYscale }
```

```
DISplay:COLOr:PALEtte:PERSistence?
```



Arguments

TEMPErature displays variable persistence style waveforms with their more frequently occurring (more persistent) waveform elements in warmer colors (red shades). Less persistent elements appear in cooler colors (blue shades).

SPECTral displays variable persistence style waveforms with their more persistent elements in blue shades. Less persistent elements appear in red shades.

GRAYscale displays variable persistence style waveforms with their more persistent elements in lighter gray shades. Less persistent elements appear in darker gray shades.

Examples

```
DISPLAY:COLOR:PALETTE:PERSISTENCE TEMPERATURE
```
sets temperature palette as the current persistence palette.

```
DISPLAY:COLOR:PALETTE:PERSISTENCE?
```
might return SPECTRAL indicating the logic scope currently displays waveforms in the spectral when persistence is the display mode.

DISplay:COLOr:PALEtte:REGular

Sets the current palette to one of the preset palettes. Using this command is equivalent to selecting Palette from the main Color menu and Normal, Bold, Hardcopy, or Monochrome from the resulting side menu.

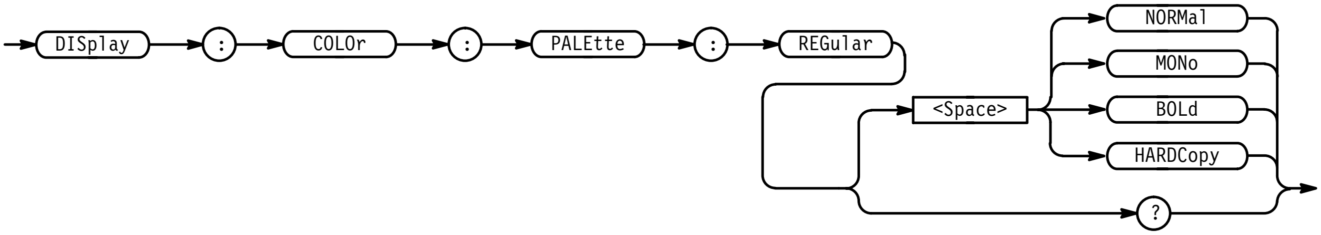
Group

Display

Syntax

```
DISplay:COLOr:PALEtte:REGular { NORMal | MONo | BOLd | HARDCopy }

DISplay:COLOr:PALEtte:REGular?
```



Arguments NORMal sets the color of the display elements to those hues and lightness levels best for overall viewing.

MONo sets waveforms to monochromic levels while setting other display elements to colors.

BOLd sets the color of the display elements for brighter, more vivid hues.

HARDCopy sets the color of the display elements to produce good copies of the display screen when sent to a wide variety of hardcopy devices.

Examples DISPLAY:COLOR:PALETTE:REGULAR HARDCOPY
sets the current palette to the hardcopy palette.

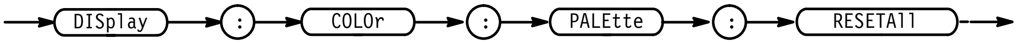
DISPLAY:COLOR:PALETTE:REGULAR?
might return BOLD indicating that the logic scope currently displays screen elements using the bold palette.

DISplay:COLOr:PALEtte:RESETAll (No Query Form)

Restores all palettes to their factory default setting. Using this command is equivalent to selecting Restore Colors from the main Color menu and Reset All Palettes to Factory from the resulting side menu.

Group Display

Syntax DISplay:COLOr:PALEtte:RESETAll



Arguments None

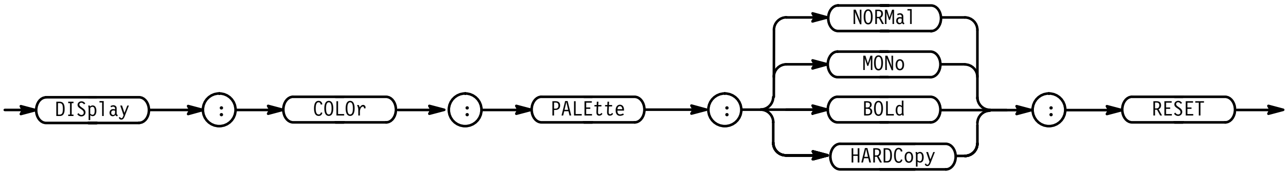
Examples DISPLAY:COLOR:PALETTE:RESETALL
resets the palette to the factory default setting.

DISplay:COLOr:PALEtte:<palette name>:RESET (No Query Form)

Restores the given palette to its factory defaults. Using this command is useful if you have edited the colors in a palette and is equivalent to selecting Restore Colors from the main Color menu and Reset Current Palette to Factory from the resulting side menu.

Group Display

Syntax `DISplay:COLOr:PALEtte:{ NORMal | MONo | BOLd | HARDCopy }:RESET`



Arguments `NORMal` is the palette with factory default settings that color the display elements in those hues and lightness levels best for overall viewing.

`MONo` is the palette with factory default settings that color waveforms to monochromic levels while setting other display elements to colors.

`BOLd` is the palette with factory default settings that color of the display elements for brighter, more vivid hues.

`HARDCopy` is the palette with factory default settings that sets the color of the display elements to produce good copies of the display screen.

Examples `DISPLAY:COLOR:PALETTE:HARDCOPY:RESET`
resets the hardcopy palette to its initial, factory-default settings.

DISplay:COLOr:PALEtte:<palette name>:<item name>

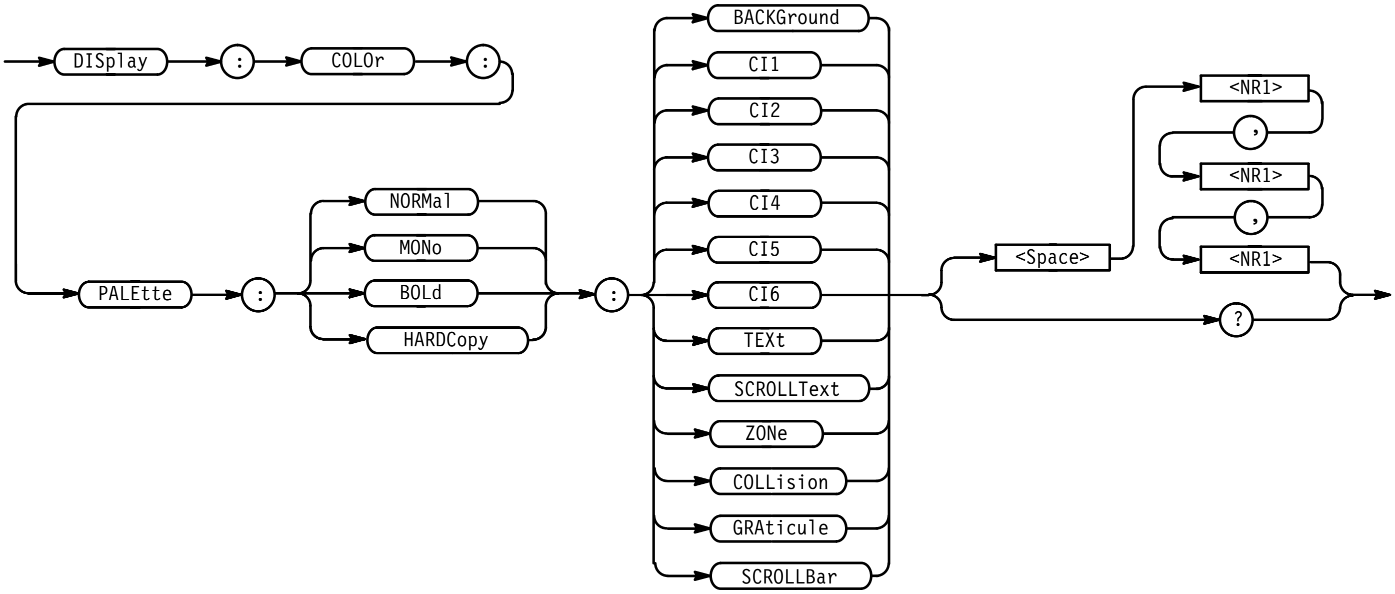
Sets the specified item to the specified color in the specified palette.

Group Display

Syntax

```
DISplay:COLOr:PALEtte:{ NORMal | MONo | BOLd | HARDCopy }:
{ BACKGround | CI1 | CI2 | CI3 | CI4 | CI5 | CI6 | TEXT |
SCROLLText | ZONe | COLLision | GRAticule | SCROLLBar }
<NR1>,<NR1>,<NR1>
```

```
DISplay:COLOr:PALEtte:{ NORMal | MONo | BOLd | HARDCopy }:
{ BACKGround | CI1 | CI2 | CI3 | CI4 | CI5 | CI6 | TEXT |
SCROLLText | ZONe | COLLision | GRAticule | SCROLLBar }?
```



Arguments

BACKGround specifies the display background color.

CI1 through CI6 are the color indexes for groups and waveforms from 1 to 6.

TEXt specifies the text color in all menus and all readouts not associated with a waveform.

SCROLLTEXT specifies text in the scroll bar. For example, this covers the scroll bars used in the file system.

ZONe specifies the color of the intensified zone on the waveform when the time base is intensified and the color of the real samples when the display style is Intensified Samples.

COLLision specifies the color of the areas where waveforms overlap each other. You can turn collision marking on or off with the Items, Waveforms, Options, Collision Contrast menu item.

GRAticule specifies the color of the graticule. Use the Settings, Display, Graticule option to select the graticule type.

SCROLLBAR specifies the color of the scrollbar. For example, you will find a scrollbar in various file system menus.

<NR1>, <NR1>, <NR1> specifies the desired colors in terms of hue, lightness and saturation values.

Hue is the wavelength of light reflected from the surface. It varies continuously along the color spectrum as produced by a rainbow. Values range from 0 to 359. Sample values are: 0 = blue, 60 = magenta, 120 = red, 180 = yellow, 240 = green, 360 = cyan.

Lightness refers to the amount of light reflected from the surface. It varies from black, to the nominal color, to white. Values range from 0 to 100. A value of 0 results in black. A value of 50 provides the nominal color. A value of 100 results in white.

Saturation is the intensity of color. Completely desaturated color is gray. Completely saturated color of any hue is that color at its most intense. Values range from 0 to 100. A value of 100 provides a pure color. A value of 50 provides gray.

Examples

`DISPLAY:COLOR:PALETTE:HARDCOPY:CI1 120 50 100`
sets the CI1 color of the hardcopy palette to 120 hue, 50 lightness, and 100 saturation, providing a pure red.

`DISPLAY:COLOR:PALETTE:BOLD:CI6?`
might return `0, 50, 100`, indicating color index CI6 is blue.

DISplay:FILTer

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

Group

Display

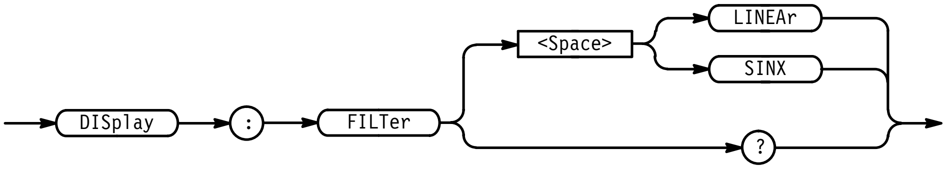
Related Commands

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 the instrument fits acquired points to a curve.

Examples

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

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

DISplay:FORMat? (Query Only)

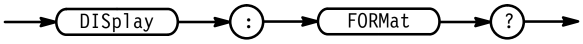
Queries the display format. The logic scope always returns YT.

Group

Display

Syntax

```
DISplay:FORMat?
```



Returns

`YT`

Examples

`DISPLAY:FORMAT?`
always returns `YT` for the display format, since the logic scope always displays in a voltage versus time format.

DISplay:GRAticule

Selects the type of graticule that the instrument will display. This command is equivalent to setting Graticule in the Display menu.

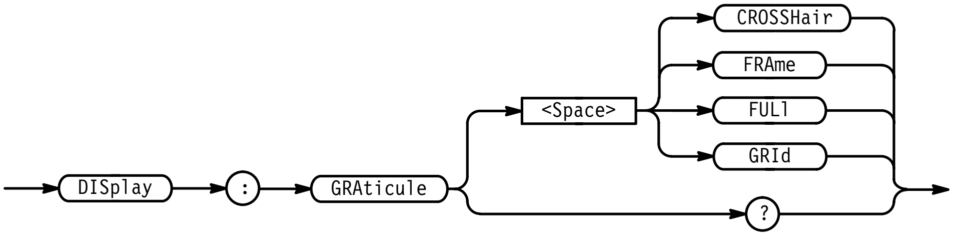
Group

Display

Syntax

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

DISplay:GRAticule?
```



Arguments

`CROSSHair` specifies a frame and cross hairs.

`FRAme` specifies a frame.

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

`GRId` specifies a frame and a grid.

Examples

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

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

DISplay:INTENSITy? (Query Only)

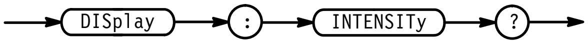
Returns the current intensity settings for different parts of the display.

Group

Display

Syntax

`DISplay:INTENSITy?`



Examples

`DISPLAY:INTENSITY?`
might return `:DISPLAY:INTENSITY:WAVEFORM 70;TEXT 60`.

DISplay:INTENSITy:TEXt

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

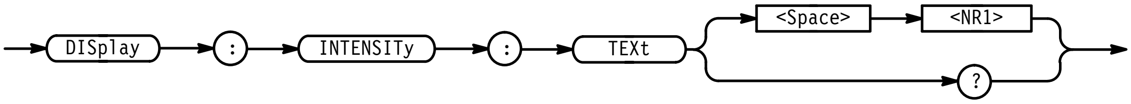
Group

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

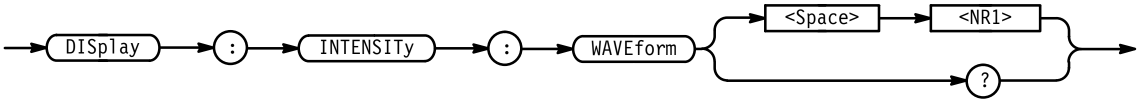
Sets waveform intensity. This command is equivalent to setting Waveform in the Display Intensity side menu.

Group Display

Syntax

```
DISplay:INTENSITy:WAVEform <NR1>
```

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



Arguments <NR1> ranges from 20 to 100 percent.

Examples

```
DISPLAY:INTENSITY:WAVEFORM 50
```

sets the intensity of waveforms to 50% of maximum.

```
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.

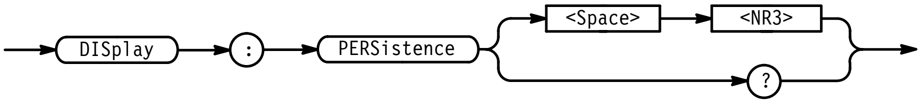
Group Display

Related Commands DISplay:STYle

Syntax

```
DISplay:PERSistence <NR3>
```

```
DISplay:PERSistence?
```



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

Examples

```
DISPLAY:PERSISTENCE 3
```

specifies that that the instrument displays waveform points on the screen for 3 seconds before fading.

```
DISPLAY:PERSISTENCE?
```

might return `1.5E+0`, indicating the persistence is 1.5 second.

DISplay:STYle

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

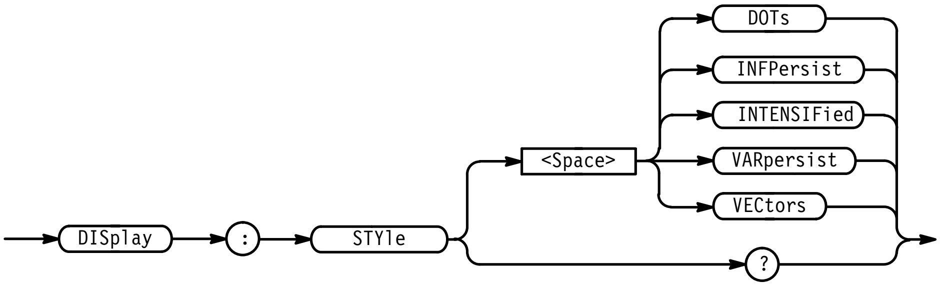
Group Display

Related Commands DISplay:PERSistence

Syntax

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

```
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.

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

`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.

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

Invokes the trigger indicator display. Using this command 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).

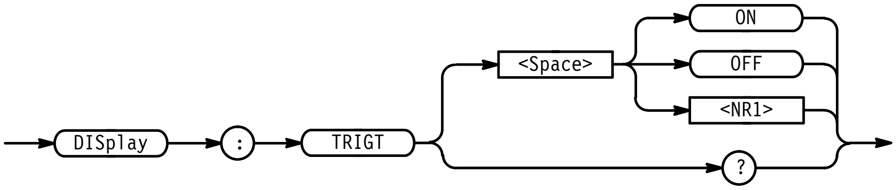
Group

Display

Syntax

```
DISplay:TRIGT { ON | OFF | <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

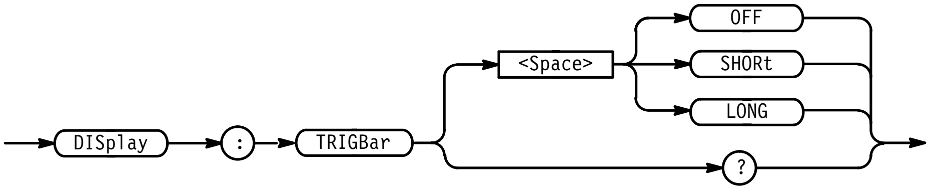
Controls the display of the trigger bar indicators. The bar indicates where the trigger will occur, in voltage.

The logic scope will only display the bar if the trigger source is also displayed. If both a main and a delayed trigger are displayed, two bars will appear; one will accompany each source. If a logic trigger is selected, multiple bars may appear.

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 a long trigger bar indicator (or indicators).

```
DISPLAY:TRIGBAR?
```
might return OFF, indicating the trigger bar is not displayed.

*ESE

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

Group

Status and Error

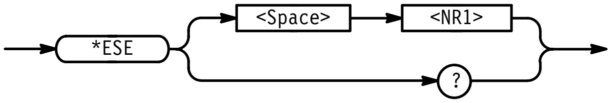
Related Commands

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

Syntax

```
*ESE <NR1>
```

```
*ESE?
```



Arguments

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

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

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

Examples `*ESE 209`

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

`*ESE?`

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

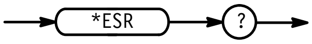
*ESR? (Query Only)

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

Group Status and Error

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

Syntax `*ESR?`



Examples `*ESR?`

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

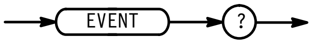
EVENT? (Query Only)

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

Group Status and Error

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

Syntax `EVENT?`



Examples

`EVENT?`

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

EVMsg? (Query Only)

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

Group

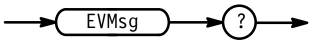
Status and Error

Related Commands

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

Syntax

`EVMsg?`



Returns

The event code and message in the following format:

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

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

where `<Command>` is the command that caused the error and may be returned when a command error is detected by the logic scope. 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"`.

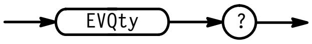
EVQty? (Query Only)

Returns the number of event codes in the Event Queue. Using this command 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 logic scope 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 logic scope to factory default:

- 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:

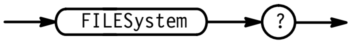
- the state of the GPIB (IEEE Std 488.1-1987) interface.
- the selected GPIB address.
- calibration data that affects device specifications.
- protected user data.
- stored settings.
- the current password (if implemented).

FILESystem? (Query Only)

Returns file system settings.

Group File System

Syntax `FILESystem?`



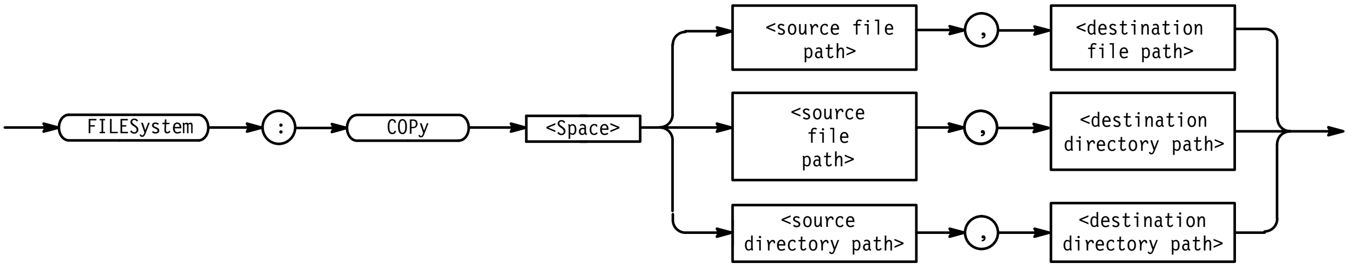
Examples `FILESystem?`
might return `:FILESYSTEM:CWD "fd0:";DELWARN 1;OVERWRITE 0`.

FILESystem:COPy (No Query Form)

Copies a named file to a new file. The new file may be in a totally separate directory than the old file. Wild card characters (*.*) may be used to copy multiple files with one command.

Group File System

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



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

`<directory path>` is a quoted string that defines the directory. Input the directory using the form `<drive>/<dir>/<directory name>`. `<drive>` and one or more `<dir>`'s are optional. If you do not specify them, the logic scope will copy the directory in the current directory. `<directory name>` stands for a directory name of up to 8 characters.

Examples
```
FILESYSTEM:COPY "TEK00001.SET","fd0:/TEK00001.SET"
```
copies the file named TEK00001.SET on the current drive to a file named TEK00001.SET on the drive fD0: in the root directory.

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

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

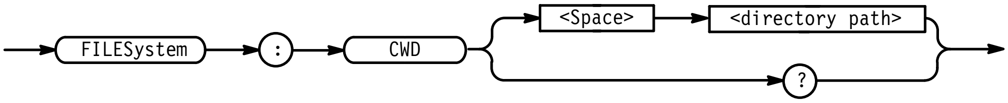
FILESystem:CWD

Sets or returns the current working directory (CWD) path.

Group File System

Syntax `FILESystem:CWD <directory path>`

`FILESystem:CWD?`



Arguments `<directory path>` is a quoted string defining the complete path to a valid directory, residing on the file system disk.

Examples `FILESYSTEM:CWD "fd0:/MYDIR"`
will define `FD0:/MYDIR` as the current directory.

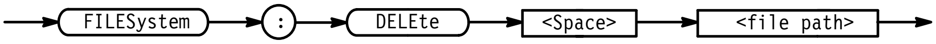
`FILESYSTEM:CWD?`
might return `"FD0:/MYDIR"`, if that was the current directory.

FILESystem:DELEte (No Query Form)

Deletes a named file.

Group File System

Syntax `FILESystem:DELEte <file path>`



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

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

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

FILESystem:DELWarn

Turns on or off the front panel file delete warning. No warning is returned by way of GPIB.

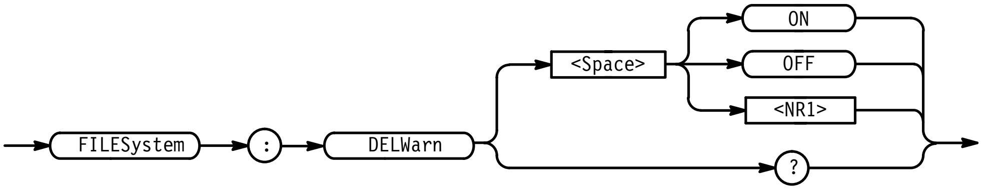
Group

File System

Syntax

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

```
FILESystem:DELWarn?
```



Arguments

ON or <NR1> ≠ 0 turns on the front panel delete warning.

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

Examples

```
FILESYSTEM:DELWARN OFF
```

disables the front panel delete warning.

```
FILESYSTEM:DELWARN?
```

might return 0 indicating the front panel warning is disabled.

FILESystem:DIR? (Query Only)

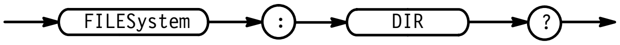
Returns a list of quoted strings. Each string contains the name of a file or directory in the current directory.

Group

File System

Syntax

```
FILESystem:DIR?
```



Examples

```
FILESYSTEM:DIR?
```

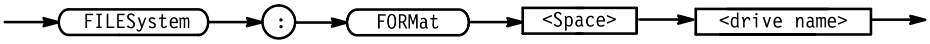
returns a list of files and directories in the default directory.

FILESystem:FORMat (No Query Form)

Formats a named drive.

Group File System

Syntax `FILESystem:FORMat <drive name>`



Arguments `<drive name>` is a quoted string that defines the disk drive to format.

`fd0` : refers to the floppy-disk drive built into the logic scope.

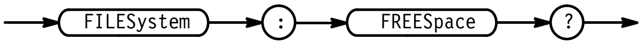
Examples `FILESYSTEM:FORMAT "fd0:"`
formats the media on drive fdo:.

FILESystem:FREESpace? (Query Only)

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

Group File System

Syntax `FILESystem:FREESpace?`



Returns `<NR1>`

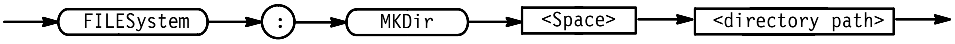
Examples `FILESYSTEM:FREESpace?`
might return `0` as the amount of free space available if the drive was full.

FILESystem:MKDir (No Query Form)

Creates a new directory.

Group File System

Syntax `FILESystem:MKDir <directory path>`



Arguments `<directory path>` is a quoted string that defines the directory. Input the directory using the form `<drive>/<dir>/<directory name>`. `<drive>` and one or more `<dir>`’s are optional. If you do not specify them, the logic scope will create the directory in the current directory. `<directory name>` stands for a directory name of up to 8 characters.

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

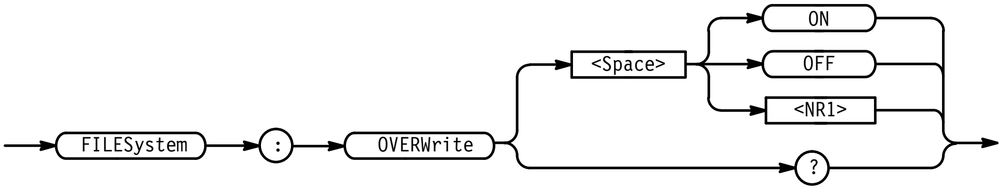
FILESystem:OVERWrite

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

Group File System

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

`FILESystem:OVERWrite?`



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

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

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

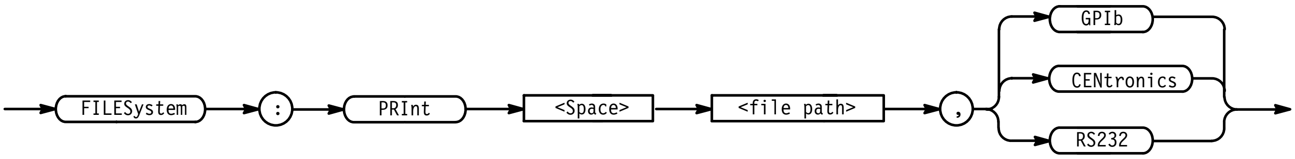
`FILESYSTEM:OVERWRITE?`
might return `0` indicating you cannot overwrite existing files.

FILESystem:PRInt (No Query Form)

Prints a named file to the named port.

Group File System

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



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

`GPIb` specifies that the hardcopy is directed to the GPIB port.

`CENtronics` specifies that the hardcopy is directed to the Centronics port.

`RS232` specifies that the hardcopy is directed to the RS232 port.

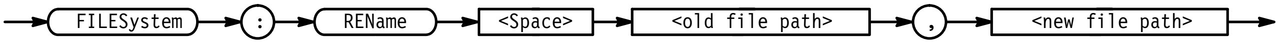
Examples `FILESYSTEM:PRINT "TEK00000.IBM",CENTRONICS`
directs the file named `TEK00000.IBM` to the Centronics port.

FILESystem:REName (No Query Form)

Assigns a new name to a file.

Group File System

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



Arguments `<old file path>` is a quoted string that defines the file to rename. Input the file path using the form `<drive>/<dir>/<filename>`. `<drive>` and one or more

<dir>'s are optional. If you do not specify them, the logic scope will look for the filename in the current directory.

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

Examples

```
FILESYSTEM:RENAME "TEK00000.SET","MYSETTING.SET"
```

gives the file named TEK00000.SET the new name of MYSETTING.SET. The file remains on the current directory.

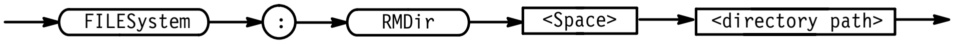
FILESystem:RMDir (No Query Form)

Deletes a named directory.

Group File System

Syntax

```
FILESystem:RMDir <directory path>
```



Arguments <directory path> is a quoted string that defines the directory. Input the directory using the form <drive>/<dir>/<directory name>. <drive> and one or more <dir>'s are optional. If you do not specify them, the logic scope will delete the directory in the current directory. <directory name> stands for a directory name of up to 8 characters.

Examples

```
FILESYSTEM:RMDIR "NEATPICS"
```

deletes the directory named NEATPICS in the current directory.

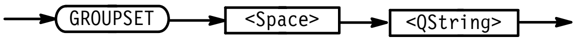
GROUPSET (No Query Form)

Sets the channel contents for *all* groups replacing the current grouping and grouping channel assignments. This command deletes the current channel grouping specification and replaces it with new grouping assignments.

Group Vertical

Syntax

```
GROUPSET <QString>
```



Arguments `<QString>` is used to assign channels to groups. A string must be from 1 to 16 characters in length containing the characters 1 through 8 and “–” (dashes). Each character position in the string is interpreted as a channel number. The right most character maps to channel 1, the second right-most character to channel 2, etc. In a 16 character string, the left-most character maps to channel 16. The number in each character position assigns the mapped channel number to the group number specified. A dash character indicates that the channel will not be assigned to a group. If the string is less than 16 characters, the channel positions omitted are assumed to have a dash character and will not be assigned to groups.

Invalid group channel specifications will be ignored. For example, assigning channels 1 and 3 to group 1 and channel 2 to group 2, will cause the command to be ignored, since channels within group 1 must be contiguous.

Examples `GROUPSET "8888888811111111"`
assigns channels 1 through 8 to group 1 and channels 1 through 16 to group 8.

`GROUPSET "----------111111"`
assigns channels 1 through 16 to group 1.

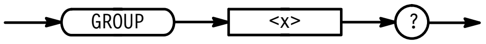
`GROUPSET "11111111--111111"`
will be ignored as group 1. All channels must be contiguous.

GROUP<x>? (Query Only)

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

Group Vertical

Syntax `GROUP<x>?`



Examples `GROUP1?`
might return the string `:GROUP1:SCALE 10.0E-3;POSITION 0.0E+0;OFFSET 0.0E+0;COUPLING DC;IMPEDANCE 75 Ω;BANDWIDTH FULL` for group 1.

GROUP<x>:BANdwidth

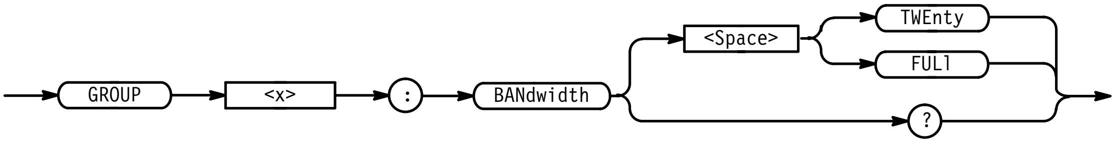
Sets or queries the bandwidth setting of the specified group. Using this command is equivalent to setting Bandwidth in the Vertical menu.

Group Vertical

Syntax

```
GROUP<x>:BANdwidth { TWEnty | FULl }
```

```
GROUP<x>:BANdwidth?
```



Arguments `TWEnty` sets the group bandwidth to 20 MHz.

`FULl` sets the group bandwidth to the full bandwidth of the logic scope.

Examples

```
GROUP2:BANDWIDTH TWENTY
```

sets the bandwidth of group 2 to 20 MHz.

```
GROUP1:BANDWIDTH?
```

might return `FULL`, indicating there is no bandwidth limiting on group 1.

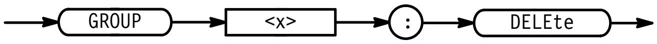
GROUP<x>:DELEte (No Query Form)

Deletes the specified group. All channels in the group are released for use in defining other groups. Sending this command is the same as press Delete in the Group Definition menu. <x> can equal 1 through 8.

Group Vertical

Syntax

```
GROUP<x>:DELEte
```



Examples

```
GROUP8:DELETE
```

would delete group 8.

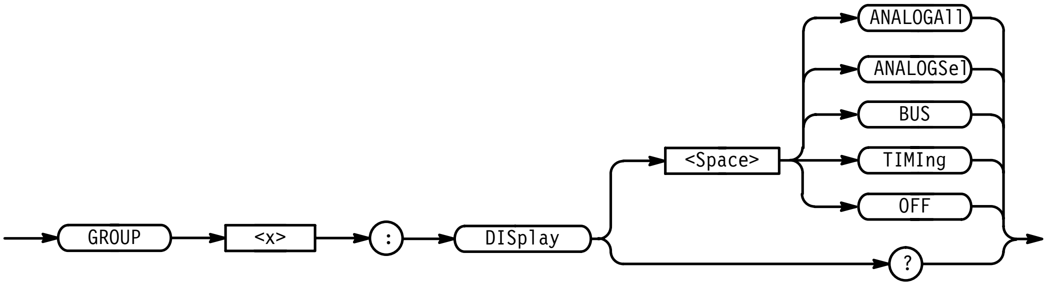
GROUP<x>:DISplay

Sets or queries the display mode for the group.

Group Vertical

Syntax

```
GROUP<x>:DISplay { ANALOGAll | ANALOGSel | BUS | TIMIng | OFF }
GROUP<x>:DISplay?
```



Arguments

ANALOGAll displays each waveform in the group as the analog of its input signal. Use Analog All mode to measure the actual response of a group of waveforms.

ANALOGSel displays only the selected waveform in a group as the analog input signal. Use Analog Select mode to measure the actual response of the selected waveform in a group.

BUS displays each waveform in the group as one or more of three digital logic levels defined as HIGH, LOW, or MID. The levels of each waveform displayed depends on how the instantaneous level of its input signal compares with two threshold levels you define. Use BusForm mode to measure the idealized response of waveforms with respect to defined thresholds.

TIMIng displays each waveform in the group as one or more of three digital logic levels, as is done for BusForm. However, unlike BusForm, the logic scope offsets the ground reference of each waveform so they skew vertically on screen. Use Timing mode to measure timing relationships between the waveforms included in the group.

OFF turns off the display of the selected group.

Examples

```
GROUP3:DISPLAY BUS
```

sets group 3 to BusForm display mode.

```
GROUP1:DISPLAY?
```

might return ANALOGALL display mode.

GROUP<x>:FIRst

Sets or queries the first channel in the group. Every group includes the first channel (lowest numbered in the group), the last channel (highest numbered), and every channel in between. A channel can only belong to one group at a time.

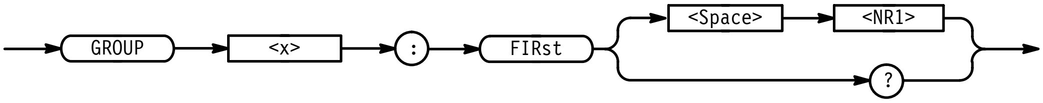
Group Vertical

Related Commands GROUP<x>:LAST, GROUP<x>:SELect

Syntax

```
GROUP<x>:FIRst <NR1>
```

```
GROUP<x>:FIRst?
```



Arguments `<NR1>` specifies the first channel in the group from 1 to 16.

Examples

```
GROUP2:FIRST 5
```

sets the first channel in group 2 to channel 5.

```
GROUP8:FIRST?
```

might return 12 indicating the first channel of group 8 is channel 12.

GROUP<x>:HEIght

Sets or queries the display height setting (in divisions) for a group set to BusForm or Timing Diagram display modes. BusForm and Timing Diagram modes display waveform levels as logical HIGHs (1) or LOWs (0). Setting the height specifies the number of vertical divisions between the HIGH and LOW levels.

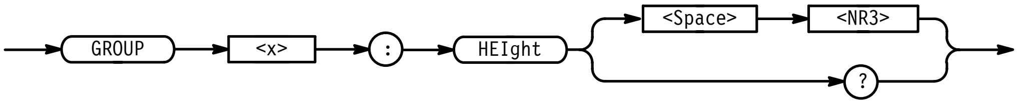
Group Vertical

Related Commands REF<x>:HEIght

Syntax

```
GROUP<x>:HEIght <NR3>
```

```
GROUP<x>:HEIght?
```



Arguments <NR3> is the height setting from 200E–3 to 8.0E+0.

Examples `GROUP2:HEIGHT 5`
displays the HIGH and LOW levels for waveforms displayed in group 2 five vertical divisions apart.

`GROUP7:HEIGHT?`
might return 2.0E+0 indicating the HIGH and LOW levels for waveforms displayed in group 7 are spaced two vertical divisions apart.

GROUP<x>:LAST

Sets or queries the last channel in the group. Every group includes the first channel (lowest numbered in the group), the last channel (highest numbered), and every channel in between. A channel can only belong to one group at a time.

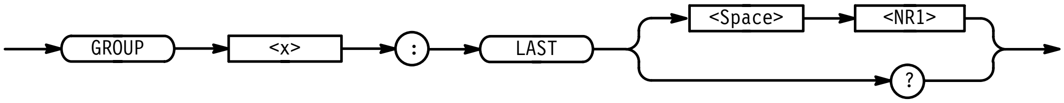
Group Vertical

Related Commands GROUP<x>:FIRst, GROUP<x>:SELect

Syntax

```
GROUP<x>:LAST <NR1>

GROUP<x>:LAST?
```



Arguments <NR1> specifies the last channel in the group from 1 to 16.

Examples `GROUP2:LAST 5`
sets the last channel in group 2 to channel 5.

`GROUP8:LAST?`
might return 16, indicating the last channel of group 8 is channel 16.

GROUP<x>:OFFSet

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

Group

Vertical

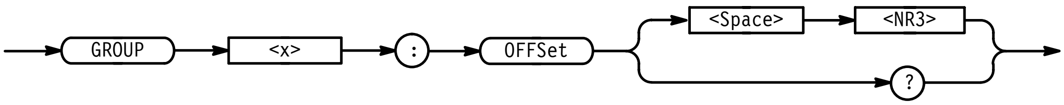
Related Commands

GROUP<x>:POSition

Syntax

```
GROUP<x>:OFFSet <NR3>
```

```
GROUP<x>:OFFSet?
```



Arguments

`<NR3>` is the desired offset in volts. The valid range for vertical offset is 15.0 ≥ Offset – 1 2.0 V, settable in 5 mV increments.

Examples

`GROUP1:OFFSET 0.5E+0`
lowers the group 1 displayed waveform by 0.5 volts.

`GROUP1:OFFSET?`
might return `500.0E-3`, indicating that the current group 1 offset is 0.5 volts.

GROUP<x>:POSition

Sets or queries the vertical position of the specified group. The position value is applied to the signal before digitization. Using this command is equivalent to setting Position in the Vertical menu or adjusting the front panel Vertical Position knob.

Group

Vertical

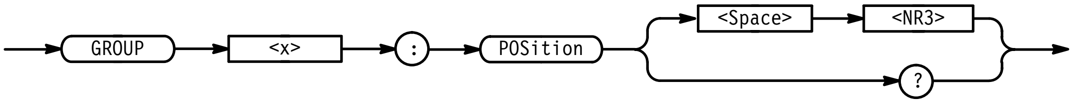
Related Commands

GROUP<x>:OFFSet

Syntax

```
GROUP<x>:POSition <NR3>
```

GROUP<x>:POSition?



Arguments

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

Examples

GROUP2:POSITION 1.3E+0
positions the group 2 input signal 1.3 divisions above the center of the display.

GROUP1:POSITION?
might return -1.3E+0, indicating that the current position of group 1 is at –1.3 divisions.

GROUP<x>:SCAle

Sets or queries the vertical gain of the specified group. Using this command is equivalent to adjusting the front panel Vertical SCALE knob. The setting for group scale determines the vertical height of any waveform group displayed in Analog display mode. It also scales a waveform group before its processed for display in BusForm and Timing Diagram display modes; however, the group scale setting does not determine the height of BusForm and Timing Diagram waveforms on screen. See the command description for GROUP<x>:HEIght on page 2–129 of this section.

Group

Vertical

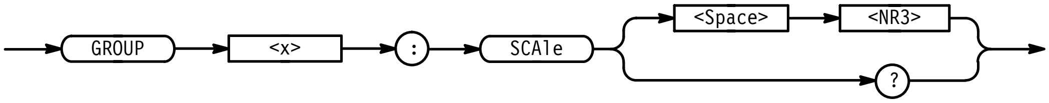
Related Commands

GROUP1:VOLts

Syntax

GROUP<x>:SCAle <NR3>

GROUP<x>:SCAle?



Arguments

<NR3> is the gain in volts per division. The range is 50 mV/div to 2 V/div.

Examples

`GROUP4:SCALE 100E-03`
sets the group 4 gain to 100 mV/div.

`GROUP2:SCALE?`
might return `1.00E+0`, indicating that the current V/div setting of group 2 is 1 V/div.

GROUP<x>:SELect

Sets or queries the selected channel within the group. Sending this command is equivalent to setting Selected Channel in the Group Definition menu. The logic scope highlights (makes brighter on screen) the selected channel in a group. When automated measurements are turned on, the logic scope takes them on the selected channel of the selected group. The logic scope also uses the selected channel when performing an Autoset. The selected channel must fall between the first channel and last channel, inclusively.

Group

Vertical

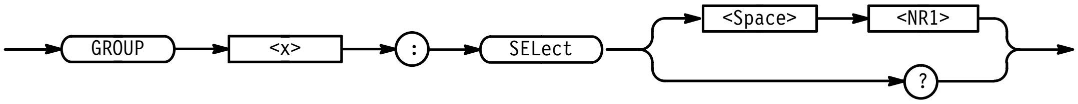
Related Commands

GROUP<x>:FIRst, GROUP<x>:LAST

Syntax

```
GROUP<x>:SELect <NR1>
```

```
GROUP<x>:SELect?
```



Arguments

`<NR1>` specifies the selected channel of the group from 1 to 16 and must be one of the channels in the specified in the group.

Examples

`GROUP2:SELECT 5`
specifies channel 5 as the selected channel for group 2.

`GROUP1:SELECT?`
might return channel 3 as the selected channel for group 1.

GROUP<x>:VHIgh

Sets or queries the upper threshold voltage (Vhigh min) for the specified group. Input signals in the group must cross this threshold going positive to be displayed at a HIGH digital level. This command affects groups set to BusForm or Timing Diagram display mode only, which display HIGH or LOW levels based on comparison to the set threshold levels.

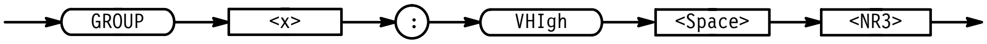
Group Vertical

Related Commands GROUP<x>:VLOw

Syntax

```
GROUP<x>:VHIgh <NR3>
```

```
GROUP<x>:VHIgh?
```



Arguments `<NR3>` is the upper threshold from –11.99E–0 to 15.00E+0 volts.

Examples

```
GROUP1:VHIGH 3.3E+0
```

sets the upper threshold voltage of group 1 to +3.3 volts.

```
GROUP2:VHIGH?
```

might return `3.3E+0` to indicate the upper threshold voltage is set to +3.3 volts for group 2.

GROUP<x>:VLOw

Sets or queries the lower threshold voltage (Vlow max) for the specified group. Input signals in the group must cross this threshold going negative to be displayed at a LOW digital level. This command affects groups set to BusForm or Timing Diagram display mode only, which display HIGH or LOW levels based on comparison to the threshold levels you set.

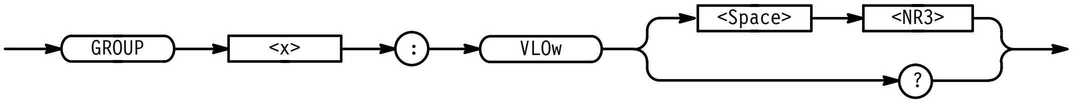
Group Vertical

Related Commands GROUP<x>:VHIGh

Syntax

```
GROUP<x>:VLOw <NR3>
```

```
GROUP<x>:VLOw?
```



Arguments <NR3> is the lower threshold from 12.00E–0 to 14.99E+0 volts.

Examples `GROUP1:VLOW 5.0E-1` sets the lower threshold voltage of group 1 to –0.5 volts.

`GROUP2:VLOW?`
might return `5.0E-1` to indicate the lower threshold voltage is set to –0.5 volts for group 2.

GROUP<x>:VOLts

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

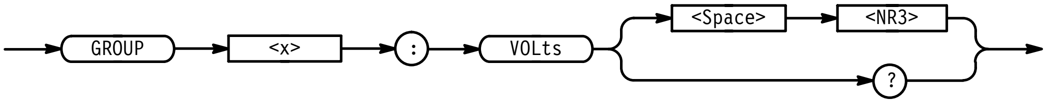
Group Vertical

Related Commands GROUP<x>:SCAle

Syntax

```
GROUP<x>:VOLts <NR3>
```

```
GROUP<x>:VOLts?
```



Arguments <NR3> is the vertical gain from 50 mV to 2 V/div.

Examples `GROUP4:VOLTS 100E-03`
sets the group 4 gain to 100 mV/div.

`GROUP2:VOLTS?`
might return `1.00E+0`, indicating that the current V/div setting of group 2 is 1 V/div.

HARDCopy

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

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

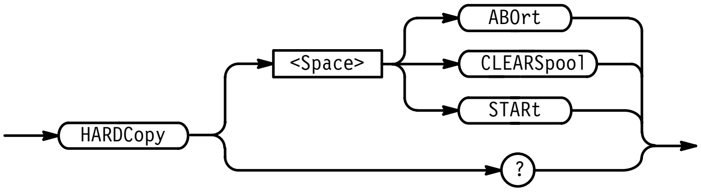
NOTE*. This command is NOT IEEE Std 488.2-1987 compatible.*

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. To abort the hardcopy process, send the HARDCopy ABOrt command. The output queue can then be cleared using DCL.*

`CLEARSpool` clears the printer output spool.

`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 initiating another.*

Examples

```
HARDCOPY ABORT
```

stops any hardcopy output that is in process.

HARDCOPY?
may return :HARDCOPY:FORMAT EPSCOLIMG;PORT GPIB;LAYOUT PORTRAIT;PALETTE HARDCOPY;FILENAME "fd0:/TEK?????.EPS".

HARDCopy:FILEName

Identifies the file to be sent when the next hardcopy command (i.e. HARDCOPY START command) is issued. Using this command is equivalent to setting the target file name in the Hardcopy menu.

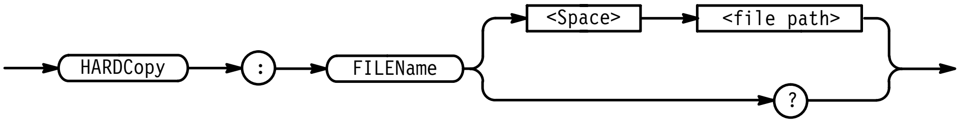
Group Hardcopy

Related Commands HARDCopy

Syntax

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

```
HARDCopy:FILEName?
```



Arguments <file path> is a quoted string defining the file name and path. Input the file path using the form <drive>/<dir>/<filename>; <drive> and one or more <dir>'s are optional. If the file path is not specified, the logic scope will write the file to the current directory. <filename> stands for a filename of up to 8 characters followed by a period (".") and any 3 character extension.

You can automatically create file names by using the question mark (?) as a wild card character. Wild card characters represent numbers the logic scope inserts sequentially in the filename. For example, if you placed two question marks at the end of the filename, the logic scope would append 00 to the first file created, 01 to the next, and 02 to the following. This is particularly useful in automated testing situations.

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

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

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

HARDCopy:FORMat

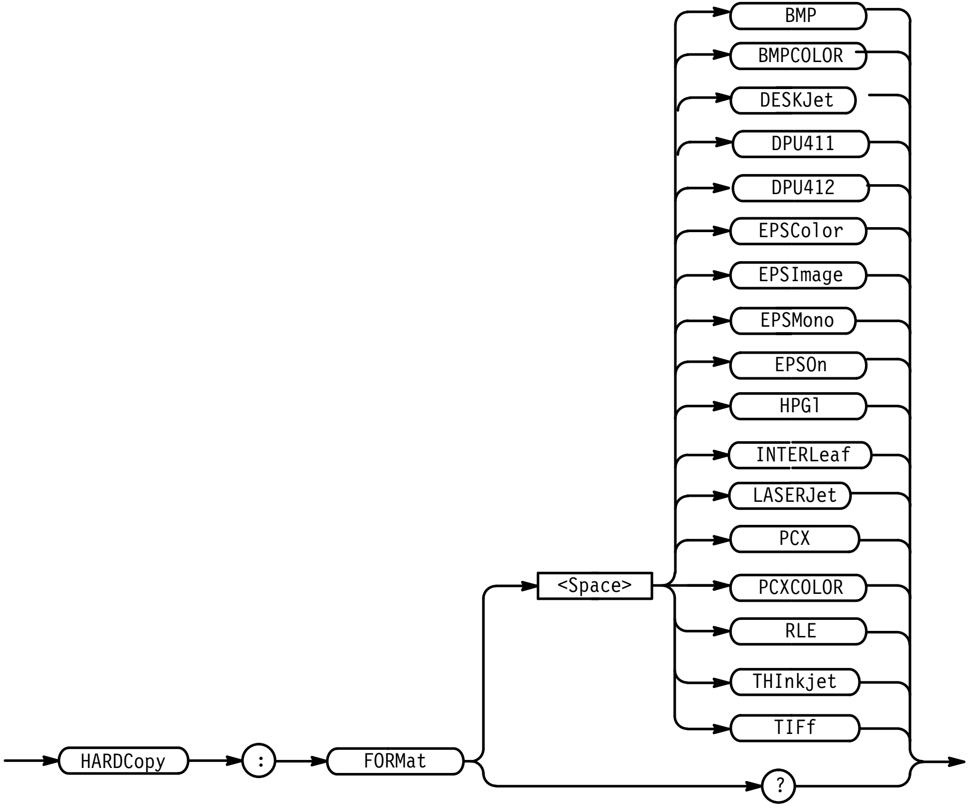
Selects the output data format for hard copies. Using this command is equivalent to setting Format in the Hardcopy menu.

Group Hardcopy

Syntax

```
HARDCopy:FORMat { BMP | BMPCOLOR | DESKJet | DPU411 | DPU412 |
EPSColor | EPSImage | EPSMono | EPSOn | HPGl | INTERLeaf |
LASERJet | PCX | PCXCOLOR | RLE | THInkjet | TIFf }
```

```
HARDCopy:FORMat?
```



Arguments `<BMP>` mono version of Microsoft Windows® file format.

`<BMPCOLOR>` color version of the Microsoft Windows® file format.

<DESKJet> HP Deskjet inkjet printer compatible format.

<DPU411> DPU-411/II portable thermal printer compatible format.

<DPU412> DPU-412 portable thermal printer compatible format.

<EPSColor> compatible format.

<EPSImage> Encapsulated Postscript, color-image, compatible format.

<EPSMono> Encapsulated Postscript, mono-image, compatible format.

<EPSon> Epson compatible format.

<HPGl> HP Graphics Language Color Plotter compatible format.

<INTERLeaf> Interleaf publishing software compatible format.

<LASERJet> HP Laserjet laser printer compatible format.

<PCX> PC Paintbrush® compatible format.

<PCXCOLOR> PC Paintbrush®, Color, compatible format.

<RLE> Microsoft Windows color image file format, compressed, compatible format.

<THInkjet> HP Thinkjet inkjet printer compatible format.

<TIFf> TIFF® (Tag Image File Format) compatible 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. Using this command is equivalent to setting Layout in the Hardcopy menu.

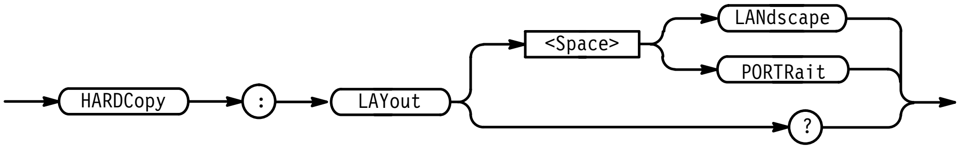
Group

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 shortest side of the page. This is the standard format.

Examples

```
HARDCOPY:LAYOUT LANDSCAPE
```
sets the printing orientation to landscape mode.

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

HARDCopy:PALEtte

Specifies which palette to use in creating the hardcopy: the current palette (as set in the Display menu and seen on the screen) or the hardcopy palette. For color hardcopies, the default hardcopy palette may be most appropriate since it has a white background and requires less ink for printing on white paper. For monochrome hardcopies, the logic scope ignores the palette and prints black (or the default color) objects on a blank background.

Group

Hardcopy

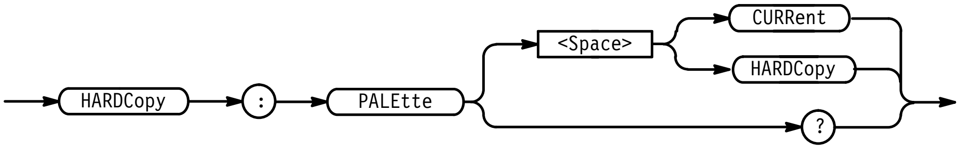
Related Commands

HARDCopy

Syntax

```
HARDCopy:PALEtte { CURRent | HARDCopy }
```

```
HARDCopy:PALEtte
```



Examples

```
HARDCOPY:PALETTE HARDCOPY
```
would print each copy using the Hardcopy palette.

`HARDCOPY:PALETTE?`
might return `HARDCOPY`.

HARDCopy:PORT

Specifies where to send hardcopy data at the next HARDCOPY START command. Using this command is equivalent to setting Port in the Hardcopy menu.

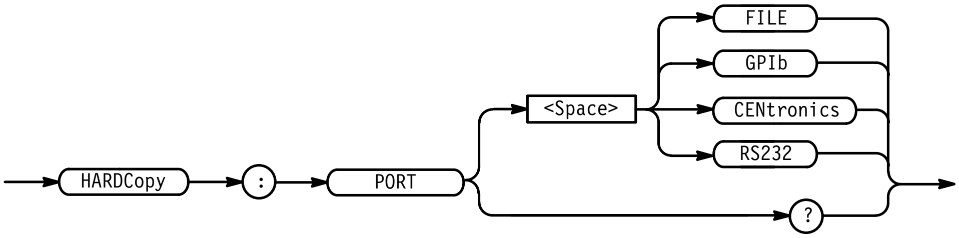
Group Hardcopy

Related Commands HARDCopy

Syntax

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

```
HARDCopy:PORT?
```



Arguments

`FILE` specifies that the hardcopy be stored in the file named by the HARDCOPY:FILENAME command.

`GPIb` specifies that the hardcopy be sent to the GPIB port.

`CENtronics` specifies that the hardcopy be sent to the Centronics port.

`RS232` specifies that the hardcopy be sent to the RS232 port.

Examples

`HARDCOPY:PORT FILE`
specifies that any hardcopy made be sent to the logic scope filesystem.

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

HDR

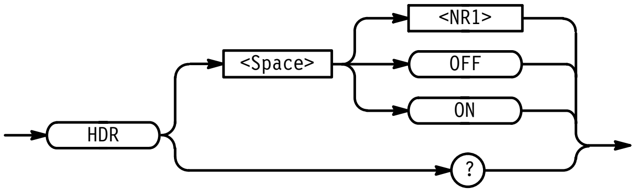
This command is identical to HEADer and is included for compatibility with older Tektronix instruments.

Group Miscellaneous

Syntax

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

HDR?
```



HEADer

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

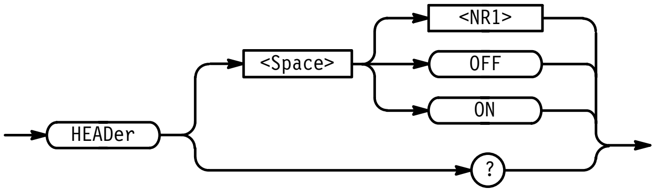
Group Miscellaneous

Related Commands VERBose

Syntax

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

HEADer?
```



Arguments ON or <NR1> $\neq$ 0 sets the Response Header Enable State to true. This causes the logic scope 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 logic scope to omit headers on query responses, only the argument is returned.

Examples

`HEADER OFF`
causes the logic scope to omit headers from query responses.

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

HORizontal? (Query Only)

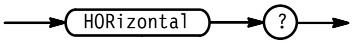
Returns all settings for the horizontal commands. The commands HORizontal:MAIn:SCAle, HORizontal:MAIn:SECdiv, HORizontal:SCAle, and HORizontal:SECdiv are equivalent. So HORizontal:MAIn:SCAle is the only value returned.

Group

Horizontal

Syntax

`HORizontal?`



Returns

See "Example"

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 HORizontal:DELay:SCAle only is returned.

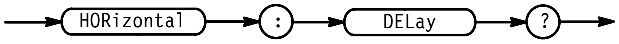
Group

Horizontal

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

Syntax

```
HORizontal:DELay?
```



Returns See “Example”

Examples

```
HORIZONTAL:DELAY?
```

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

HORizontal:DELay:MODe

Sets the delayed time base mode. Using this command is equivalent to setting Time Base in the Horizontal menu.

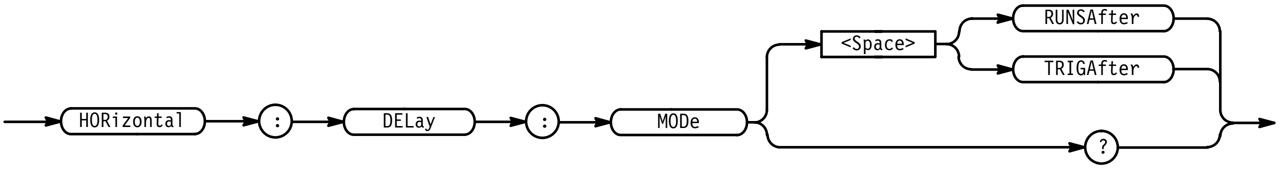
Group Horizontal

Related Commands HORizontal:DELay:TIMe

Syntax

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

HORizontal:DELay:MODe?
```



Arguments `RUNSAfter` instructs the delayed time base to issue a trigger automatically after a user-specified delay time (set by the `HORizontal:DELay:SCAle` command expires. The delay time is from the main trigger event.

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

Examples

`HORIZONTAL:DELAY:MODE RUNSAFTER` sets the delayed time base to runs after delay mode

`HORIZONTAL:DELAY:MODE?`
might return `TRIGAFTER`, which indicates the delayed time base is set to trigger on the first valid trigger occurring after the delay time elapses (delay time is from the main trigger event).

HORizontal:DELay:SCAle

Sets the time per division for the delayed time base. Using this command is equivalent to setting Delayed Scale in the Horizontal Scale side menu.

Group

Horizontal

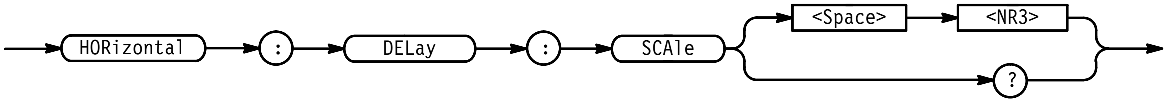
Related Commands

HORizontal:DELay:SECdiv

Syntax

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

```
HORizontal:DELay:SCAle?
```



Arguments

<NR3> is the time per division. The range is 5 s to 500 ps in a 1-2.5-5 sequence. Values not set in the specified sequence will be adjusted 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 1-2.5-5 sequence, it is automatically set to the closest valid value.

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

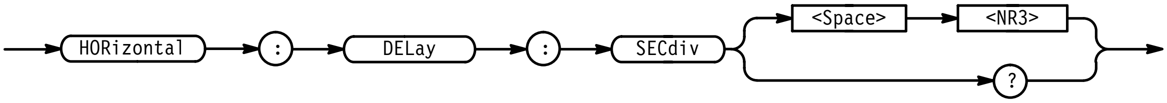
HORizontal:DELay:SECdiv

This command is identical to the HORizontal:DELay:SCAle command.

Group Horizontal

Syntax `HORizontal:DELay:SECdiv <NR3>`

`HORizontal:DELay:SECdiv?`



Arguments `<NR3>` is the time per division. The range is 5 s to 500 ps in a 1-2.5-5 sequence. Values not set to the specified sequence will be adjusted 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:SECdiv 2.0E-6`
sets the delay scale to 2 μs per division.

`HORIZONTAL:DELay:SECdiv 9.0E-6`
sets the delay scale to 10 μs per division. Since 9 μs is not a valid value within the 1-2.5-5 sequence, it is automatically set to the closest valid value.

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

HORizontal:DELay:TIMe

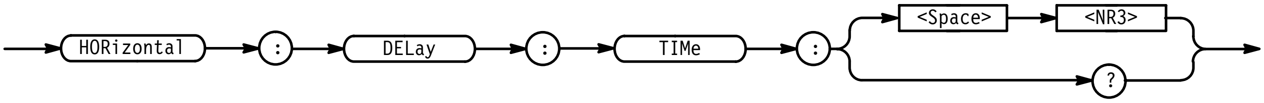
Sets or queries the delay time following the main trigger before the delayed time base triggers. Using this command is equivalent to setting Time Base to Delayed Runs After Main in the Horizontal menu.

Group Horizontal

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

Syntax `HORizontal:DELay:TIMe <NR3>`

```
HORizontal:DELay:TIMe?
```



Arguments <NR3> is the time in seconds, between the main trigger and the delayed trigger. The delay range is 16 ns to 250 seconds with a resolution of 2 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?`
might return `:HORIZONTAL:DELAY:TIME:16.0E-9` for the delay time.

HORizontal:DELay:TIMe:RUNSAfter

Sets or queries the delay time following the main trigger before the delayed time base begins. Using this command is equivalent to setting Time Base to Delayed Runs After Main in the Horizontal menu.

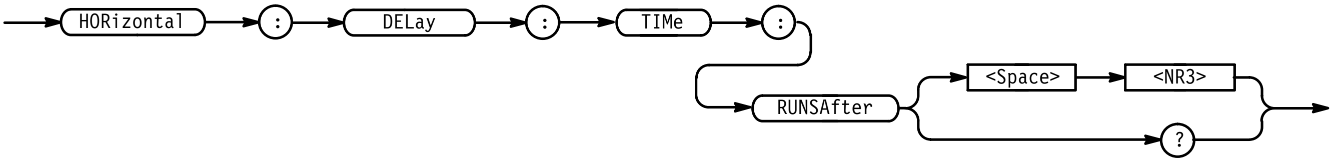
Group Horizontal

Related Commands HORizontal:DELay:MODe

Syntax

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

HORizontal:DELay:TIMe:RUNSAfter?
```



Arguments <NR3> is the time, in seconds, between when the logic scope receives a valid main trigger and when it forces the delayed trigger. The range is from 16 ns to 250 seconds with a resolution of 2 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:RUNSAFTER?`
might return `6.0E–6`, which indicates the delay time is set to 6.0 µS.

HORizontal:DELay:TIMe:TRIGAfter

Sets the delay time following a trigger before a delayed trigger is accepted. This command is equivalent to setting Delay by Time time in the Delayed Trigger menu.

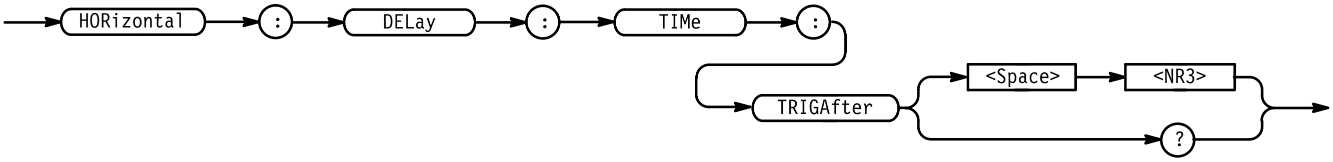
Group Horizontal

Related Commands HORizontal:DELay:MODe

Syntax

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

```
HORizontal:DELay:TIMe:TRIGAfter?
```



Arguments `<NR3>` is the time in seconds, between when the logic scope receives a valid main trigger, and when the delay trigger system enables recognition of a valid delay trigger event. The range is from 16 ns to 250 seconds with a resolution of 2 ns.

Examples

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

`HORIZONTAL:DELAY:TIME:TRIGAFTER?`
might return `6.0E–6`, which indicates the delay time is set to 6.0 µs.

HORizontal:FITtoscreen

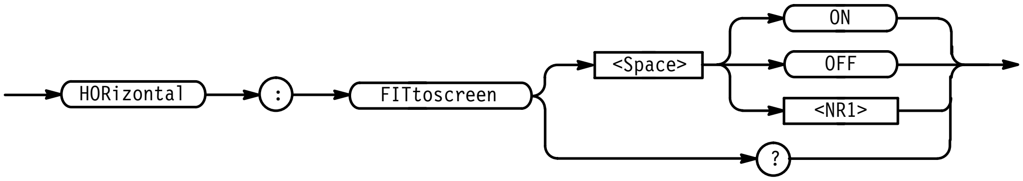
Sets the horizontal waveform compression operation. Using this command is equivalent to setting Record Length in the Horizontal menu and the side menu’s Fit to screen menu item. Waveform compress lets you fit a captured waveform to the visible screen. It provides the same functionality as if you were in zoom mode and altered the time/division until the waveform fit the screen.

Group Horizontal

Syntax

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

```
HORizontal:FITtoscreen?
```



Arguments `ON` turns on waveform compression.

`OFF` turns off waveform compression.

`<NR1>` 0 indicates waveform compression is OFF; a non-zero value indicates waveform compression is ON.

Examples

`HORIZONTAL:FITTOSCREEN ON`
turns on waveform compression.

`HORIZONTAL:FITTOSCREEN?`
might return `0`, indicating data is not fit to the screen.

HORizontal:MAIn? (Query Only)

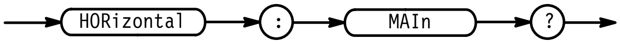
Returns the time per division of the main time base. The commands HORizontal:MAIn:SECdiv and HORizontal:MAIn:SCAle are identical, so only HORizontal:MAIn:SCAle will be returned.

Group Horizontal

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

Syntax

```
HORizontal:MAIn?
```



Returns `<NR3>`

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.

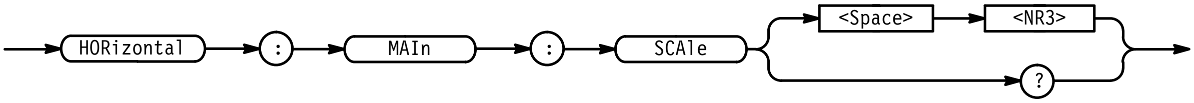
Group Horizontal

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. The range is 5 s to 500 ps in a 1-2.5-5 sequence. Values not set to the specified sequence will be adjusted to the closest valid value.

Examples

`HORIZONTAL:MAIN:SCALE 2.5E-6`
sets the main scale to 2.5 μs per division.

`HORIZONTAL:MAIN:SCALE?`
might return `5E-9`, which indicates the main time base is set to 5 nanoseconds.

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 logic scopes.

Group Horizontal

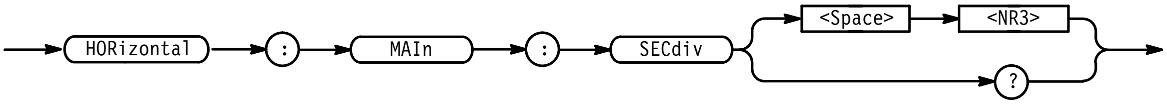
Related Commands

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

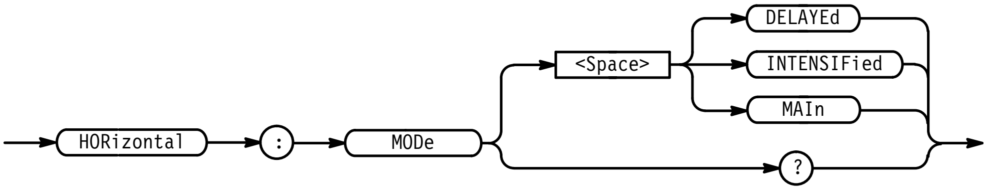
Related Commands

DISplay:INTENSITy:CONTRast

Syntax

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

```
HORizontal:MODe?
```



Arguments

DELAYEd specifies 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 specifies 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 the waveform is displayed using both the main and delayed time base scale.

HORizontal:POSition

Positions the waveform horizontally on the display. Used for the 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.

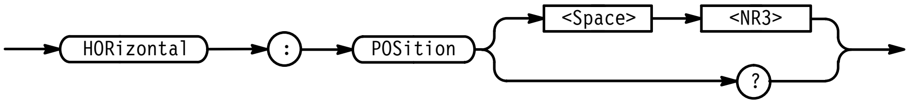
Group

Horizontal

Syntax

```
HORizontal:POSition <NR3>
```

```
HORizontal:POSition?
```



Arguments

`<NR3>` from 0 to 100, is the percentage 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:POSITION?`
might return 5.0E+1, which indicates that the waveform is position so 50% of the waveform record is to the left of center screen.

HORizontal:RECOrdlength

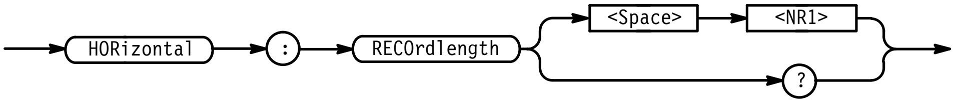
Sets the number of data points that are acquired for each record. Using this command is equivalent to setting Record Length in the Horizontal menu.

Group Horizontal

Syntax

```
HORizontal:RECOrdlength <NR1>

HORizontal:RECOrdlength?
```



Arguments `<NR1>` is 500, 1000, or 2000.

Examples

```
HORIZONTAL:RECORDLENGTH 1000
```

specifies that 1000 data points will be acquired for each record.

```
HORIZONTAL:RECORDLENGTH?
```

might return `2000` as the number of data points per record.

HORizontal:SCAle

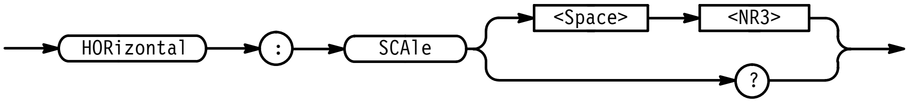
Sets or queries the time per division for the main time base and is identical to the HORizontal:MAIn:SCAle command. It is included here for compatibility with other Tektronix instruments.

Group Horizontal

Syntax

```
HORizontal:SCAle <NR3>

HORizontal:SCAle?
```



Arguments

<NR3> is the time per division. The range is 5 s to 500 ps in a 1-2.5-5 sequence. Values not set to the specified sequence will be adjusted to the closest valid value.

Examples

```
HORIZONTAL:MAIN 2.5E-6
```

sets the main scale to 2.5 µs per division.

```
HORIZONTAL:MAIN?
```

might return 5E-9, which indicates the main time base is set to 5 nanoseconds.

HORizontal:SECdiv

Sets or queries the time per division for the main time base and is identical to the HORizontal:MAIn:SCAle command. It is included here for compatibility with other Tektronix instruments.

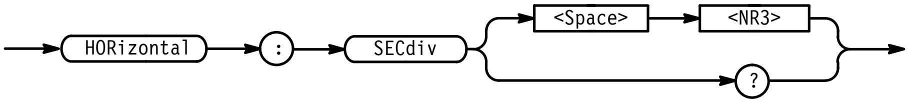
Group

Horizontal

Syntax

```
HORizontal:SECdiv <NR3>

HORizontal:SECdiv?
```



Arguments

<NR3> is the time per division. The range is 5 s to 500 ps in a 1-2.5-5 sequence. Values not set to the specified sequence will be adjusted to the closest valid value.

Examples

```
HORIZONTAL:SECDIV 2.5E-6
```

sets the main scale to 2.5 µs per division.

```
HORIZONTAL:SECDIV?
```

might return 5E-9, which indicates the main time base is set to 5 nanoseconds.

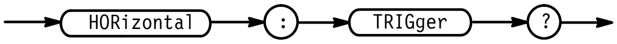
HORizontal:TRIGger? (Query Only)

Returns the horizontal trigger parameters.

Group Horizontal

Syntax

```
HORizontal:TRIGger?
```



Examples

```
HORIZONTAL:TRIGGER?
```

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

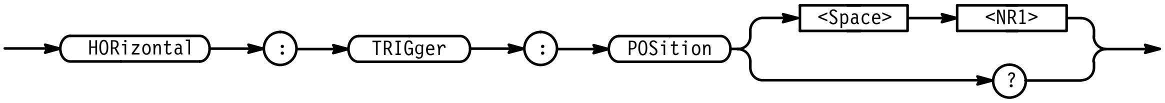
HORizontal:TRIGger:POSition

Sets or queries the trigger position. Using this command is equivalent to setting Trigger Position in the Horizontal menu.

Group Horizontal

Syntax `HORizontal:TRIGger:POSition <NR1>`

`HORizontal:TRIGger:POSition?`



Arguments `<NR1>` is the amount of pretrigger information in the waveform from 20 to 80%.

Examples `HORIZONTAL:TRIGGER:POSITION 40`
sets the amount of pretrigger to 40% of the waveform record.

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

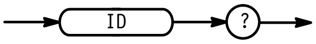
ID? (Query Only)

Returns instrument identification and firmware information.

Group Status and Error

Related Commands *IDN?

Syntax `ID?`



The instrument id in the following format:

```
TEKTRONIX,<model number>,CF:92.1CT,FV:<firmware version
number>
```

Examples `ID?`
might return `TEK,TLS216,CF:92.1CT,FV:2.0`.

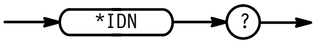
*IDN? (Query Only)

Returns the logic scope's unique identification code.

Group Miscellaneous

Related Commands ID

Syntax `*IDN?`



Returns The instrument id in the following format:

```
TEKTRONIX,<model number>,0,CF:92.1CT FV:<firmware version
number>
```

Examples `*IDN?`
might return
`TEK,TLS216,0,CF:92.1CT FV:2.0`.

LOCk

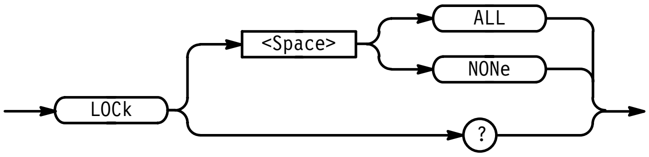
Enables and disables all front panel buttons and knobs. There is no front panel equivalent.

Group Miscellaneous

Related Commands UNLock, Remote Enable Group, Local Lockout Group

Syntax `LOCk { ALL | NONe }`

`LOCk?`



Arguments

`ALL` disables all front panel controls.

`NONe` enables all front panel controls. Using this argument is equivalent to the UNLock ALL command.

***NOTE**. If the logic scope 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 logic scope settings, except for configuration information for the calibration values. You can use this string to return the logic scope to the state it was in when you made the *LRN? query.

Group

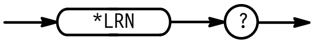
Miscellaneous

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. The returned string is intended to be sent back to the logic scope 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:

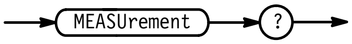
```
:ACQUIRE:STATE 1;MODE SAMPLE;NUMENV 10;NUMAVG 16;
REPET 1;STOPAFTER RUNSTOP;:DIAG:LOOP:OPTION ONCE;
COUNT 1;:DIAG:STATE HALT;:HEADER 1;:VERBOSE 1;
:CURSOR:FUNCTION OFF;VBARS:UNITS SECONDS;
POSITION1 1.00E-6;POSITION2 9.00E-6;SELECT CURSOR1;.
```

MEASUrement? (Query Only)

Returns all measurement parameters for the four displayed measurements and the immediate measurement the logic scope provides.

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 FOR-
WARDS;:MEASUREMENT:METHOD HISTOGRAM;REFLEVEL:METHOD PERCENT;
ABSOLUTE:HIGH 0.0E+0;LOW 0.0E+0;MID 0.0E+0; MID2.0.0E+0;
```

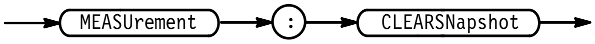
```
:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0E+0;LOW 10.0E+0;MID
50.0E+0;MID2 50.0E+0.
```

MEASUrement:CLEARSNapshot (No Query Form)

Clears (removes) the measurement snapshot display.

Group Measurement

Syntax `MEASUrement:CLEARSNapshot`



MEASUrement:GATing

Sets or queries measurement gating.

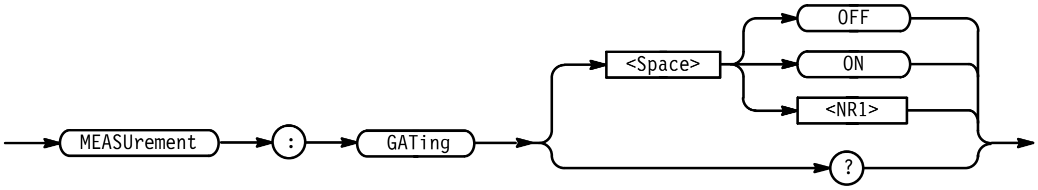
Group Measurement

Related Commands CURSor:VBARS

Syntax

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

MEASUrement:GATing?
```



Arguments OFF (or 0) turns off measurement gating.

ON (or 1) turns on measurement gating.

NR1 is 0 which turns off, or 1 which turns on, measurement gating.

Examples `MEASUREMENT:GATING ON`

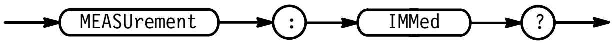
`MEASUREMENT:GATING?`
might return `MEASUREMENT:GATING 1`, indicating gating is turned on.

MEASUrement:IMMed? (Query Only)

Returns all setup parameters for the immediate measurement. IMMed specifies the immediate measurement that is taken without displaying the measurement results on screen.

Group Measurement

Syntax `MEASUrement:IMMed?`



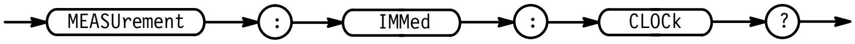
Examples `MEASUREMENT:IMMED?`
might return `:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s"; SOURCE1 CH1;SOURCE2 CH1;DELAY:EDGE1 RISE;EDGE2 RISE; DIRECTION FORWARDS.`

MEASUrement:IMMed:CLOCk? (Query Only)

Returns the current setup (source, direction, and so on) for the clock associated with the immediate measurement. (The immediate measurement is taken without displaying the results on screen.) The clock specification only affects setup or hold measurement types.

Group Measurement

Syntax `MEASUrement:IMMed:CLOCk?`



Examples `MEASUREMENT:IMMED:CLOCK?`
might return `CH1.`

MEASUrement:IMMed:CLOCk:DIREction

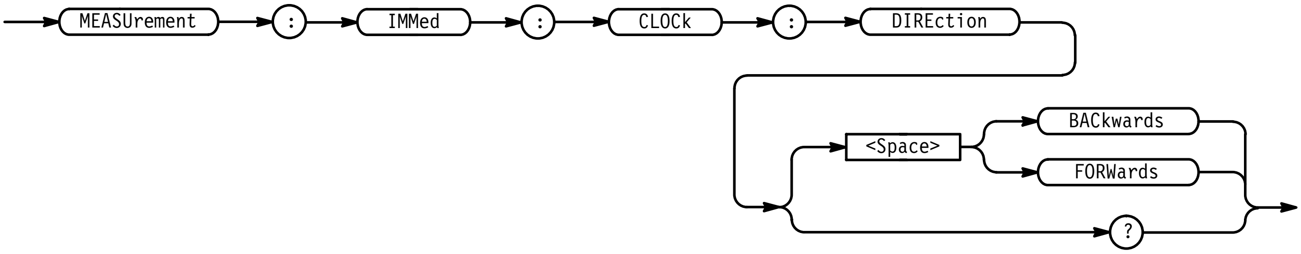
Sets or queries the starting point and direction of the clock edge when taking a setup or hold immediate measurement. (The immediate measurement is taken without displaying the results on screen.) The clock waveform only affects setup or hold immediate measurement types.

Group Measurement

Syntax

```
MEASUrement:IMMed:CLOCk:DIREction { FORWards | BACKWards }
```

```
MEASUrement:IMMed:CLOCk:DIREction?
```



Arguments `BACkwards` initiates the search from 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:CLOCK:EDGE.

`FORWards` initiates the search from 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:CLOCK:EDGE.

Examples

```
MEASUREMENT:IMMED:CLOCK:DIRECTION FORWARDS
```

starts searching from the beginning of the waveform record.

```
MEASUREMENT:IMMED:CLOCK:DIRECTION?
```

returns either `BACkwards` or `FORWARDS`.

MEASUrement:IMMed:CLOCk:EDGE

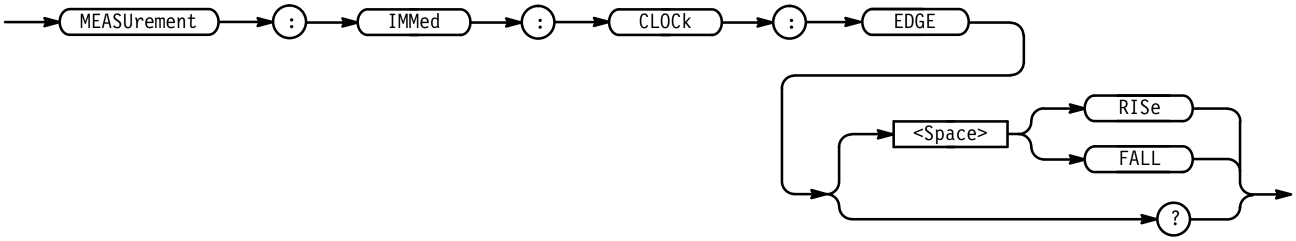
Sets or queries the clocking-edge slope of the clock waveform for the immediate measurement. (The immediate measurement is taken without displaying the results on screen.) The clock waveform only affects setup or hold immediate measurement types. The clock is specified by the MEASUREMENT:IMMed:CLOCk:SOUrce command.

Group Measurement

Syntax

```
MEASUrement:IMMed:CLOCk:EDGE { RISe | FALL }
```

```
MEASUrement:IMMed:CLOCk:EDGE?
```



Arguments `RISe` specifies the rising edge.

`FALL` specifies the falling edge.

Examples

`MEASUREMENT:IMMED:CLOCK:EDGE RISE`
which specifies that the rising edge be used for the clock waveform when taking a Setup or Hold Immediate measurement.

`MEASUREMENT:IMMED:CLOCK:EDGE?`
might return `FALL`, which indicates the logic scope currently uses the falling edge of the clock for taking a Setup or Hold Immediate measurement.

MEASUrement:IMMed:CLOCk:SOUrce

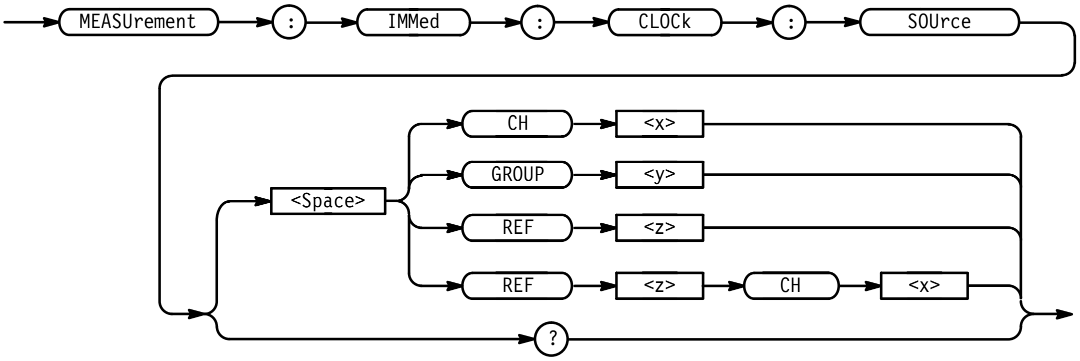
Sets or queries the source of the clock waveform for the immediate measurement. (The immediate measurement is taken without displaying the results on screen.) The clock waveform only affects setup or hold immediate measurement types.

Group Measurement

Syntax

```
MEASUrement:IMMed:CLOCk:SOUrce { CH<x> | GROUP<y> | REF<z> |
REF<z>CH<x> }
```

```
MEASUrement:IMMed:CLOCk:SOUrce?
```



Arguments

`CH<x>` specifies that channel <x> be used as the waveform for the clock source.

`GROUP<y>` specifies that the channel currently selected in group <y> be used as the waveform for the clock source. The command `GROUP<x>:SELECT`, described on page 2–133, sets the selected channel.

`REF<z>` specifies that the currently selected channel in reference group <z> be used as the waveform for the clock source. The command `REF<x>:SELECT`, described on page 2–212, sets the selected channel.

`REF<z>CH<x>` specifies that channel <x> in reference group <z> to be used as the waveform for the clock source.

Examples

`MEASUREMENT:IMMED:CLOCK:SOURCE CH16`

`MEASUREMENT:IMMED:CLOCK:SOURCE?`
might return `REF1CH1`, indicating the first channel of the first reference group is the clock source.

MEASUrement:IMMed:DELay? (Query Only)

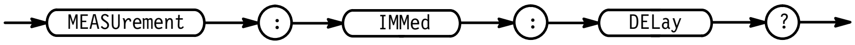
Returns the Delay measurement parameters for the immediate measurement. (The immediate measurement is taken without displaying the results on screen.) The delay parameters only affect delay type immediate measurements.

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 of the "delay to" edge when taking a delay immediate measurement. (The immediate measurement is taken without displaying the results on screen.)

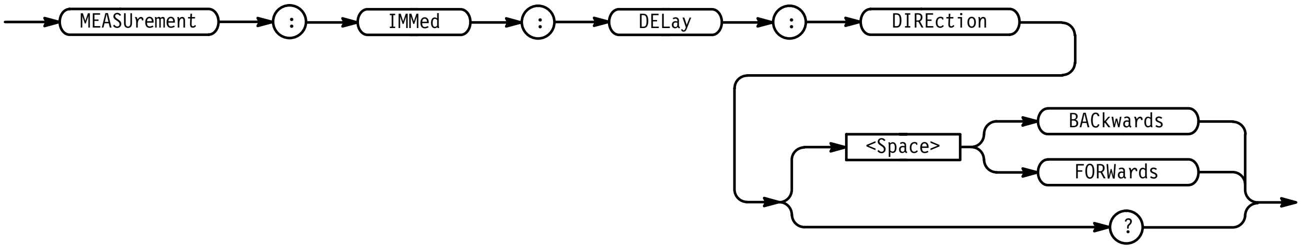
Use the MEASUrement:IMMed:SOURCE2 command to specify the delay "to" waveform. The delay direction parameter only affects immediate measurements of delay.

Group Measurement

Syntax

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

```
MEASUrement:IMMed:DELay:DIREction?
```



Arguments `BACkwards` starts the search 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` starts the search 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 used for the delay “from” waveform when taking an delay immediate measurement. (The immediate measurement is taken without displaying the results on screen.)

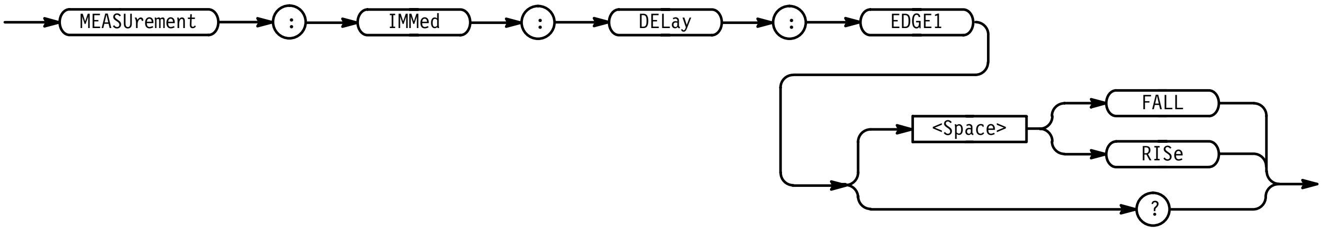
Use the MEASUrement:IMMed:SOURCE1 command to specify the delay “from” waveform. The delay slope parameter only affects immediate measurements of delay.

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 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 used for the delay “to” waveform when taking an immediate measurement of delay. (The immediate measurement is taken without displaying the results on screen.)

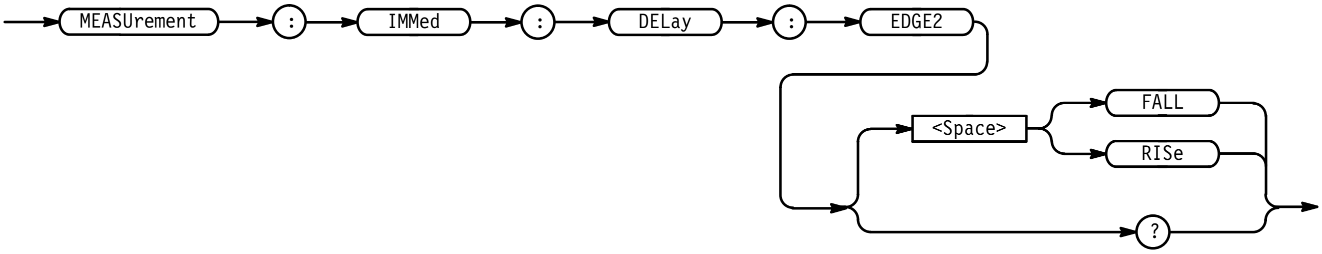
Use the MEASUrement:IMMed:SOURCE2 to specify the “delay to” waveform. The delay slope parameter only affects immediate measurements of delay.

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.

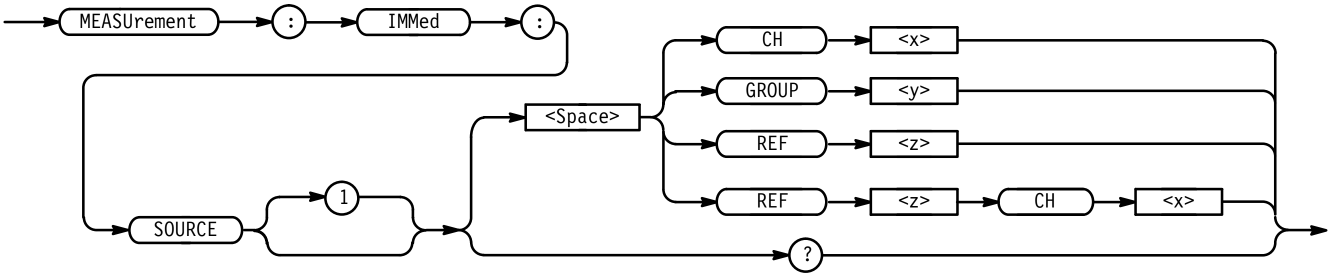
MEASUrement:IMMed:SOURCE[1]

Sets or queries the source of the waveform to be measured for all types of immediate measurements that can be taken on a single waveform. Also sets or queries the source of the waveform to be measured “from” for all immediate measurements of delay or phase.

Group Measurement

Syntax `MEASUrement:IMMed:SOURCE[1] { CH<x> | GROUP<y> | REF<z> | REF<z>CH<x> }`

`MEASUrement:IMMed:SOURCE[1]?`



Arguments

`CH<x>` specifies that channel <x> be the waveform source for the immediate measurement.

`GROUP<y>` specifies that the channel currently selected in group <y> be the waveform source for the immediate measurement. The command `GROUP<x>:SELECT`, described on page 2–133, sets the selected channel.

`REF<z>` specifies that the currently selected channel in reference group <z> be the waveform source for the immediate measurement. The command `REF<x>:SELECT`, described on page 2–212, sets the selected channel.

`REF<z>CH<x>` specifies that channel <x> in reference group <z> to be used as the waveform source for the immediate measurement.

Examples

`MEASUREMENT:IMMED:SOURCE CH1`
specifies channel 1 as the immediate measurement source.

`MEASUREMENT:IMMED:SOURCE1?`
might return `REF1CH1`, indicating the first channel of the first reference group is the clock source.

MEASUrement:IMMed:SOURCE2

Specifies the source to measure "to" when taking an Immediate Delay or Phase measurement.

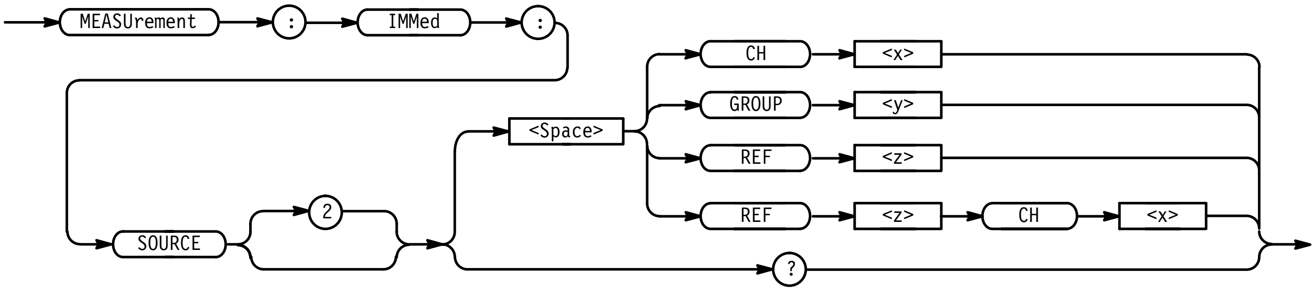
Group

Measurement

Syntax

```
MEASUrement:IMMed:SOURCE2 { CH<x> | GROUP<y> | REF<z> |
REF<z>CH<x> }
```

```
MEASUrement:IMMed:SOURCE2?
```



Arguments

CH<x> specifies that channel <x> be waveform source for the immediate measurement.

GROUP<y> specifies that the channel currently selected in group <y> be the waveform source for the immediate measurement. The command GROUP<x>:SELECT, described on page 2–133, sets the selected channel.

REF<z> specifies that the currently selected channel in reference group <z> be the waveform source for the immediate measurement. The command REF<x>:SELECT, described on page 2–212, sets the selected channel.

REF<z>CH<x> specifies that channel <x> in reference group <z> to be used as the waveform source for the immediate measurement.

Examples

```
MEASUREMENT:IMMED:SOURCE2 REF3
```
sets the selected channel of reference group 3 as the waveform source for making delay measurements.

```
MEASUREMENT:IMMED:SOURCE2?
```
might return REF1CH1, indicating the first channel of the first reference group is the clock source.

MEASUrement:IMMed:TYPe

Specifies or returns the format of the Immediate measurement to be performed.

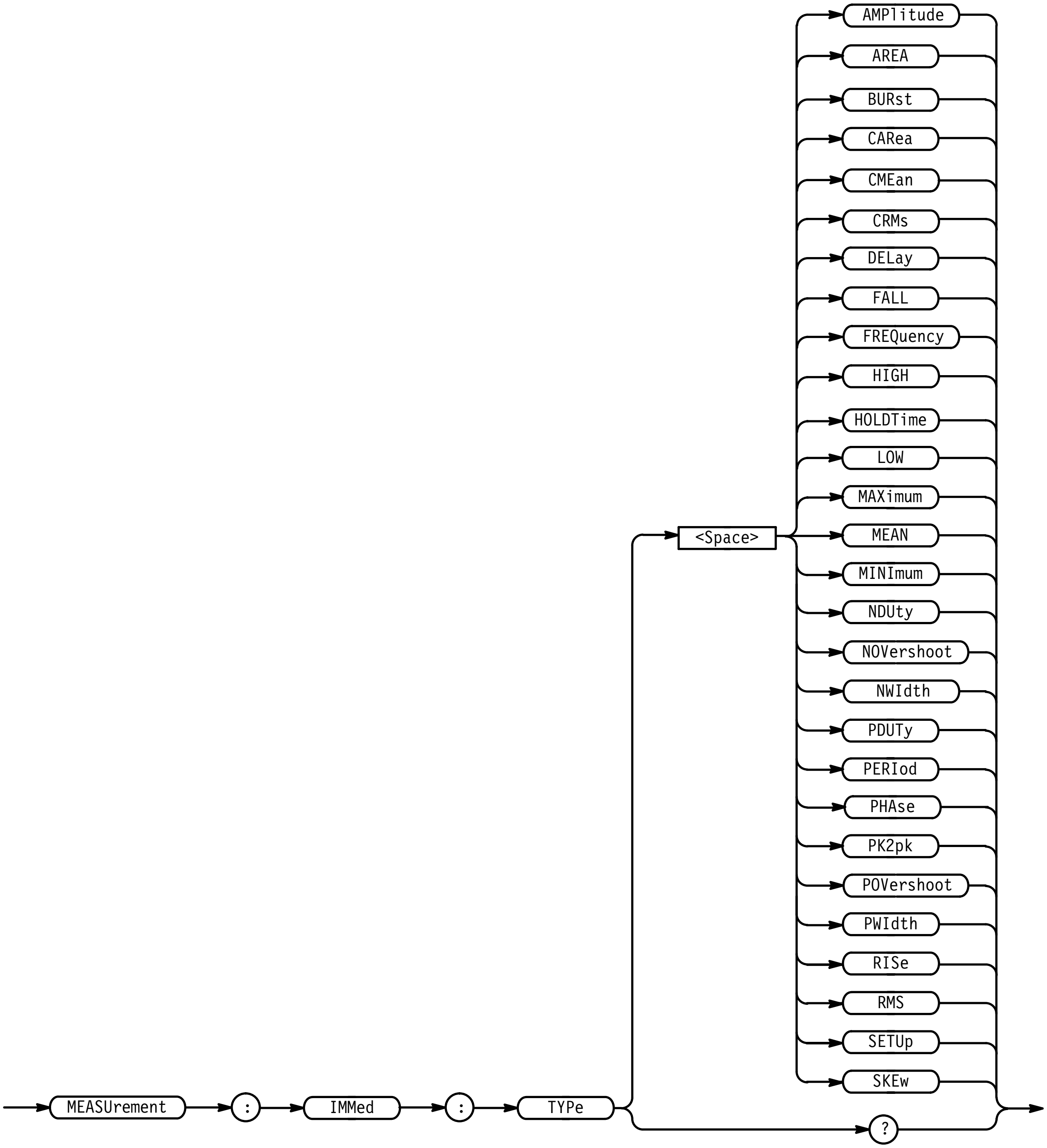
Group

Measurement

Syntax

```
MEASUrement:IMMed:TYPe { AMPlitude | AREA | BURst | CARea | CMEan
| CRMs | DELay | FALL | FREQuency | HIGH | HOLDTime | LOW |
MAXimum | MEAN | MINImum | NDUty | NOVershoot | NWIdth | PDUTy |
PERIod | PHAse | PK2pk | POVershoot | PWIdth | RISe | RMS |
SETUp | SKEw }
```

```
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.

HOLDTime is time between the first transition to an illegal state for all waveforms in a group and the clocking edge of the waveform in the channel specified as the clock. The clock can be any channel. It does not have to be included in the measured group. Hold measurements define “illegal state” as any level between the low reference the high reference levels. These levels can be queried or set using the MEASUREMENT:REFLEVEL commands that begin on page 2–189.

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.

`SETUp` is the time between the last transition to a legal state for all waveforms in a group and the clocking edge of the waveform specified as the clock. The clock can be any channel. It does not have to be included in the measured group. Setup measurements define "legal state" as a level either below the low reference level or above the high reference level. These levels can be queried or set using the MEASUREMENT:REFLEVEL commands that begin on page 2–189.

`SKEw` is the time between the first transition to an illegal state for all waveforms in a group and last transition to a legal state for the same waveforms. Skew measurements define "legal state" as a level either below the low reference level or above the high reference level. They define an "illegal state" as a level between the the low and high reference levels. These levels can be queried or set using the MEASUREMENT:REFLEVEL commands that begin on page 2–189.

Examples

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

`MEASUREMENT:IMMED:TYPE?`
might return `RMS` indicating the logic scope is set to take an immediate measurement of a true root mean square voltage.

MEASUrement:IMMed:UNIts? (Query Only)

Returns the units for the Immediate measurement.

Group

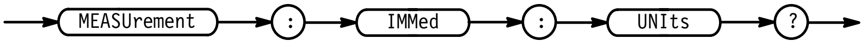
Measurement

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.

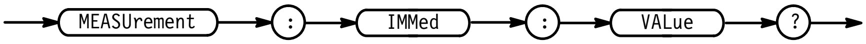
Examples `MEASUREMENT:IMMED:UNITS?`
might return `"s"`, indicating the units of the Immediate measurement are in seconds.

MEASUrement:IMMed:VALue? (Query Only)

Executes the Immediate measurement specified by the MEASUrement:IMMed:TYPe command. The measurement is taken on the source(s) specified by the MEASUrement:IMMed:SOURCE command.

Group Measurement

Syntax `MEASUrement:IMMed:VALue?`



Examples `MEASUrement:IMMed:VALue?`
might return `1.0E-3`, indicating a rise time measurement of 1 μs.

MEASUrement:MEAS<x>? (Query Only)

Returns all measurement parameters for the displayed measurement specified by MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through 4), where <x> specifies 1 through 4.

Group Measurement

Syntax `MEASUrement:MEAS<x>?`

MEASUrement
:
MEAS
<x>
?

Examples `MEASUREMENT:MEAS3?`

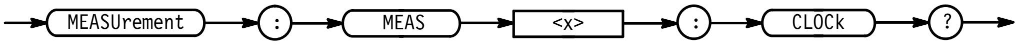
might return `:MEASUREMENT:MEAS3:STATE 0;TYPE PERIOD; UNITS "s";SOURCE1 CH1;SOURCE2 CH2;DELAY:EDGE1 RISE; EDGE2 RISE;DIRECTION FORWARDS`, indicating the type (period) of measurement and other parameters associated with measurement 3.

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

Returns the current setup (source, direction, and so on) for the clock associated with the measurement specified by MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through 4) where <x> specifies 1 through 4. The clock specification only affects setup or hold measurement types.

Group Measurement

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



Examples `MEASUREMENT:MEAS<x>:CLOCK?`

might return `CH16.`

MEASUrement:MEAS<x>:CLOCk:DIRection

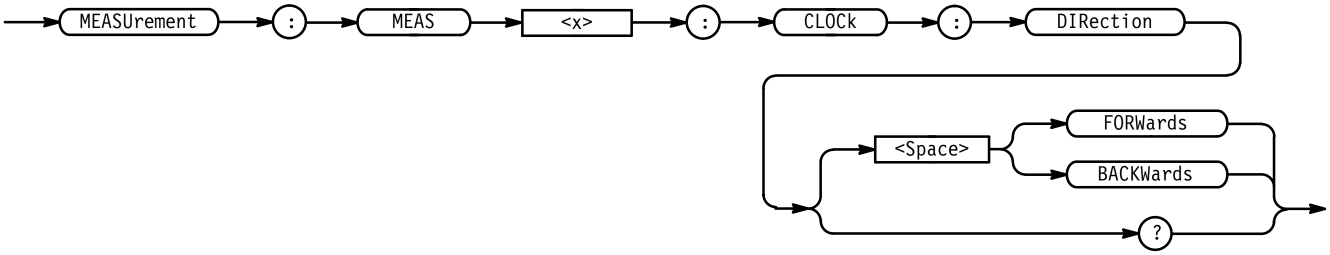
Sets or queries the starting point and direction of the clock edge when taking a setup or hold measurement for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through 4) where <x> specifies 1 through 4. The clock waveform only affects setup or hold measurement types. This command is equivalent to setting the direction in the Clock Edges and Direction side menu.

Group Measurement

Syntax

```
MEASUrement:MEAS<x>:CLOCk:DIRection { FORWards | BACKWards }
```

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



Arguments

`BACkwards` initiates the search from 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>:CLOCk:EDGE.

`FORWards` initiates the search from 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>:CLOCk:EDGE.

Examples

`MEASUREMENT:MEAS<x>:CLOCK:DIRECTION FORWARDS`
initiates a search from the beginning of the waveform record.

`MEASUREMENT:MEAS<x>:CLOCK:DIRECTION?`
returns either `BACkwards` or `FORWARDS`.

MEASUrement:MEAS<x>:CLOCk:EDGe

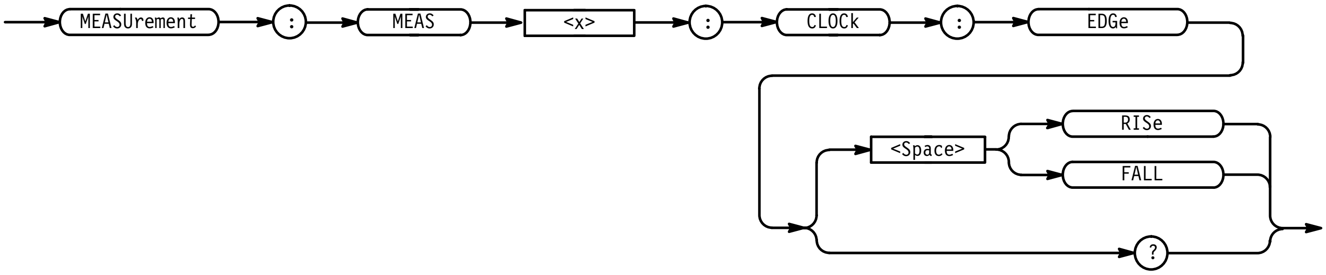
Sets the clocking-edge slope of the clock waveform for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4), where <x> specifies 1 through 4. The clock waveform only affects setup or hold measurement types. Use the MEASUREMENT:MEAS<x>:CLOCK:SOURCE command to select the clock source. This command is equivalent to setting the clock edge in the Clock Edges and Direction side menu.

Group Measurement

Syntax

```
MEASUrement:MEAS<x>:CLOCk:EDGe { RISe | FALL }
```

```
MEASUrement:MEAS<x>:CLOCk:EDGe?
```



Arguments `RISe` specifies the rising edge.

`FALL` specifies the falling edge.

Examples

```
MEASUREMENT:MEAS<x>:CLOCK:EDGE RISE
```

specifies the rising clocking-edge slope of the clock waveform for MEAS<x>.

```
MEASUREMENT:MEAS<x>:CLOCK:EDGE?
```

might return `RISE`.

MEASUrement:MEAS<x>:CLOCk:SOUrce

Sets or queries the source of the clock waveform for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4), where <x> specifies 1 through 4. The clock source only affects setup or hold measurement types. This command is equivalent to setting the clock source in the Source Setup side menu.

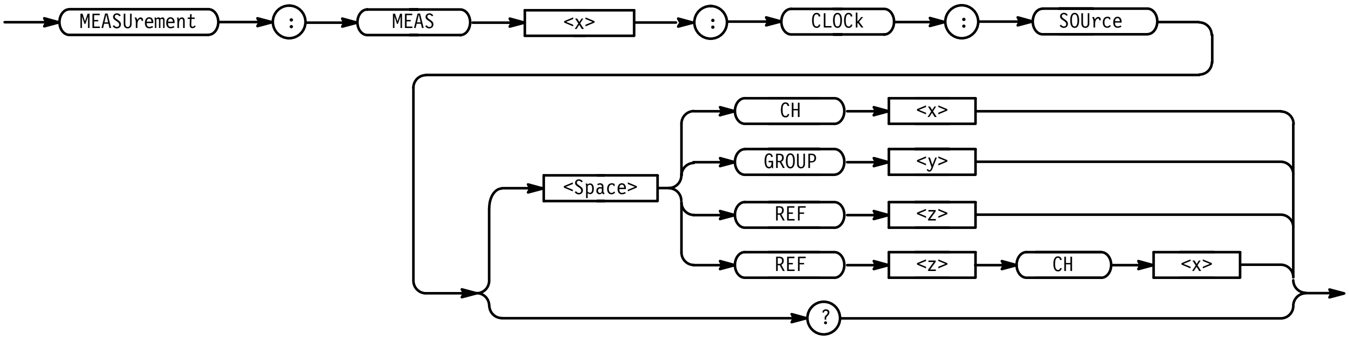
Group

Measurement

Syntax

```
MEASUrement:MEAS<x>:CLOCk:SOUrce { CH<x> | GROUP<y> | REF<z> |
REF<z>CH<x> }
```

```
MEASUrement:MEAS<x>:CLOCk:SOUrce?
```



Arguments

CH<x> specifies that waveform <x> is the waveform selected as the clock source. If CH<x> is specified, the channel must be included in a group.

GROUP<y> specifies that the channel currently selected in group <y> is the waveform selected as the clock source. The command GROUP<x>:SELECT, described on page 2–133, sets the selected channel.

REF<z> specifies that the currently selected channel in reference group <z> is the waveform selected as the clock source. The command REF<x>:SELECT, described on page 2–212, sets the selected channel.

REF<z>CH<x> specifies that channel <x> in reference group <z> is the waveform selected as the clock source.

NOTE*. If a reference waveform is specified as a source, the waveform must have a display type of ANALOGALL, ANALOGSET, BUS, or TIMING.*

Examples

```
MEASUREMENT:MEAS<x>:CLOCK:SOURCE REF<z>
```

specifies that the waveform in the currently selected channel in reference group <z> is the clock source.

```
MEASUREMENT:MEAS<x>:CLOCK:SOURCE?
```

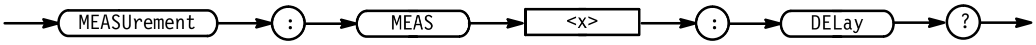
might return REF2CH2.

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

Returns the Delay measurement parameters for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4) where <x> specifies 1 through 4. The delay parameters only affect delay type measurements.

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 of the “delay to” edge when taking a delay measurement for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4) where <x> specifies 1 through 4.

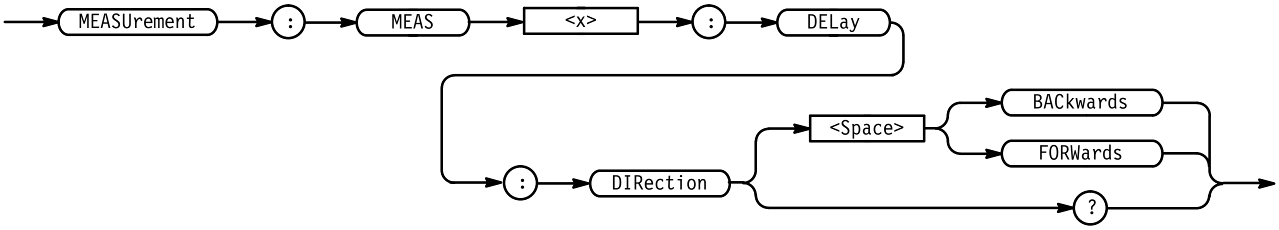
The waveform for the “delay to” edge is specified by the command MEASUrement:MEAS<X>:SOURCE2. This command is equivalent to setting the direction in the Delay Edges & Direction side menu. The delay direction parameter only affects measurements of delay.

Group Measurement

Syntax

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

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



Arguments BACkwards means 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 the command MEASUrement:MEAS<x>:DELay:EDGE2.

FORWards means 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 the command 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

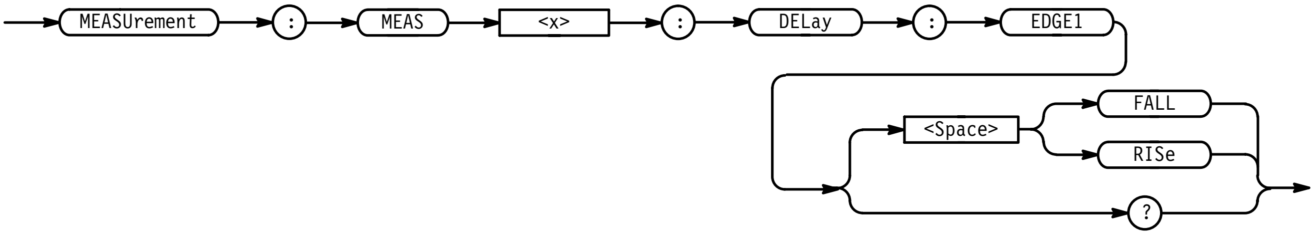
Sets or queries the slope of the edge used for the delay “from” waveform taking a delay measurement for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4), where <x> specifies 1 through 4.

Use the command MEASUrement:MEAS<X>:SOURCE1 to specify the “delay from” edge. This command is equivalent to setting the slope in the Delay Edges & Direction side menu. The delay direction parameter only affects measurements of delay.

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 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

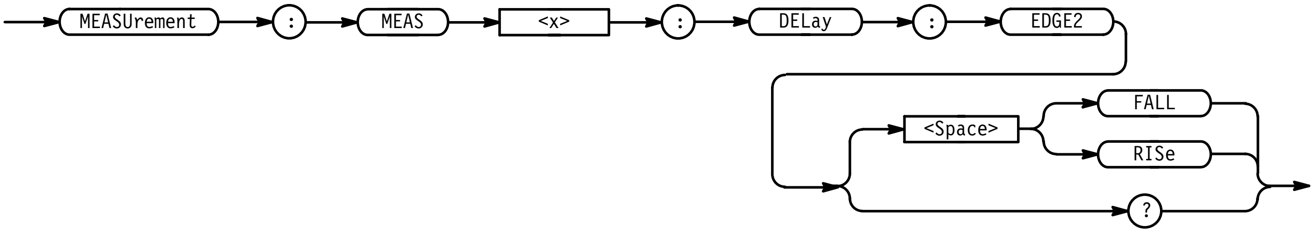
Sets or queries the slope of the edge used for the delay “to” waveform when taking a delay measurement for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4), where <x> specifies 1 through 4. This command is equivalent to selecting the edges in the Delay Edges & Direction side menu.

Use the command MEASUrement:MEAS<X>:SOURCE2 to specify the “delay to” waveform.

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 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 of the waveform to be measured for all types of measurements on a single waveform for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4) where <x> specifies 1 through 4.

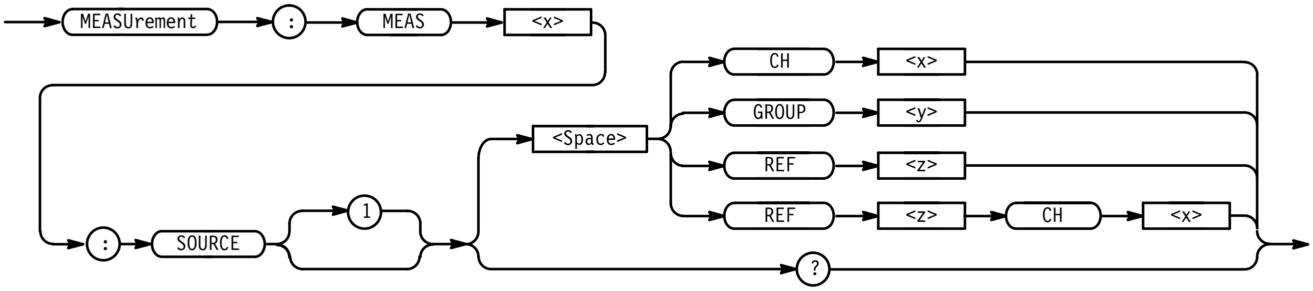
Also sets or queries the source of the waveform to be measured “from” for all delay or phase measurements for MEAS<x>.

Group Measurement

Syntax

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

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



Arguments

`CH<x>` specifies that channel <x> is the waveform source for the measurement.

`GROUP<y>` specifies that the channel currently selected in group <y> is the waveform source for the measurement. The command `GROUP<x>:SELECT`, described on page 2–133, sets the selected channel.

`REF<z>` specifies that the currently selected channel in reference group <z> is the waveform source for the measurement. The command `REF<x>:SELECT`, described on page 2–212, sets the selected channel.

`REF<z>CH<x>` specifies that channel <x> in reference group <z> is the waveform source for the measurement.

***NOTE**. If a reference waveform is specified as a source, the waveform must have a display type of ANALOGALL, ANALOGSEL, BUS, or TIMING.*

Examples `MEASUREMENT:MEAS2:SOURCE1 CH1`
specifies channel 1 as the measurement 2 source.

`MEASUREMENT:MEAS2:SOURCE1?`
might return `REF1CH3`, indicating channel 3 of reference #1 is the current source waveform for this measurement.

MEASUrement:MEAS<x>:SOURCE2

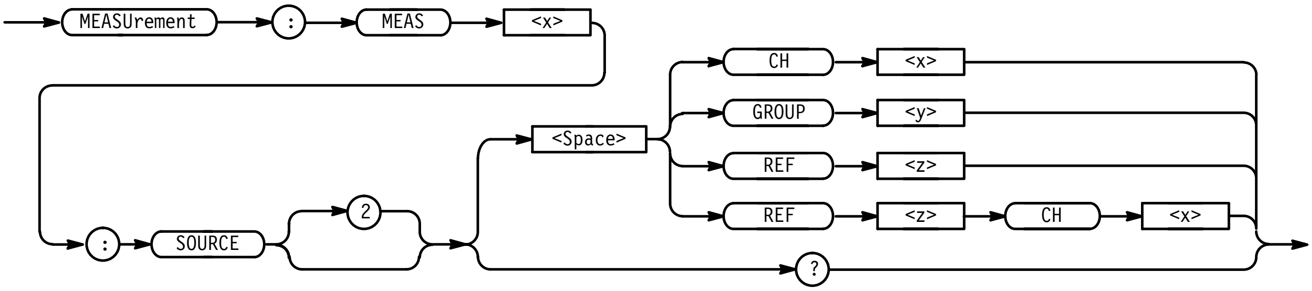
Sets or queries the source to measure "to" when taking a delay or phase measurement for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4) where <x> specifies 1 through 4. Using this command is equivalent to setting the source in the Delay from Selected Wfm side menu or the Phase from Selected Wfm side menu.

Group Measurement

Syntax

```
MEASUrement:MEAS<x>:SOURCE2 { CH<x> | GROUP<y> | REF<z> |
REF<z>CH<x> }
```

```
MEASUrement:MEAS<x>:SOURCE2?
```



Arguments `CH<x>` specifies that channel <x> is waveform source for the measurement.

GROUP<y> specifies that the channel currently selected in group <y> is the waveform source for the measurement. The command GROUP<x>:SELECT, described on page 2–133, sets the selected channel.

REF<z> specifies that the currently selected channel in reference group <z> is the waveform source for the measurement. The command REF<x>:SELECT, described on page 2–212, sets the selected channel.

REF<z>CH<x> specifies that channel <x> in reference group <z> is the waveform source for the measurement.

Examples

```
MEASUREMENT:MEAS4:SOURCE2 CH1
```

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

```
MEASUREMENT:MEAS2:SOURCE2?
```

might return REF1CH3, indicating channel 3 of reference #1 is the current source waveform for this measurement.

MEASUrement:MEAS<x>:STATE

Sets or queries the state (on or off) of MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4) where <x> specifies 1 through 4. The source for the measurement can be selected using the MEASUrement:MEAS<x>:SOURCE1 command.

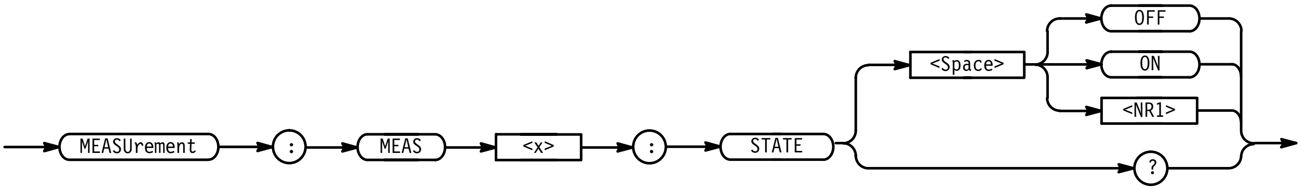
Group

Measurement

Syntax

```
MEASUrement:MEAS<x>:STATE { OFF | ON | <NR1> }

MEASUrement:MEAS<x>:STATE?
```



Arguments

OFF or <NR1> = 0 turns the selected measurement off.

ON or <NR1> $\neq$ 0 turns measurements on.

Examples

```
MEASUREMENT:MEAS1:STATE ON
```

turns measurement defined as MEAS1 on.

```
MEASUREMENT:MEAS4:STATE?
```

returns either 0 or 1, indicating the state of MEAS4.

MEASUrement:MEAS<x>:TYPe

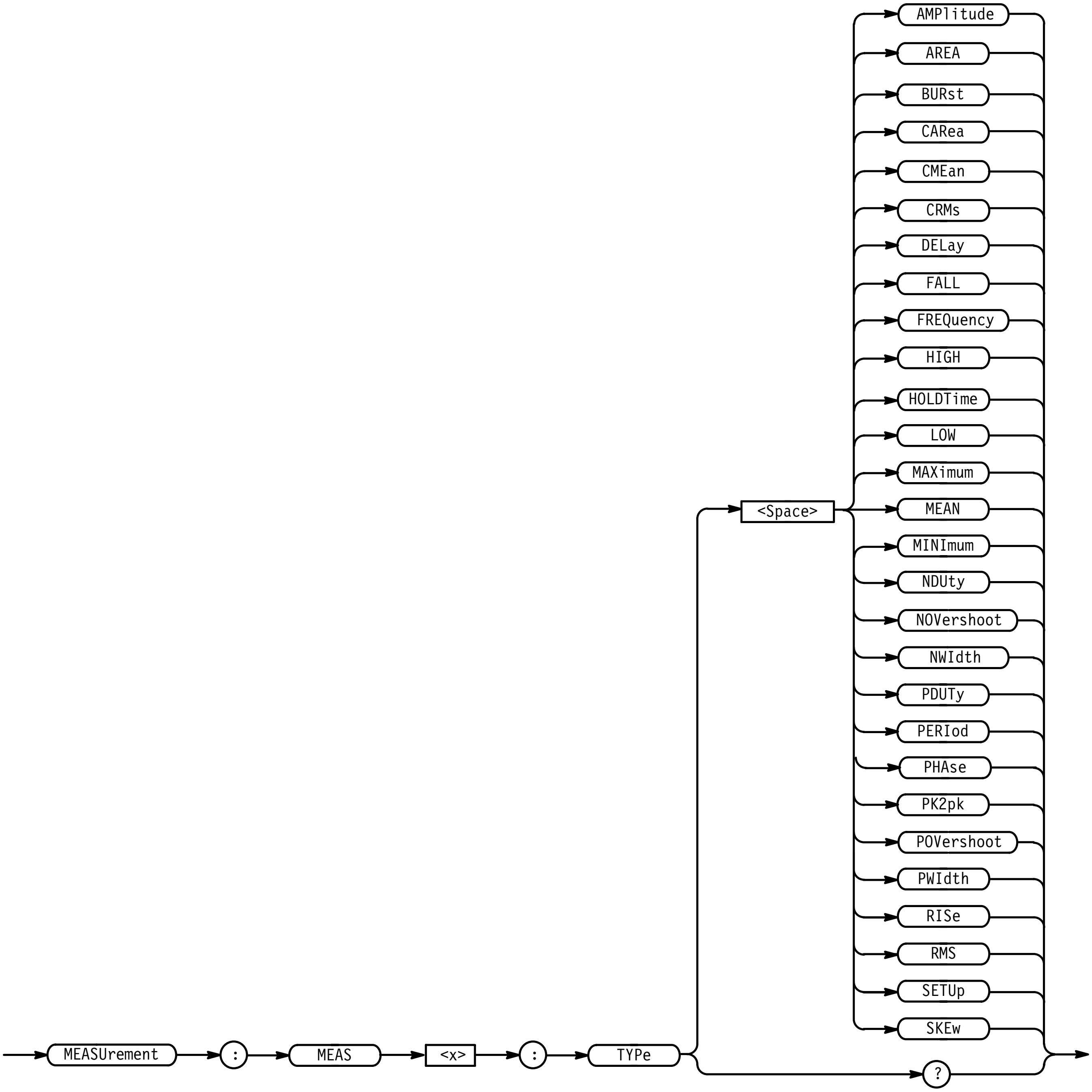
Sets or queries the measurement type for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4) where <x> specifies 1 through 4. Using this command is equivalent to selecting the measurement in the Select Measurement side menu.

Group Measurement

Syntax

```
MEASUrement:MEAS<x>:TYPe { AMPlitude | AREA | BURst | CARea |
CMEan | CRMs | DELay | FALL | FREQuency | HIGH | HOLDTime | LOW |
MAXimum | MEAN | MINImum | NDUty | NOVershoot | NWIdth | PDUTy |
PERIod | PHAse | PK2pk | POVershoot | PWIdth | RISe | RMS | SETUp
| SKEw }
```

```
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.

`HOLDTime` is time between the first transition to an illegal state for all waveforms in a group and the clocking edge of the waveform in the channel specified as the clock. The clock can be any channel. It does not have to be included in the measured group. Hold measurements define “illegal state” as any level between the low reference the high reference levels. These levels can be queried or set using the MEASUREMENT:REFLEVEL commands that begin on page 2–189.

`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.

SETUp is the time between the last transition to a legal state for all waveforms in a group and the clocking edge of the waveform specified as the clock. The clock can be any channel. It does not have to be included in the measured group. Setup measurements define "legal state" as a level either below the low reference level or above the high reference level. These levels can be queried or set using the MEASUREMENT:REFLEVEL commands that begin on page 2–189.

SKEw is the time between the first transition to an illegal state for all waveforms in a group and last transition to a legal state for the same waveforms. Skew measurements define "legal state" as a level either below the low reference level or above the high reference level. They define an "illegal state" as a level between the the low and high reference levels. These levels can be queried or set using the MEASUREMENT:REFLEVEL commands that begin on page 2–189.

Examples

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

specifies MEAS3 to calculate the Root Mean Square voltage.

```
MEASUREMENT:MEAS4:TYPE?
```

might return HOLDTIME.

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

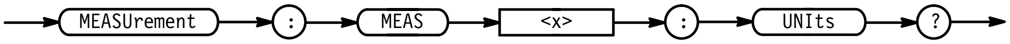
Returns the measurement units for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4) where <x> specifies 1 through 4. The type of measurement selected (see the MEASUrement:MEAS<x>:TYPE command) determines the unit type returned.

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.

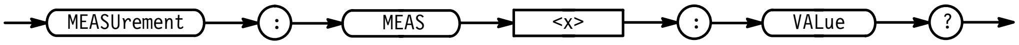
MEASUrement:MEAS<x>:VALue? (Query Only)

Returns the measurement value calculated for MEAS<x>. MEAS<x> is one of the four measurements available (named Measurement 1 through Measurement 4), where <x> specifies 1 through 4.

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

Group Measurement

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



Returns `<NR3>`, the current value of the measurement.

Examples `MEASUREMENT:MEAS<x>:VALUE?`
might return `1.20E+0`, indicating the measurement value is 1.20 volts.

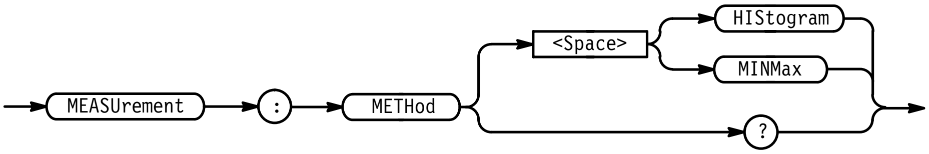
MEASUrement:METHod

Sets or queries the method used to calculate the 0 and 100% reference level. This command 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.

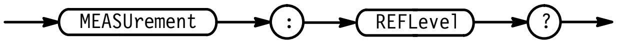
`MEASUREMENT:METHOD?`
returns `MINMAX` when the reference levels are set to MIN and MAX.

MEASUrement:REFLevel? (Query Only)

Returns all measurement system reference level settings.

Group Measurement

Syntax `MEASUrement:REFLevel?`



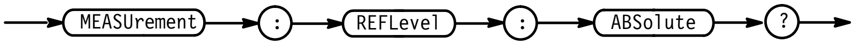
Examples `MEASUREMENT:REFLEVEL?`
might return a string similar to: `:MEASUREMENT:REFLEVEL:METHOD PERCENT;ABSOLUTE:HIGH 0.0E+0;LOW 0.0E+0;MID 0.0E+0;MID2 0.0E+0;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0E+0;LOW 10.0E+0;MID 50.0E+0;MID2 50.0E+0`.

MEASUrement:REFLevel:ABSolute? (Query Only)

Returns the absolute reference level settings.

Group Trigger

Syntax `MEASUrement:REFLevel:ABSolute?`



Examples `MEASUREMENT:REFLEVEL:ABSOLUTE?`
might return a string similar to: `:MEASUREMENT:REFLEVEL:ABSOLUTE:HIGH 0.0E+0;LOW 0.0E+0;MID 0.0E+0;MID2 0.0E+0`.

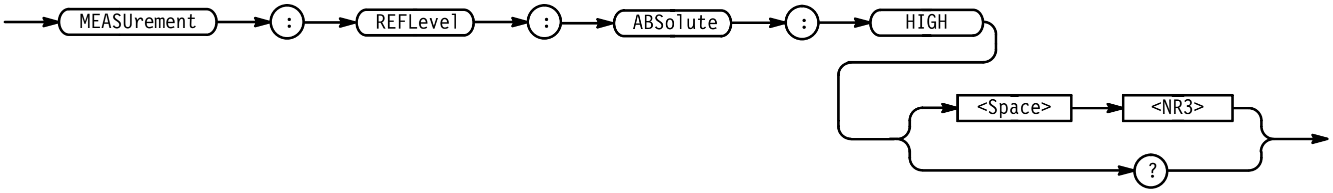
MEASUrement:REFLevel:ABSolute:HIGH

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 1.71E+0`
sets the high reference level value to 1.71 V.

`MEASUREMENT:REFLEVEL:ABSOLUTE:HIGH?`
might return `4.0E+0`, indicating the high reference level value is 4.0 volts.

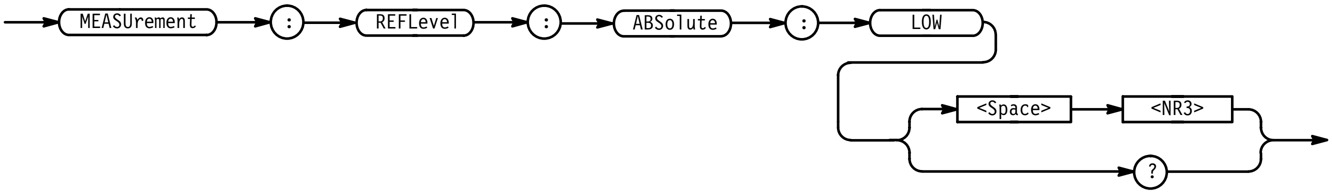
MEASUrement:REFLevel:ABSolute:LOW

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 1.0E+0
sets the low reference level value to 1.0 volts

MEASUREMENT:REFLEVEL:ABSOLUTE:LOW?
might return 0.0E+0, indicating the low reference level value is 0.0 volts.

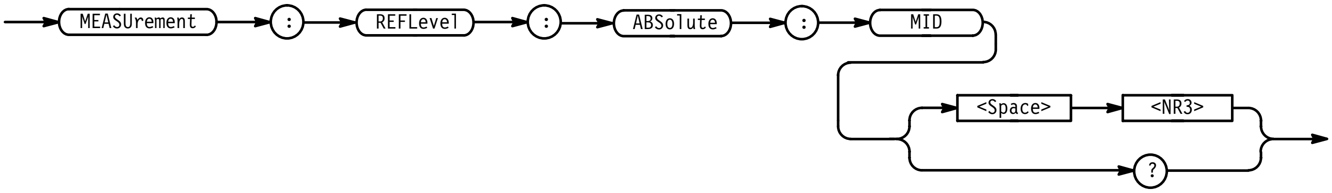
MEASUrement:REFLevel:ABSolute:MID

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 7.1E–1
sets the mid reference level value to 0.71 volts.

```
MEASUREMENT:REFLEVEL:ABSOLUTE:MID?
```
might return `3.0E+0`, indicating the mid reference level value is 3.0 volts.

MEASUrement:REFLevel:ABSolute:MID2

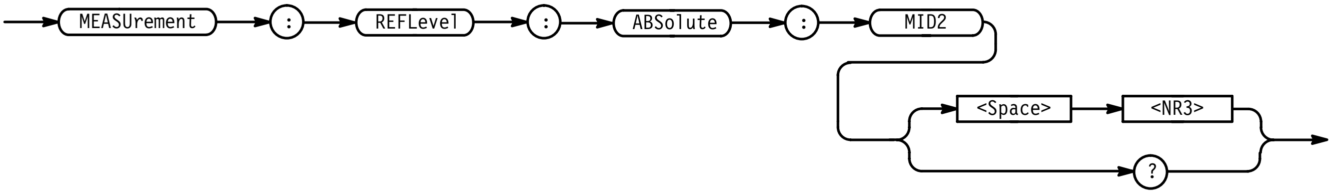
Sets or queries the Mid reference level for the "to" waveform when taking a Delay measurement. Comprises the 50% reference level when MEASUrement:REFLevel:METHod is set to ABSolute. This command is equivalent to setting the Reference Levels in the Measure menu.

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 0.5E+0
```
sets the Mid2 reference level value for the delay waveform to 0.5 volts.

```
MEASUREMENT:REFLEVEL:ABSOLUTE:MID2?
```
might return `3.2E+0`, indicating the mid2 reference level value is 3.2 volts.

MEASUrement:REFLevel:METHod

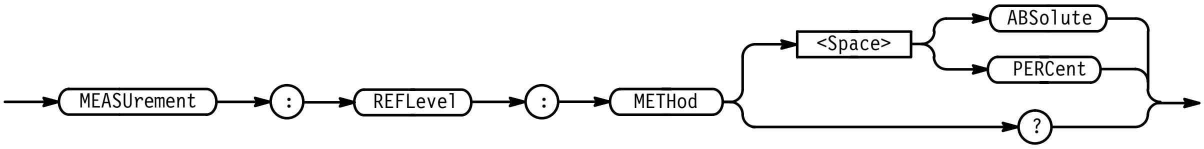
Specifies which reference levels are used for Measurement calculations. This command is equivalent to setting the levels in the Reference Levels side menu.

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.

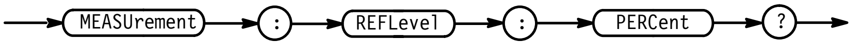
`MEASUREMENT:REFLEVEL:METHOD?`
returns either `ABSolute` or `PERCENT`, indicating the reference levels currently in use.

MEASUrement:REFLevel:PERCent? (Query Only)

Returns the percent settings reference level.

Group Measurement

Syntax `MEASUrement:REFLevel:PERCent?`



Examples `MEASUREMENT:REFLEVEL:PERCENT?`
might return a string similar to: `:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0E+0;LOW 10.0E+0;MID 50.0E+0;MID2 50.0E+0`.

MEASUrement:REFLevel:PERCent:HIGH

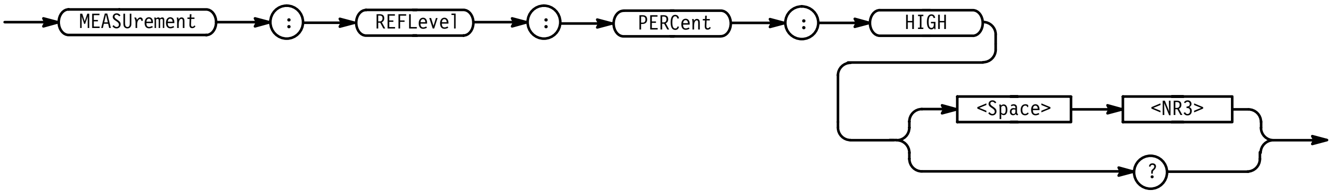
Sets or queries the High reference level (as a percentage of HIGH). Use this command to calculate the high reference level when MEASUrement:REFLevel:METHod is set to PERCent. This command is equivalent to setting the Reference Levels in the Measure menu.

Group Measurement

Syntax

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

```
MEASUrement:REFLevel:PERCent:HIGH?
```



Arguments `<NR3>` ranges from 0 to 100%, and is the high reference level. The default is 90%.

Examples

```
MEASUREMENT:REFLEVEL:PERCENT:HIGH 95
```

sets the high reference level value to 95% of HIGH.

```
MEASUREMENT:REFLEVEL:PERCENT:HIGH?
```

might return `70.0E+0`, indicating the high reference level value is set to 70% of HIGH.

MEASUrement:REFLevel:PERCent:LOW

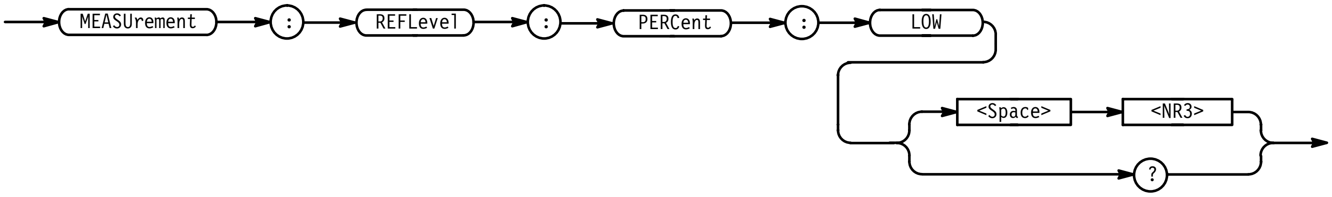
Sets or queries the Low reference level (as a percentage of HIGH). Use this command to calculate the low reference level when MEASUrement:REFLevel:METHod is set to PERCent. This command is equivalent to setting the Reference Levels in the Measure menu.

Group Measurement

Syntax

```
MEASUrement:REFLevel:PERCent:LOW <NR3>
```

```
MEASUrement:REFLevel:PERCent:LOW?
```



Arguments <NR3> ranges from 0 to 100%, and is the low reference level. The default is 10%.

Examples `MEASUREMENT:REFLEVEL:PERCENT:LOW 30`
sets the low reference level value to 30% of HIGH.

`MEASUREMENT:REFLEVEL:PERCENT:LOW?`
might return `15.0E+0`, indicating the low reference level value is set to 15% of HIGH.

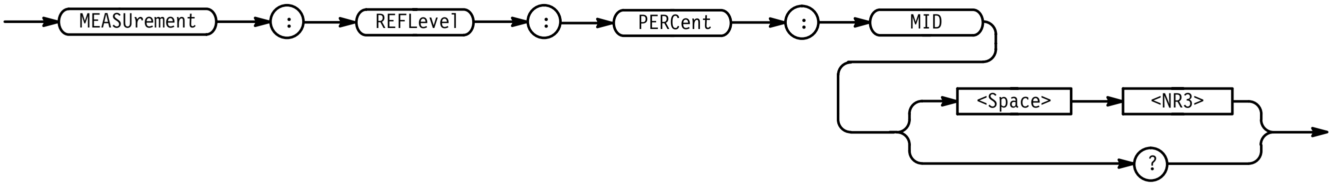
MEASUrement:REFLevel:PERCent:MID

Sets or queries the Mid reference level (as a percentage of HIGH). Use this command to calculate the mid reference level when MEASUrement:REFLevel:METHod is set to PERCent. This command is equivalent to setting the Reference Levels in the Measure menu.

Group Measurement

Syntax `MEASUrement:REFLevel:PERCent:MID <NR3>`

`MEASUrement:REFLevel:PERCent:MID?`



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

Examples `MEASUREMENT:REFLEVEL:PERCENT:MID 60`
sets the mid reference level value to 60% of HIGH.

MEASUREMENT:REFLEVEL:PERCENT:MID?
might return 80.0E+0 indicating the mid reference level value is set to 80% of HIGH.

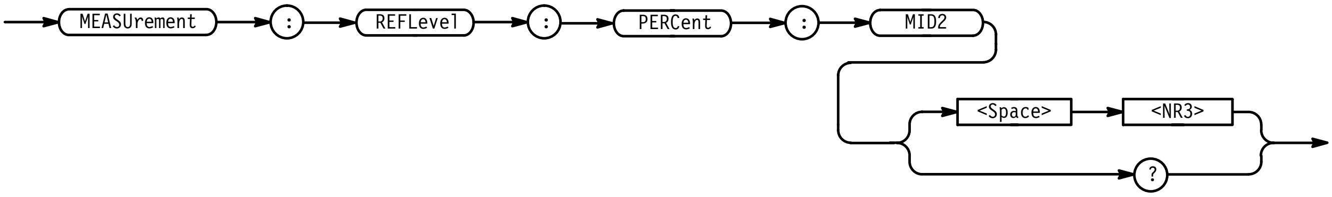
MEASUrement:REFLevel:PERCent:MID2

Sets or queries the Mid reference level (as a percentage of HIGH). 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%, and is the mid reference level. The default is 50%.

Examples MEASUREMENT:REFLEVEL:PERCENT:MID2 40
sets the mid2 reference level value (for the second waveform specified) to 40% of HIGH.

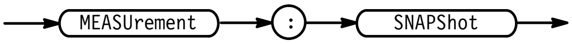
MEASUREMENT:REFLEVEL:PERCENT:MID2?
might return 80.0E+0, indicating the mid2 reference level value for the second waveform specified is set to 80% of HIGH.

MEASUrement:SNAPShot (No Query Form)

Displays the measurement snapshot.

Group Measurement

Syntax MEASUrement:SNAPShot



Examples `MEASUREMENT:SNAPSHOT`
invokes the measurement snapshot for the selected channel of the selected group.

MESSage

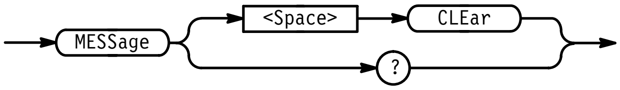
Clears the message window. The MESSage? query returns the current message parameters.

Group Display

Syntax

```
MESSage CLEar
```

```
MESSage?
```



Arguments `CLEar` removes the message from the message window. Using this argument is equivalent to sending `MESSage SHOW ""`.

Examples `MESSAGE CLEAR`
clears the message from the window.

`MESSAGE?`
might return a string similar to: `:MESSAGE:SHOW "";STATE 0;BOX 74,84,475,135`.

MESSage:BOX

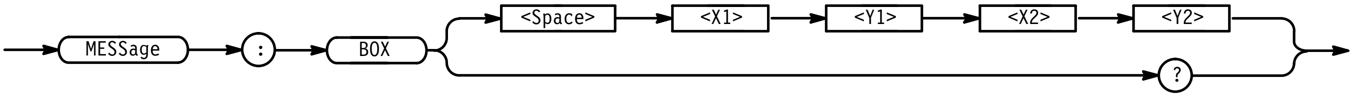
Defines the size and position of the message window. This command does not display the window unless MESSage:STATE is ON.

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–2. Shorter windows clip characters.

Figure 2–5 shows the coordinate system relative to the screen.

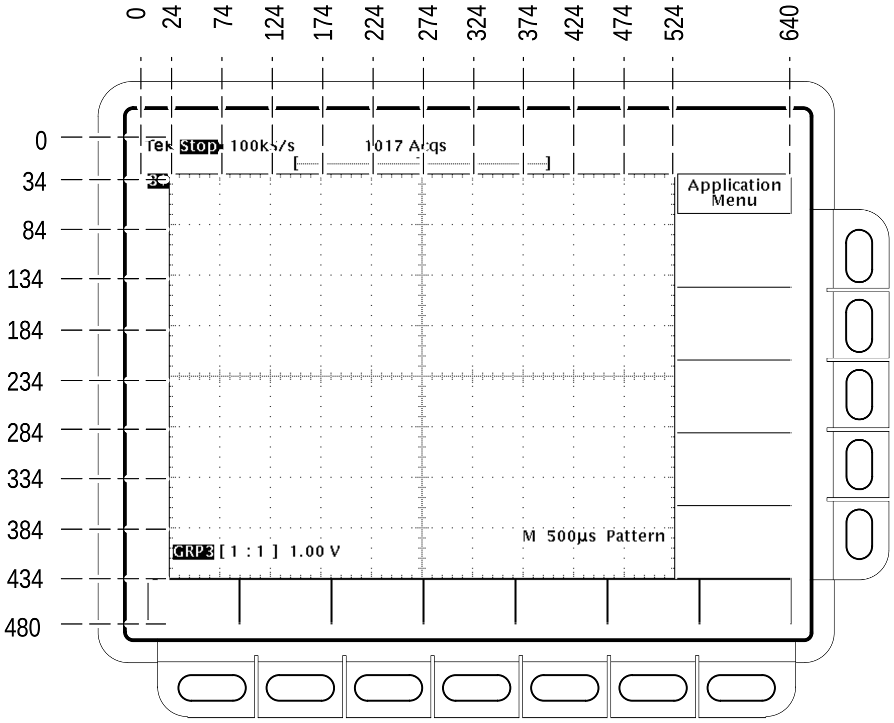


Figure 2–5: Message Window Coordinates

MESSage:SHOW

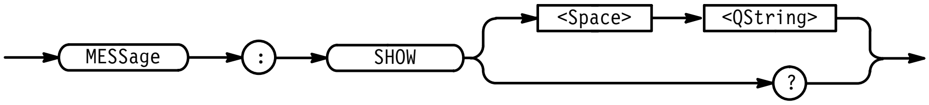
Clears the contents of the message window and displays the new message.

Group Display

Syntax

```
MESSage:SHOW <QString>
```

```
MESSage:SHOW?
```



Arguments `<QString>` is the message and can include any of the characters shown in the Logic Scope Character Chart in Appendix A. The maximum length of the message is 1000 characters.

The message is left-justified and 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 escape codes other than those 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–2.

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

Turns the display message window on and off.

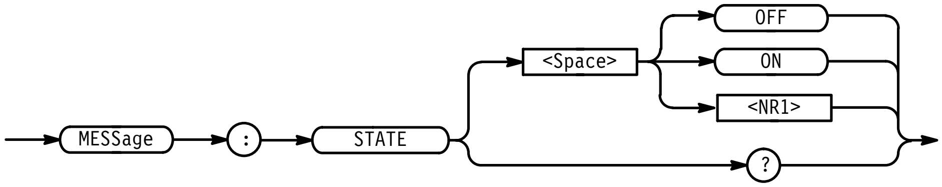
Group

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

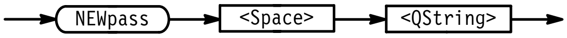
Miscellaneous

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 3–1.

Group

Status and Error

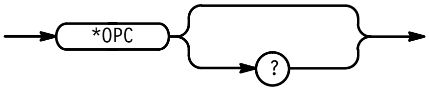
Related Commands

BUSY?, *WAI

Syntax

```
*OPC

*OPC?
```



The *OPC command allows you to synchronize the operation of the logic scope with your application program. Synchronization methods are described starting on page 3–7.

Table 2–24: Commands that Generate an Operation Complete Message

Operation	Command
Automatic scope adjustment	AUTOSet EXECute
Internal self-calibration	*CAL

Table 2–24: Commands that Generate an Operation Complete Message (Cont.)

Operation	Command
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 logic scope 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 the password is case sensitive.

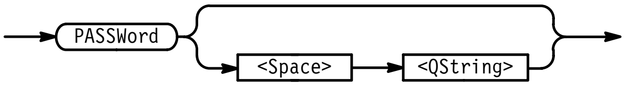
Group Miscellaneous

Related Commands NEWpass, *PUD

Syntax

```
PASSWord
```

```
PASSWord <QString>
```



Arguments `<QString>` is the password and can include up to 10 characters. The factory default password is “XYZZY” and is always valid.

Examples

```
PASSWORD "XYZZY"
```

Enables the *PUD and NEWpass set commands.

```
PASSWORD
```

Disables the *PUD and NEWpass set commands. You can still use the query version of *PUD.

*PSC

Sets and queries the power on status flag that controls the automatic handling of the DESER, SRER, and ESER registers. When *PSC is true, the DESER register is set to 255 and the SRER and ESER registers are set to 0 at power on. When *PSC is false, the current values in the DESER, SRER, and ESER registers are preserved in nonvolatile memory when power is shut off and are restored at power on. For a complete discussion of the use of these registers, see page 3–1.

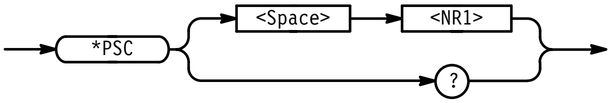
Group Status and Error

Related Commands DESE, *ESE, FACtory, *RST, *SRE

Syntax

```
*PSC <NR1>
```

```
*PSC?
```



Arguments <NR1> = 0 sets the power on status clear flag to false and disables the power on status clear, allowing the logic scope 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

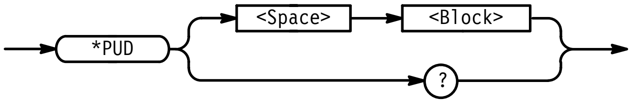
Sets or queries a string of user data protected by the PASSWord command. This data can only be modified by first entering the correct password. The password is not necessary to query the data.

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`.

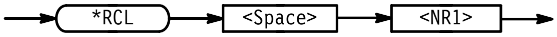
*RCL (No Query Form)

Restores the state of the logic scope 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 of 1 to 10, and specifies a setup storage location. Using an out-of-range value invokes execution error 222: “Data out of range”.

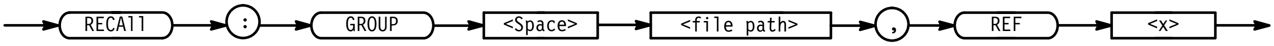
Examples `*RCL 3`
restores the logic scope from a copy of the settings stored in memory location 3.

RECAll:GROUP (No Query Form)

Recalls a stored group into a reference group location. This command is equivalent to selecting Recall File in the Save/Recall Group menu.

Group Save and Recall

Syntax `RECAll:GROUP <file path>,REF<x>`



Arguments `REF<x>` is the internal reference memory location where the waveform group is recalled to.

`<file path>` is a quoted string that defines the file name and path. Input the file path using the form `<drive>/<dir>/<filename>`. `<drive>` and one or more `<dir>`'s are optional. If you do not specify them, the logic scope will recall the waveform from the default directory. `<filename>` stands for a filename of up to 8 characters followed by a period (".") and any 3-character extension. Do not use wild card characters.

Examples `RECALL:GROUP "TEK00000.GRP",REF1`

recalls the group stored in the file named TEK00000.GRP to reference location 1.

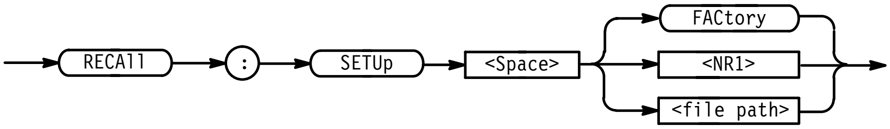
RECAll:SETUp (No Query Form)

Restores a saved or factory front-panel setup to the logic scope. This command is equivalent to selecting Recall Saved Setup or Recall Factory Setup or Recall Saved 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> | <file path> }`



Arguments FACtory selects the factory setup.

<NR1> is a value in the range of 1 to 10 and specifies a setup storage location. Using an out-of-range value invokes execution error 222: “Data out of range”.

<file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>/<dir>/<filename>. <drive> and one or more <dir>’s are optional. If you do not specify them, the logic scope will write the file to the default directory. <filename> stands for a filename of up to 8 characters followed by a period (“.”) and any 3-character extension. Do not use wild card characters.

Examples RECALL:SETUP FACTORY
recalls (and makes current) the front panel setup to its factory defaults.

RECALL:SETUP 1
recalls the front panel setup from setup 1.

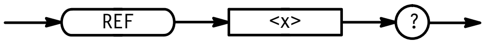
RECALL:SETUP "TEK00000.SET"
recalls the front panel setup from the file TEK00000.SET in the default directory on the default drive.

REF<x>? (Query Only)

Returns all group parameters for the specified reference group.

Group Vertical

Syntax REF<x>?



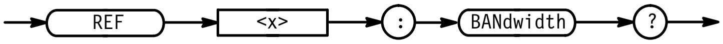
Examples REF1?
might return a string similar to: :REF1:FIRST 5;LAST 14;SELECT 5; DISPLAY OFF;SCALE 1.000E+0;POSITION 0.0E+0;OFFSET 0.0E+0; BANDWIDTH FULL;VHIGH 2.00E+0;VLOW 800.0E–3;HEIGHT 1.0E+0 for reference group 1.

REF<x>:BANdwidth? (Query Only)

Returns the bandwidth of the specified reference group.

Group Vertical

Syntax `REF<x>:BANdwidth?`



Returns `FULL` or `TWENTY` the acquisition bandwidth of the specified reference group.

Examples `REF2:BANDWIDTH?`

might return `FULL`, indicating there is no bandwidth limiting on reference group 1.

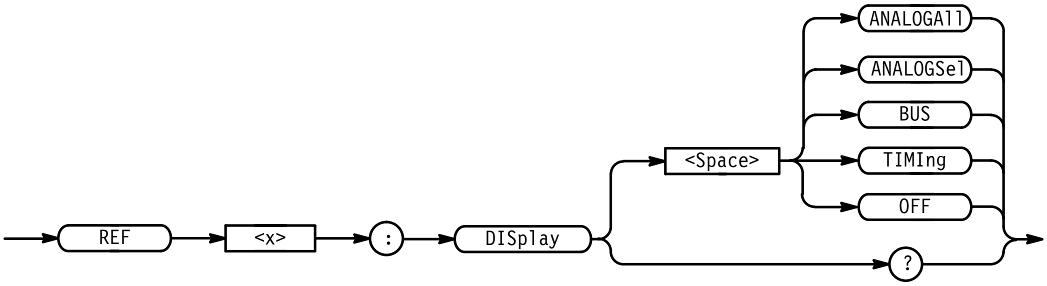
REF<x>:DISplay

Sets or returns the display mode for the specified reference group.

Group Vertical

Syntax `REF<x>:DISplay { ANALOGAll | ANALOGSel | BUS | TIMIng | OFF }`

`REF<x>:DISPLAY?`



Arguments `ANALOGAll` displays each waveform in the reference group as the analog of its input signal. Use Analog mode to measure the actual response of a group of waveforms.

ANALOGSel displays only the selected waveform in a reference group as the analog input signal. Use Analog Select mode to measure the actual response of the selected waveform in a reference group.

BUS displays each waveform in the reference group as one or more of three digital logic levels defined as HIGH, LOW, or MID. The levels of each waveform displayed depends on how the instantaneous level of its input signal compares with two threshold levels you define. Use BusForm mode to measure the idealized response of waveforms with respect to defined thresholds.

TIMIng displays each waveform in the reference group as one or more of three digital logic levels, as is done for BusForm. However, unlike BusForm, the logic scope offsets the ground reference of each waveform so they skew vertically on screen. Use Timing mode to measure timing relationships between the waveforms included in the group.

OFF turns off the display of the selected reference group.

Examples

REF3:DISPLAY BUS
sets reference group 3 to BusForm display mode.

REF1:DISPLAY?
might return ANALOG display mode.

REF<x>:FIRst? (Query Only)

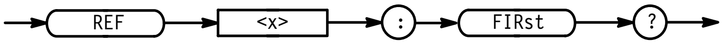
Returns the first channel in the specified reference group. Every group includes the first channel (lowest numbered in the group), the last channel (highest numbered), and every channel in between.

Group

Vertical

Syntax

```
REF<x>:FIRst?
```



Returns

NR1 the first channel in the group from 1 to 16.

Examples

REF4:FIRST?
might return 12, indicating the first channel of reference group 8 is channel 12.

REF<x>:HEIght

Sets or returns the display height setting (in divisions) for a specified reference group when set to BusForm or Timing Diagram modes. BusForm and Timing Diagram modes display waveform levels as logical HIGHS or LOWS. Setting the height sets the number of vertical division between the HIGH and LOW levels.

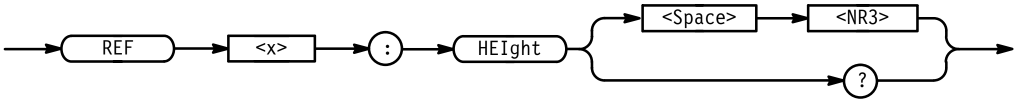
Group Vertical

Related Commands GROUP<x>:HEIght

Syntax

```
REF<x>:HEIght NR3
```

```
REF<x>:HEIght?
```



Arguments <NR3> is the number of display divisions between the HIGH and LOW levels from 200.0E–3 to 8.0E+0.

Examples

`REF2:HEIGHT 5`
displays the HIGH and LOW levels for waveforms displayed in group 2 five vertical divisions apart.

`REF7:HEIGHT?`
might return 2.0E+0 indicating the HIGH and LOW levels for waveforms displayed in group 7 are spaced two vertical divisions apart.

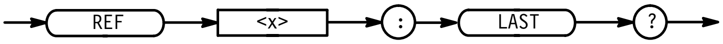
REF<x>:LAST? (Query Only)

Returns the last channel in the specified reference group. Every reference group includes the first channel (lowest numbered in the group), the last channel (highest numbered), and every channel in between.

Group Vertical

Syntax

```
REF<x>:LAST?
```



Returns <NR1> the last channel in the group from 1 to 16.

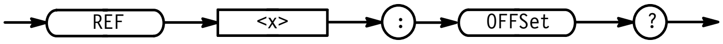
Examples `REF4:LAST?`
might return 16, indicating that the last channel of group 4 is channel 16.

REF<x>:OFFSet? (Query Only)

Returns the specified reference group offset in volts.

Group Vertical

Syntax `REF<x>:OFFSet?`



Returns <NR3> the offset in effect when the group was acquired.

Examples `REF1:OFFSET?`
might return 500.0E–3, indicating the current reference group 1 offset is 0.5 volts.

REF<x>:POSition

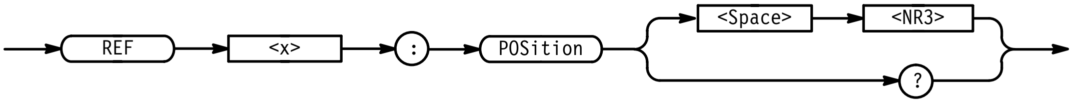
Sets or returns the vertical position of the specified reference group.

Group Vertical

Syntax

```
REF<x>:POSition <NR3>

REF<x>:POSition?
```



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

Examples `REF2:POSITION 1.3E+0`

positions the reference group 2 input signal 1.3 divisions above the center of the display.

`REF1:POSITION?`

might return `–1.3E+0`, indicating that the current position of reference group 1 is at –1.3 divisions.

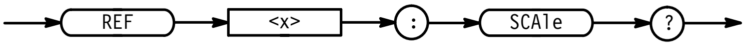
REF<x>:SCAle (Query Only)

Returns the volts per division scale of the specified reference group. The setting for group scale determines the vertical height of any waveform group displayed in Analog Display mode. It also scales a waveform group, before processing for display, in BusForm and Timing Diagram display modes.

Group Vertical

Related Commands GROUP<x>:SCAle , GROUP<x>:HEIght

Syntax `REF<x>:SCAle?`



Examples `REF2:SCALE?`

might return `1.00E+0`, indicating that the current V/div setting of group 2 is 1 V/div.

REF<x>:SELect

Sets or returns the selected channel within the reference group. This command is similar to setting Selected Channel in the Group Definition menu. The logic scope highlights the selected channel in the group. When automated measurements are implemented, the logic scope makes the measurement on the selected channel within the selected group. The selected channel must fall between the first channel and last channel, inclusively.

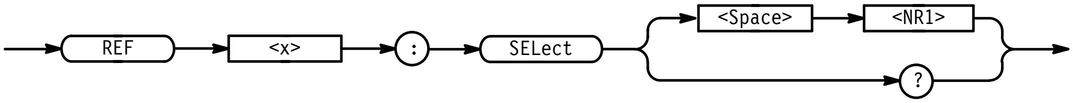
Group Vertical

Related Commands GROUP<x>:SELect

Syntax

```
REF<x>:SELect <NR1>
```

```
REF<x>:SELect?
```



Arguments <NR1> specifies the selected channel of a reference group from 1 to 16, and must be one of the channels specified in the group.

Examples

```
REF2:SELECT 5
```

specifies channel 5 as the selected channel for reference group 2.

```
REF1:SELECT?
```

might return channel 3 as the selected channel for reference group 1.

REF<x>:VHIgh

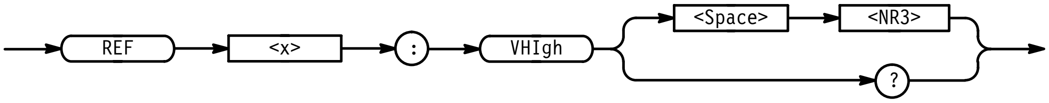
Sets or returns the upper threshold voltage (Vhigh min) for the specified reference group. Input signals in the group must cross this threshold going positive to be displayed at a HIGH digital level. This command affects groups set to BusForm or Timing Diagram display mode only.

Group Vertical

Syntax

```
REF<x>:VHIgh <NR3>
```

```
REF<x>:VHIgh?
```



Arguments <NR3> is the high threshold value from –11.99E+0 to 15.0E+0 volts

Examples REF1:VHIGH 3.3E+0
sets the upper threshold voltage of reference group 1 to +3.3 volts.

REF2:VHIGH?
might return 3.3E+0, indicating the upper threshold voltage is set to +3.3 volts for reference group 2.

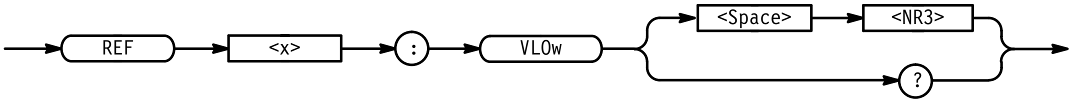
REF<x>:VLOw

Sets or returns the lower threshold voltage (Vlow max) for the specified reference group. Input signals in the group must cross this threshold going negative to be displayed at a LOW digital level. This command affects groups set to BusForm or Timing Diagram display mode only.

Group Vertical

Syntax REF<x>:VLOw <NR3>

REF<x>:VLOw?



Arguments <NR3> is the low threshold value from –12.0E+0 to 14.99E+0.

Examples REF1:VLOW 5.0–01 sets the lower threshold voltage of reference group 1 to +0.5 volts.

REF2:VLOW?
might return 5.0E–01 indicating the lower threshold voltage is set to +0.5 volts for reference group 2.

REF<x>:VOLts? (Query Only)

Returns the vertical gain of the specified reference group. Using this command is equivalent to adjusting the front panel Vertical SCALE knob.

Group

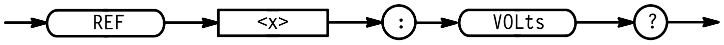
Vertical

Related Commands

GROUP<x>:SCAle, GROUP<x>:VOLts

Syntax

```
REF<x>:VOLts?
```



Returns

<NR3> is the current vertical gain from 50 mV to 2 V/division.

Examples

```
REF2:VOLTS?
```

might return `1.00E+0`, indicating the current V/div setting of reference group 2 is 1 V/div.

REM (No Query Form)

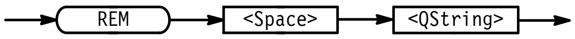
Specifies a comment. This line is ignored by the instrument.

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)

Resets the logic scope to a known state; 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 returns the instrument settings to the factory defaults (see Appendix D).

The *RST command does not alter the following:

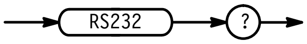
- the state of the IEEE Std 488.1-1987 interface
- the selected IEEE Std 488.1-1987 address of the logic scope
- calibration data that affect device specifications
- the Output Queue
- the Service Request Enable Register setting
- the Standard Event Status Enable Register setting
- the power on status clear flag setting
- alias definitions
- stored settings
- the *PUD? response

RS232? (Query Only)

Queries the RS-232 settings.

Group RS232

Syntax `RS232?`



Examples `RS232?`
might return `RS232 BAUD: 9600, SOFTFLAGGING: OFF, HARDFLAGGING: ON, PARITY: NONE, STOPBITS: 1.`

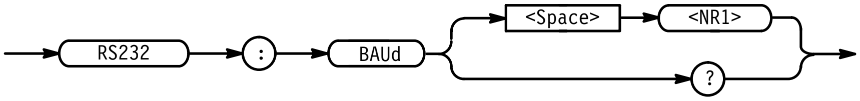
RS232:BAUd

Sets or queries RS-232-C interface transmission speed.

Group 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:BAUD?`
might return `4800`, indicating the current baud rate setting.

RS232:HARDFlagging

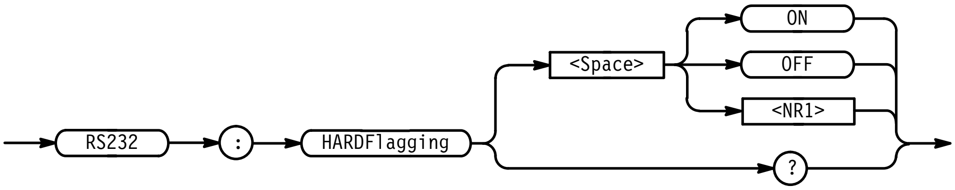
Sets or queries the input and output hard flagging over the RS-232 port. It uses the RFR (Ready For Receive) and CTS (Clear To Send) lines to control data transmission. On output, the logic scope transmits data only when CTS is asserted. When CTS is not asserted, the logic scope stops transmitting data. 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. Then CTS is asserted again to restart transmission.

Group 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:HARDFLAGGING?`
might return `0`, indicating hardflagging is off.

RS232:PARity

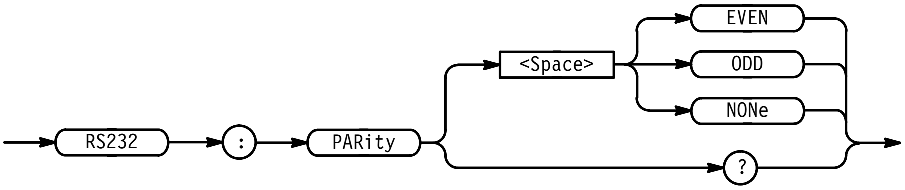
Sets or queries the parity used for all RS-232-C data transfers. Parity adds a bit to the character sequence. When parity is odd or even, the logic scope generates the selected parity on output and checks all input against the selected parity. When parity is none, the logic scope performs no input parity error checks and generates no output parity.

Group RS232

Syntax

```
RS232:PARity { EVEN | ODD | NONe }
```

```
RS232:PARity?
```



Arguments

`EVEN` indicates the parity bit is sent with even parity and bytes received are expected to have even parity.

`ODD` indicates the parity bit is sent with odd parity and bytes received are expected to have odd parity.

`NONe` indicates that no parity bit is sent and none are expected.

Examples

`RS232:PARITY EVEN`
sets the parity to even.

`RS232:PARITY?`
might return `NONE`, indicating parity is turned off.

RS232:SOFTFlagging

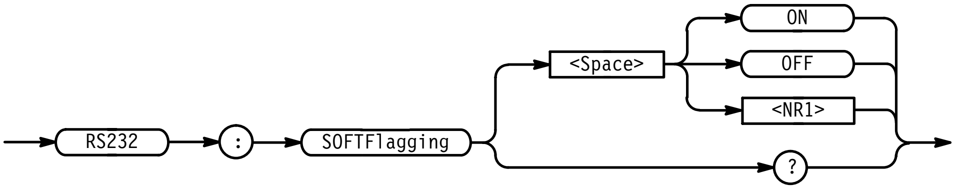
Sets or queries the input and output soft flagging over the RS-232 port. It stops transmitting data any time it receives an XOFF (DC3) character. It sends an XOFF character when its 512 byte input buffer has 80 free bytes. The logic scope begins transmitting data again when it receives an XON (DC1) characters. It sends XON when its input buffer is has 100 free bytes.

Group 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

Sets or queries the number of transmission stop bits sent with each character to identify the end of data for that character.

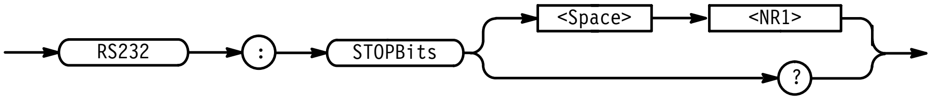
Group

RS232

Syntax

```
RS232:STOPBits <NR1>
```

```
RS232:STOPBits?
```



Arguments

<NR1> is 1 or 2.

Examples

RS232:STOPBITS 1
sets the number of stop bits to 1.

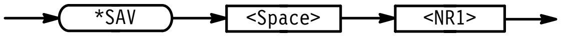
*SAV (No Query Form)

Saves the state of the logic scope to a specified memory location. You can later use the *RCL command to restore the logic scope to this saved state. Using this command 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.

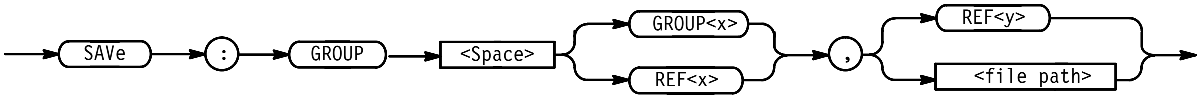
SAVe:GROUP (No Query Form)

Stores a group or reference group in one of four reference memory locations or a mass storage file. This command is equivalent to selecting either the Save GROUP or the Save to File Waveform item in the Save/Recall Group menu.

Group Save and Recall

Related Commands DELEte:GROUP

Syntax `SAVe:GROUP { GROUP<x> | REF<x> },{ REF<y> | <file path> }`



Arguments

GROUP<x> specifies the group to save.

REF<x> specifies the reference group to save.

REF<y> specifies the reference memory location.

<file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>/<dir>/<filename>. <drive> and one or more <dir>'s are optional. If you do not specify them, the logic scope will write the file to the default directory. <filename> stands for a filename of up to 8 characters followed by a period (".") and the 3-character extension "GRP". The logic scope will generate an error if you use any other extension for saving a waveform.

Examples

SAVE:WAVEFORM GROUP2,REF1
saves Group 2 waveform(s) in reference memory location 2.

SAVE:WAVEFORM GROUP1,"TEK00000.GRP"
saves Group 1 waveform(s) to the file TEK00000.GRP in the default directory and on the default drive.

SAVe:SETUp (No Query Form)

Saves the current front panel setup into the specified memory location. Using this command is equivalent to selecting the Save Current Setup in the Save/Recall Setup menu.

Group

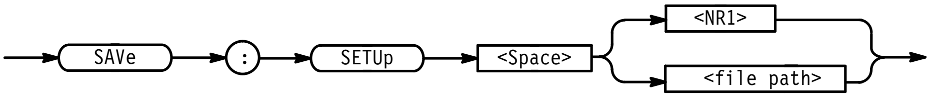
Save and Recall

Related Commands

DELEte:SETUp, RECAll:SETUp, *RCL, *SAV

Syntax

```
SAVe:SETUp { <NR1> | <file path> }
```



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.

<file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>/<dir>/<filename>. <drive> and one or more <dir>'s are optional. If you do not specify them, the logic scope will write the file to the default directory. <filename> stands for a filename of up to 8

characters followed by a period (".") and the 3-char extension "SET". The logic scope will generate an error if you use any other extension for saving a setup.

Examples `SAVE:SETUP 5`
saves the current front panel setup in memory location 5.

`SAVE:SETUP "TEK00000.SET"`
saves the current front panel setup to the file `TEK00000.SET` in the default directory and on the default drive.

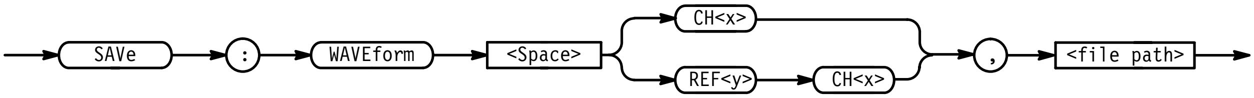
SAVe:WAVEform (No Query Form)

Saves the designated channel waveform to the file specified. A reference channel saved to a file must be displayed.

***NOTE**. Waveforms saved with this command cannot be recalled on the logic scope. This command is provided to save waveforms for viewing on Tektronix TDS oscilloscopes or processing waveforms on a personal computer.*

Group Save and Recall

Syntax `SAVe:WAVEform { CH<x> | REF<y>CH<x> },<file path>`



Arguments `CH<x>` specifies the channel to be saved from 1 to 16.

`REF<y>CH<x>` specifies the reference storage location <y> from 1 to 4 and the channel to be saved <x> from 1 to 16.

`<file path>` is a quoted string that defines the file name and path. Input the file path using the form <drive>/<dir>/<filename>. <drive> and one or more <dir>'s are optional. If you do not specify them, the `logic scope` will write the file to the default directory. <filename> stands for a filename of up to 8 characters followed by a period (".") and the 3-char extension "WFM". The `logic scope` will generate an error if you use any other extension for saving a waveform.

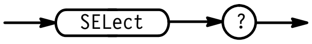
Examples `SAVE:WAVEFORM CH4,"TEK00000.WFM"` saves the channel 4 waveform to the file TEK00000.WFM in the default directory on the default drive.

SELect? (Query Only)

Returns the selected waveform and the display status of all waveforms.

Group Vertical

Syntax `SELect?`



Examples `SELECT?`

might return `GROUP1 1;GROUP2 0;GROUP3 0;GROUP4 0;GROUP5 0;GROUP6 0;GROUP7 0;GROUP8 0;REF1 0;REF2 0;REF3 0;REF4 0;CONTROL GROUP1`.

SELect:GROUP<x>

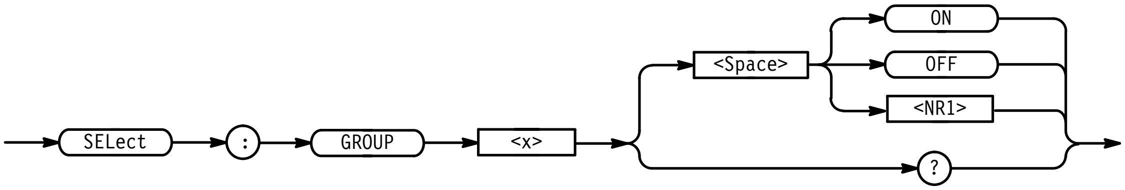
Controls or queries the display of groups. There can be up to eight groups displayed at one time, but only one group can be selected at a time. The “selected group” is the group most recently turned on. This command is equivalent to pressing the front panel GROUP or MORE button.

Group Vertical

Syntax

```
SELect:GROUP<x> { ON | OFF | <NR1> }

SELect:GROUP<x>?
```



Arguments `ON` or `<NR1>` = 1 turns on the display of the specified group.

`OFF` or `<NR1>` = 0 turns off the display of the specified group.

Examples `SELECT:GROUP5 ON`

turns on the display of group 5.

`SELECT:GROUP2?`
might return 1, indicating that group 2 is currently being displayed.

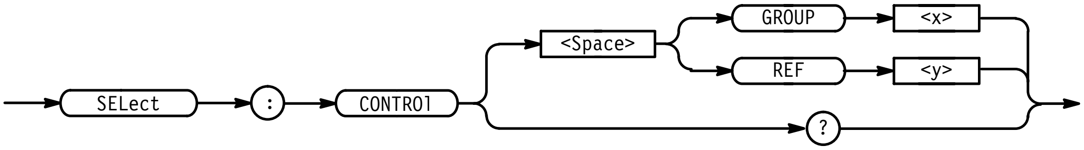
SELect:CONTROl

Sets or queries the group or reference group that is currently affected by the cursor and vertical commands.

Group Vertical

Syntax `SELect:CONTROl { GROUP<x> | REF<y> }`

`SELect:CONTROl?`



Arguments `REF<x>` specifies one of the reference groups from 1 to 4.

`GROUP<x>` specifies one of the acquisition groups from 1 to 8.

Examples `SELECT:CONTROL GROUP8`
specifies group 8 as the active group.

`SELECT:CONTROL?`
might return `REF1` as the selected reference group.

SELect:REF<x>

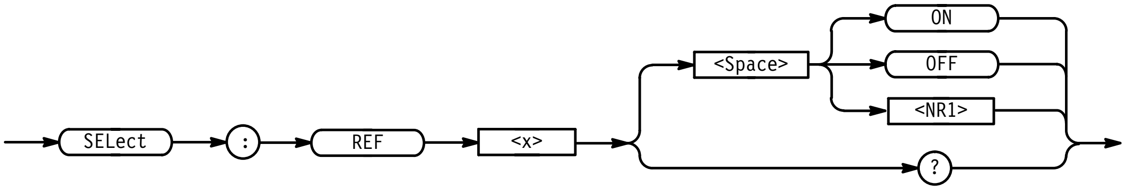
Controls or queries the display of reference groups. There can be up to four reference groups displayed at one time, but only one group can be selected at a time. The “selected group” is the group most recently turned on.

Group Vertical

Related Commands SELect:GROUP

Syntax `SELect:REF<x> { ON | OFF | <NR1> }`

SELect:REF<x>?



Arguments ON or <NR1> = 1 turns on the display of the specified group.

OFF or <NR1> = 0 turns off the display of the specified group.

Examples SELECT:REF4 ON
turns on the display of group 4.

SELECT:REF3?
might return 1, indicating that reference group 3 is currently being displayed.

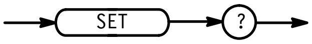
SET? (Query Only)

Returns a string listing the logic scope settings, except for configuration information for the calibration values. You can use this string to return the logic scope 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. The string may be sent back to the logic scope as a command string, if desired. 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 VECTORS;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 EPSIMAGE;PORT GPIB;LAYOUT PORTRAIT;.

*SRE

Sets or queries the bits in the Service Request Enable Register (SRER). For a complete discussion of the use of these registers, see page 3–1.

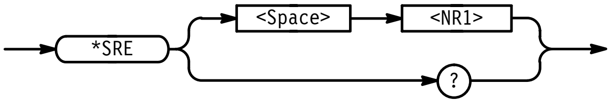
Group Status and Error

Related Commands *CLS, DESE, *ESE, *ESR?, EVENT?, EVMSg?, FACtory, *PSC, *STB?

Syntax

```
*SRE <NR1>
```

```
*SRE?
```



Arguments <NR1> is a value in the range from 0 to 255. The binary bits of the SRER are set according to this value. Using an out-of-range value causes an execution error. The power-on default for SRER is 0 if *PSC is 1. If *PSC is 0, the SRER maintains its value through a power cycle.

Examples *SRE 48

sets the bits in the SRER to 00110000 binary.

*SRE?

might return a value of 32, showing that the bits in the SRER have the binary value 00100000.

*STB? (Query Only)

Reads or queries the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. For a complete discussion of the use of these registers, see page 3–1.

Group Status and Error

Related Commands *CLS, DESE, *ESE, *ESR?, EVENT?, EVMSg?, FACtory, *SRE

Syntax `*STB?`

Returns `<NR1>`

Examples `*STB?`

might return the value `96`, showing that the SBR contains the binary value 01100000.

TEKSecure (No Query From)

Initializes both waveform and setup memories, overwriting any previously stored data.

TEKSecure writes 0's to all waveform reference memory, regardless of selected record length, and places all setups in the factory "init" state. It then verifies that waveform and setup memory are in the desired state and displays a pass or fail notifier on completion.

Group Miscellaneous

Syntax `TEKSecure`

TIMe

Sets or queries the logic scope time display.

Group

Miscellaneous

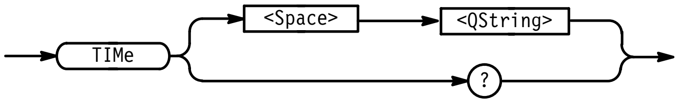
Related Commands

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"
```

sets the time to 01:24 AM.

```
TIME?
```

might return `13.33.00`, indicating the time is set to 1:33 p.m.

*TRG (No Query Form)

(Trigger) executes commands 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 defined by *DDT.

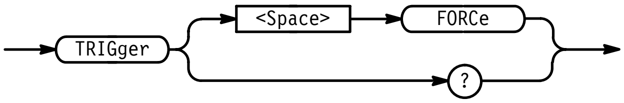
TRIGger

Forces a trigger event to occur, 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 argument will be ignored. Using this argument is equivalent to pressing the front panel FORCE TRIGGER button.

Examples `TRIGGER FORCe`

forces a trigger event to occur.

`TRIGGER?`

might return

```
:TRIGGER:THRESHOLD:CH1 0.0E+0;CH2 0.0E+0;CH3
0.0E+0;CH4 0.0E+0;CH5 0.0E+0;CH6 0.0E+0;CH7 0.0E+0;CH8
0.0E+0;CH9 0.0E+0;CH10 0.0E+0;CH11 0.0E+0;CH12 0.0E+0;CH13
0.0E+0;CH14 0.0E+0;CH15 0.0E+0;CH16 0.0E+0;:TRIGGER:MAIN:MODE
AUTO;TYPE EDGE;LEVEL 0.0E+0;HOLDOFF:VALUE 0;:TRIGGER:MAIN:
EDGE:SOURCE CH1;SLOPE RISE;:TRIGGER:MAIN:PATTERN:QUALIFIER
LIMIT;LIMIT:WHEN MORETHAN;WIDTH 2.0E-9;:TRIGGER:MAIN:PATTERN:
RANGE:WHEN INSIDE;HIGHLIMIT 2.0E-9;LOWLIMIT 2.0E-9;:TRIGGER:
MAIN:PATTERN:INPUT 65537,1;LOGIC AND;:TRIGGER:MAIN:STATE:
INPUT 65537,1;LOGIC AND;CLOCK CH1;SLOPE RISE;:TRIGGER:MAIN:
SEQUENCE:QUALIFIER LIMIT;LIMIT:WHEN MORETHAN;WIDTH 2.0E-9;
:TRIGGER:MAIN:SEQUENCE:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E-9;
LOWLIMIT 2.0E-9;:TRIGGER:MAIN:SEQUENCE:START:LOGIC AND;INPUT
65537,1;:TRIGGER:MAIN:SEQUENCE:END:LOGIC AND;INPUT 65537,2;
:TRIGGER:MAIN:PULSE:CLASS GLITCH;SOURCE CH1;GLITCH:WIDTH
2.0E-9;WHEN LESSTHAN;POLARITY POSITIVE;:TRIGGER:MAIN:
```

```
PULSE:WIDTH:LOWLIMIT 2.0E–9;HIGHLIMIT 2.0E–9;WHEN WITHIN;
POLARITY POSITIVE;:TRIGGER:DEL.
```

TRIGger:DELay

Sets the delayed trigger level to 50% of the trigger signal or queries the current delayed trigger parameters.

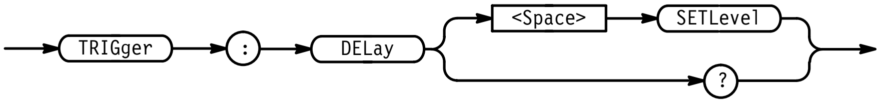
Group

Trigger

Syntax

```
TRIGger:DELay SETLevel
```

```
TRIGger:DELay?
```



Arguments

`SETLevel` sets the delayed trigger level to half way between the minimum and maximum amplitudes of the trigger source input. Using this argument is equivalent to pressing the front panel Set Level to 50% button.

Examples

```
TRIGGER:DELAY SETLEVEL
```

sets the delayed trigger level to 50% of the minimum and maximum values.

```
TRIGGER:DELAY?
```

might return `:TRIGGER:DELAY:TYPE EDGE;BY RUNSAFTER;EDGE:SOURCE CH1;SLOPE RISE;:TRIGGER:DELAY:PULSE:CLASS GLITCH;SOURCE CH1;GLITCH:WIDTH 2.0E–9;WHEN LESSTHAN;POLARITY POSITIVE;:TRIGGER:DELAY:PULSE:WIDTH:LOWLIMIT 2.0E–9;HIGHLIMIT 2.0E–9;WHEN WITHIN;POLARITY POSITIVE;:TRIGGER:DELAY:PATTERN:QUALIFIER LIMIT;LIMIT:WHEN MORETHAN;WIDTH 2.0E–9;:TRIGGER:DELAY:PATTERN:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E–9;LOWLIMIT 2.0E–9;:TRIGGER:DELAY:PATTERN:INPUT 65537,1;LOGIC AND;:TRIGGER:DELAY:STATE:INPUT 65537,1;LOGIC AND;CLOCK CH1;SLOPE RISE;:TRIGGER:DELAY:SEQUENCE:QUALIFIER LIMIT;LIMIT:WHEN MORETHAN;WIDTH 2.0E–9;:TRIGGER:DELAY:SEQUENCE:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E–9;LOWLIMIT 2.0E–9;:TRIGGER:DELAY:SEQUENCE:START:LOGIC AND;INPUT 65537,1;:TRIGGER:DELAY:SEQUENCE:END:LOGIC AND;INPUT 65537,2;:TRIGGER:DELAY:TIME 16.0E–9;EVENTS:COUNT`.

TRIGger:DELay:BY

Selects whether the delayed trigger occurs after a specified number of events or a specified period of time following the main trigger. Using this command is equivalent to setting Delay by in the Delayed Trig menu.

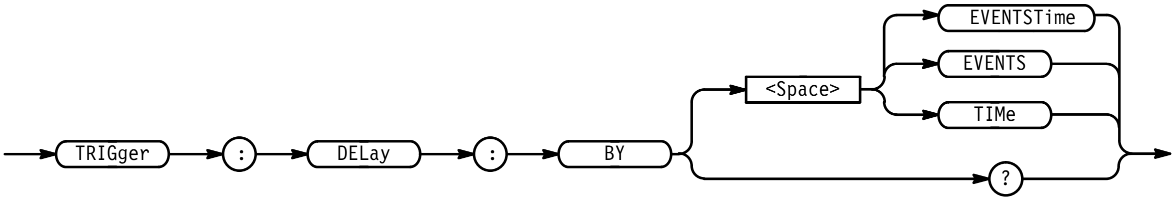
Group Trigger

Related Commands TRIGger:DELay:EVENTS:COUNt, TRIGger:DELay:TIMe

Syntax

```
TRIGger:DELay:BY { EVENTSTime | EVENTS | TIMe }
```

```
TRIGger:DELay:BY?
```



Arguments

EVENTSTime introduces a specified time following a specified number of delay trigger events proceeding the main trigger event. For example in examining a pulse train, you might use the main trigger to detect the start of the train, use the Delay By Events to go to the position of interest within the pulse train, then implement the Time Delay to wait a specified time before starting the data acquisition.

EVENTS sets the delayed trigger to occur after a specified number of trigger events following the main trigger event. The number of events is specified by TRIGger:DELay:EVENTS:COUNt.

TIMe sets the delayed trigger to occur after a specified time following the main trigger event. The time period is specified by TRIGger:DELay:TIMe.

Examples

```
TRIGGER:DELAY:BY TIMe
```

sets the trigger delay to TIME

```
TRIGGER:DELAY:BY?
```

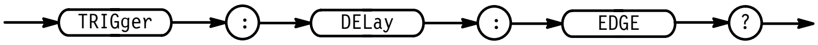
might return EVENTS.

TRIGger:DELay:EDGE? (Query Only)

Returns the trigger coupling, slope, and source for the delayed edge trigger.

Group Trigger

Syntax `TRIGger:DELay:EDGE?`



Examples `TRIGGER:DELAY:EDGE?`
might return `:TRIGGER:EDGE:SOURCE CH1;SLOPE RISE`.

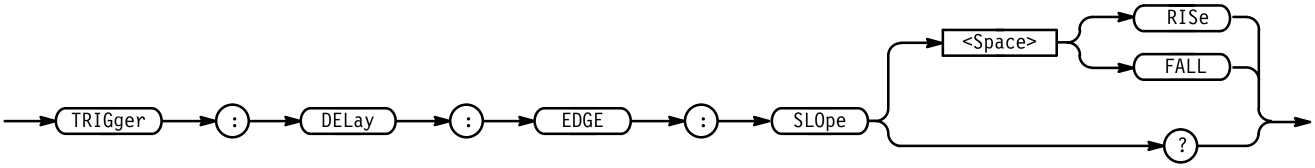
TRIGger:DELay:EDGE:SLOpe

Sets or returns the transition edge (rising or falling) that triggers the logic scope. This command is equivalent to setting Slope in the Delayed Trigger menu.

Group Trigger

Syntax `TRIGger:DELay:EDGE:SLOpe { RISe | FALL }`

`TRIGger:DELay:EDGE:SLOpe?`



Arguments `RISe` instructs the logic scope to trigger on the rising transition (positive) edge of a signal.

`FALL` instructs the logic scope to trigger on the falling transition (negative) edge of a signal.

Examples `TRIGGER:DELAY:EDGE:SLOPE FALL`
sets the logic scope to trigger on the falling transition edge of a signal.

`TRIGGER:DELAY:EDGE:SLOPE?`
might return `RISE`, indicating the delayed trigger occurs on the rising edge of the signal.

TRIGger:DELay:EDGE:SOUrce

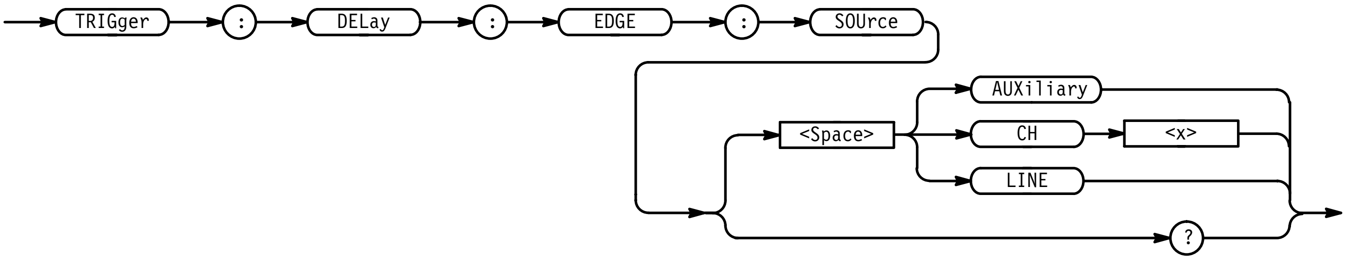
Sets or queries the source for the delayed edge trigger. This command is equivalent to selecting Delayed Trigger Source in the Delayed Pulse Trigger Source side menu.

Group Trigger

Syntax

```
TRIGger:DELay:EDGE:SOUrce { AUXiliary | CH<x> | LINE }
```

```
TRIGger:DELay:EDGE:SOUrce?
```



Arguments `AUXiliary` specifies the rear panel connector labeled AUX TRIGGER INPUT as the delayed edge trigger source.

`CH<x>` specifies one of the 16 input channels as the delayed edge trigger source.

`LINE` specifies the AC line as the delayed edge trigger source.

Examples `TRIGGER:DELAY:EDGE:SOURCE CH1`
specifies channel 1 as the delayed edge trigger source.

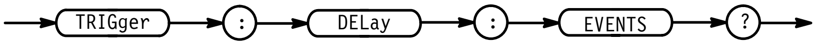
`TRIGGER:DELAY:EDGE:SOURCE?`
might return `CH12` for the delayed edge trigger source.

TRIGger:DELay:EVENTS? (Query Only)

Returns the current setting of the delayed trigger event parameter.

Group Trigger

Syntax TRIGger:DELay:EVENTS?



Returns <NR1> from 2 to 10E+7 indicating the current delayed trigger event parameters.

Examples TRIGGER:DELAY:EVENTS?
might return :TRIGGER:DELAY:EVENTS:COUNT 2

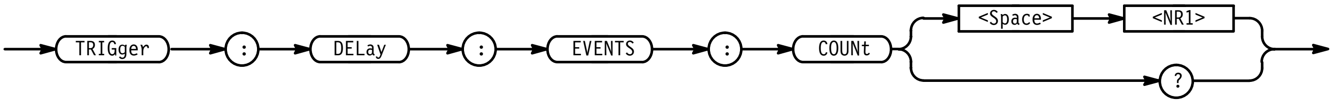
TRIGger:DELay:EVENTS:COUNt

Sets or queries the number of events that must occur before the delayed trigger occurs when TRIGger:DELay:BY is set to EVENTS. Using this command is equivalent to setting the Delay by Events count in the Delayed Edge Events 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 occurs four trigger events after the main trigger.

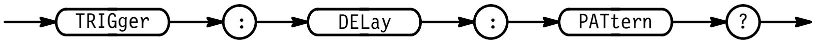
TRIGGER:DELAY:EVENTS:COUNT?
might return 2, indicating that 2 events must occur after the main trigger before the delayed trigger can occur.

TRIGger:DELay:PATtern? (Query Only)

Returns all delayed pattern trigger parameters.

Group Trigger

Syntax `TRIGger:DELay:PATtern?`



Examples `TRIGGER:DELAY:PATTERN?`

might return `:TRIGGER:DELAY:PATTERN:QUALIFIER LIMIT;LIMIT:WHEN MORETHAN;WIDTH 2.0E–9;:TRIGGER:DELAY:PATTERN:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E–9;LOWLIMIT 2.0E–9;:TRIGGER:DELAY:PATTERN:INPUT 65537,1;LOGIC AND` as the current delayed pattern trigger parameters.

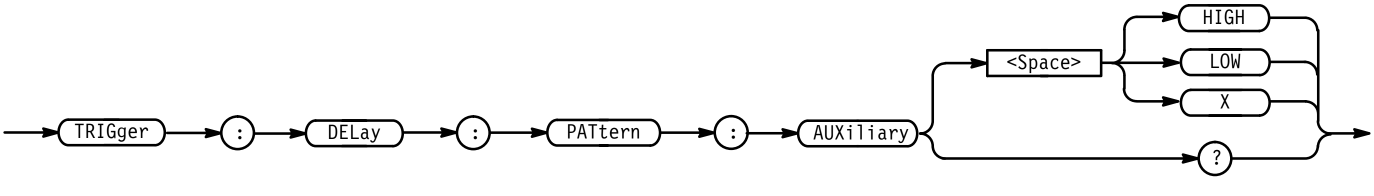
TRIGger:DELay:PATtern:AUXiliary

Sets or returns the Boolean requirement (a logic level of High, Low, or Don’t Care) for the rear-panel connector labeled AUX TRIGGER INPUT. The Boolean requirements for all inputs define the logic pattern that triggers the logic scope. Using this command is the same as setting the auxiliary input bit in the Define Pattern menus found in the Pattern Trigger menu for the delayed trigger.

Group Trigger

Syntax `TRIGger:DElay:PATtern:AUXiliary { HIGH | LOW | X }`

`TRIGger:DELay:PATtern:AUXiliary?`



Arguments `HIGH` specifies that the AUX TRIGGER INPUT must be High relative to the threshold level.

LOW specifies that the AUX TRIGGER INPUT must be Low relative to the threshold level.

X specifies that the AUX TRIGGER INPUT can be either High or Low.

Examples

```
TRIGGER:DELAY:PATTERN:AUXILIARY LOW
```

specifies that the AUX TRIGGER INPUT must be Low to meet the Boolean requirements of the defined pattern.

```
TRIGGER:DELAY:PATTERN:AUXILIARY?
```

might return X, indicating the AUX TRIGGER INPUT may be either High or Low relative to the to the threshold level.

TRIGger:DELay:PATtern:CH<x>

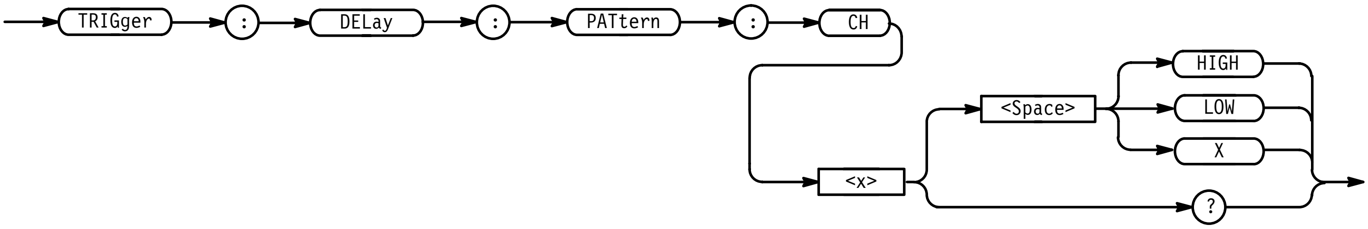
Sets or returns the Boolean requirement (a logic level of High, Low, or Don't Care) for the selected channel. The Boolean requirements for all inputs define the logic pattern that triggers the logic scope. Using this command is the same as setting the selected channel's bit in the Define Pattern menus found in the Pattern Trigger menu for the delayed trigger.

Group Trigger

Syntax

```
TRIGger:DELay:PATtern:CH<x> { HIGH | LOW | X }
```

```
TRIGger:DELay:PATtern:CH<x>?
```



Arguments

HIGH specifies that the selected channel must be High relative to the threshold level.

LOW specifies that the selected channel must be Low relative to the threshold level.

X specifies that the selected channel can be either High or Low relative to the threshold level.

Examples `TRIGGER:DELAY:PATTERN:CH14 HIGH`

specifies that channel 14 must be High to meet the Boolean requirements of the defined pattern for the delayed pattern trigger.

`TRIGGER:DELAY:PATTERN:CH12?`

might return `X`, indicating that channel 12 may be either High or Low relative to the threshold level, to meet the Boolean requirements of the defined pattern for the delayed trigger.

TRIGger:DELay:PATtern:INPut

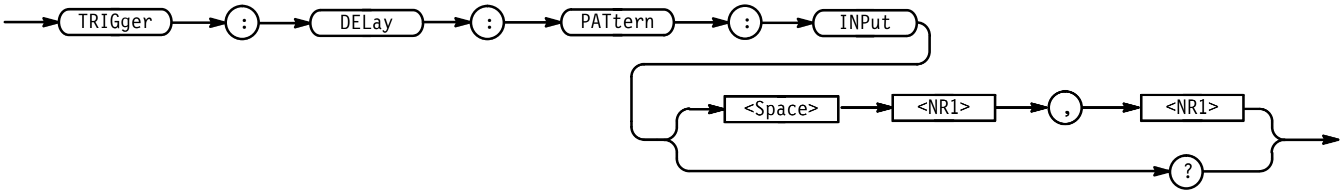
Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

Group Trigger

Syntax

```
TRIGger:DELay:PATtern:INPut <NR1>,<NR1> NOTE: 1 High, 0 low
```

```
TRIGger:DELay:PATtern:INPut?
```



Arguments `<NR1>` is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

`<NR1>` is a value from 0 to 131071. The “Don’t Care” values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the “Don’t Care” status of channels 1 through 16. Bit 17 sets the “Don’t Care” of the Auxiliary Input. A bit value of zero will set the Boolean value plus “Don’t Care” for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples

`TRIGGER:DELAY:PATTERN:INPUT 65535, 65535`
sets all 16 channels to trigger on High input values and sets the Auxiliary Input to “Don’t Care”.

`TRIGGER:DELAY:PATTERN:INPUT 1, 1`
sets channel 1 High and all other inputs to to “Don’t Care”.

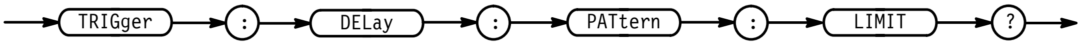
`TRIGGER:DELAY:PATTERN:INPUT?`
might return `3,3`, indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to “Don’t Care”.

TRIGger:DELay:PATtern:LIMIT? (Query Only)

Returns the time-qualified pattern parameters for the delayed pattern trigger.

Group Trigger

Syntax `TRIGger:DELay:PATtern:LIMIT?`



Examples

`TRIGGER:DELAY:PATTERN:LIMIT?`
might return `:TRIGGER:DELAY:PATTERN:LIMIT:WHEN MORETHAN;WIDTH 2.0E-9`.

TRIGger:DELay:PATtern:LIMIT:WHEn

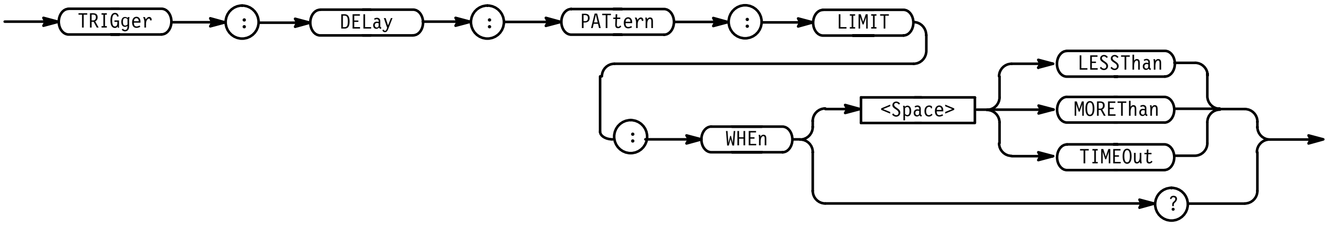
Sets or queries the definition of “when” the Boolean requirements must cease to meet the conditions of the defined pattern to trigger the logic scope.

Group Trigger

Syntax

```
TRIGger:DELay:PATtern:LIMIT:WHEn { LESSThan | MOREThan
| TIMEOut }
```

```
TRIGger:DELay:PATtern:LIMIT:WHEn?
```



Arguments `LESSThan` specifies that the logic scope will trigger if the trigger inputs cease to meet the Boolean requirements before the specified time.

`MOREThan` specifies that the logic scope will trigger if the trigger inputs cease to meet the Boolean requirements after the specified time.

`TIMEOut` specifies that the logic scope will trigger after the specified time elapses and the trigger inputs still meet the Boolean requirements.

Examples `TRIGGER:DELAY:PATTERN:LIMIT:WHEN LESSTHAN`

instructs the logic scope to trigger if the trigger inputs cease to meet the LessThan pattern requirements before the specified time.

`TRIGGER:DELAY:PATTERN:LIMIT:WHEN?`

might return `TIMEOUT`, indicating the logic scope will trigger after the specified time if the trigger inputs still meet the TimeOut pattern requirements.

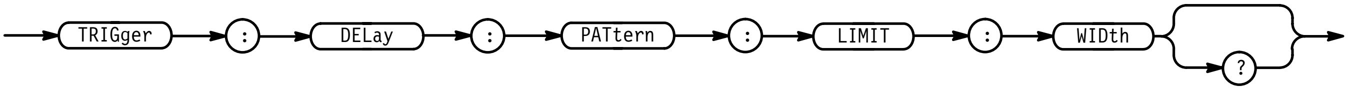
TRIGger:DELay:PATtern:LIMIT:WIDth

Sets or queries the time limit for the Trigger When definitions of Less Than, More Than, or Time Out. This is equivalent to pressing side menu Time for delayed pattern time triggers.

Group Trigger

Syntax `TRIGger:DELay:PATtern:LIMIT:WIDth`

`TRIGger:DELay:PATtern:LIMIT:WIDth?`



Arguments `<NR3>` is the time limit from 2 ns to 1.0 second.

Examples

```
TRIGGER:DELAY:PATTERN:LIMIT:WIDTH 1.0E-3
```

sets the time limit width to 1.0 ms.

```
TRIGGER:DELAY:PATTERN:LIMITWIDTH?
```

might return 2.0E–9, indicating the time limit width is 2 ns.

TRIGger:DELay:PATtern:LOGIc

Sets or queries the combinatorial logic pattern applied to the delayed pattern trigger.

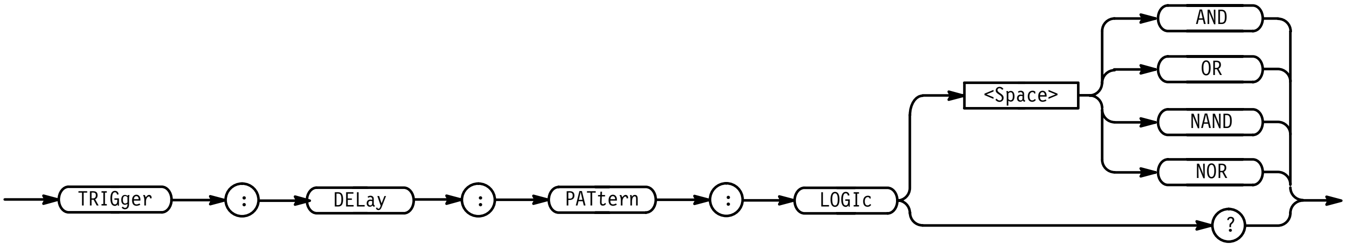
Group

Trigger

Syntax

```
TRIGger:DELay:PATtern:LOGIc { AND | OR | NAND | NOR }
```

```
TRIGger:DELay:PATtern:LOGIc?
```



Arguments

AND applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

OR applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

NAND applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

NOR applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

```
TRIGGER:DELAY:PATTERN:LOGIC AND
```

applies AND logic to the input channels.

`TRIGGER:DELAY:PATTERN:LOGIC AND?`
might return `OR`, indicating that the Delayed Pattern Trigger Logic is set to OR.

TRIGger:DELay:PATtern:QUAlifier

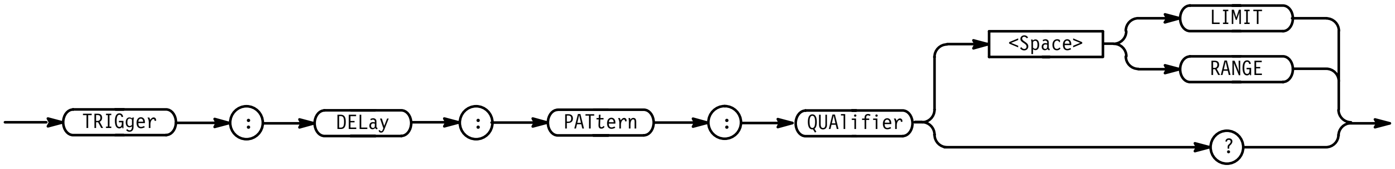
Sets or queries the Time Qualifier used to define when the logic scope triggers. Using this command is equivalent to selecting the time qualifier Limit or Range for the delayed pattern trigger.

Group Trigger

Syntax

```
TRIGger:DELay:PATtern:QUAlifier { LIMIT | RANGE }
```

```
TRIGger:DELay:PATtern:QUAlifier?
```



Arguments `LIMIT` defines a maximum time limit for the delayed trigger event to occur.

`RANGE` defines a range of time for the delayed trigger event to occur.

Examples `TRIGGER:DELAY:PATTERN:QUALIFIER RANGE`
specifies that the delayed trigger event must occur either within or outside a specified amount of time.

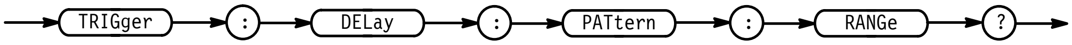
`TRIGGER:DELAY:PATTERN:QUALIFIER RANGE?`
might return `LIMIT`, indicating there is a maximum time limit for the delayed trigger event.

TRIGger:DELay:PATtern:RANGE? (Query Only)

Returns the delayed pattern trigger range parameters.

Group Trigger

Syntax `TRIGger:DELay:PATtern:RANGe?`



Examples `TRIGGER:DELAY:PATTERN:RANGE?`
might return `:TRIGGER:DELAY:PATTERN:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E-9;LOWLIMIT 2.0E-9`.

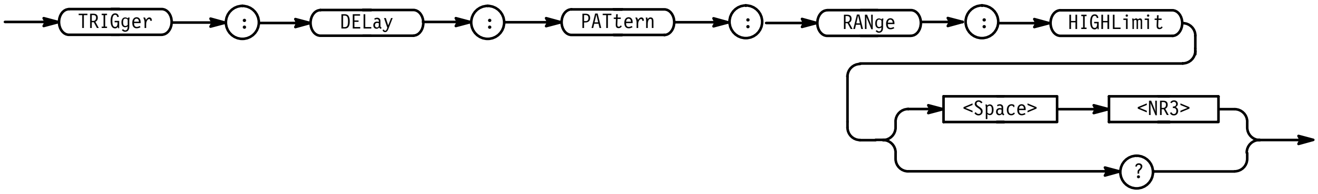
TRIGger:DELay:PATtern:RANge:HIGHLimit

Sets or queries the upper time limit range for the range-qualified delay trigger. This command is equivalent to selecting the Upper Limit for delayed pattern triggers (range selected) as the time qualifier.

Group Trigger

Syntax `TRIGger:DELay:PATtern:RANge:HIGHLimit <NR3>`

`TRIGger:DELay:PATtern:RANge:HIGHLimit?`



Arguments `<NR3>` is the upper limit of the range in seconds.

Examples `TRIGGER:DELAY:PATTERN:RANGE:HIGHLIMIT 6.5E-9`
sets the upper time limit of the range to 6.5 ns.

`TRIGGER:DELAY:PATTERN:RANGE:HIGHLIMIT?`
might return 100.0E–9, indicating the upper time limit is 100 ns.

TRIGger:DELay:PATtern:RANge:LOWLimit

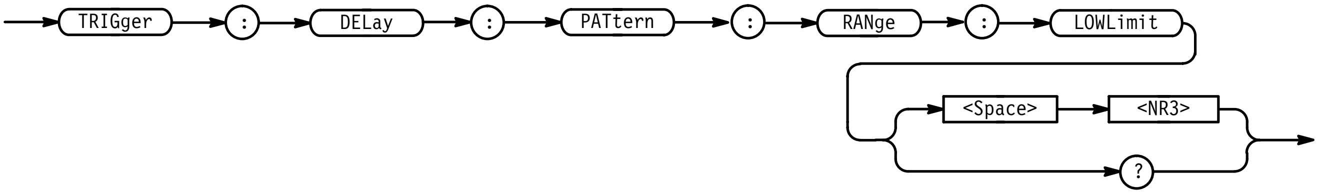
Sets or queries the lower time limit range for the range-qualified delay trigger. This command is equivalent to selecting Lower Limit for delayed pattern triggers (range selected) as the time qualifier.

Group Trigger

Syntax

```
TRIGger:DELay:PATtern:RANge:LOWLimit <NR3>
```

```
TRIGger:DELay:PATtern:RANge:LOWLimit?
```



Arguments `<NR3>` is the lower limit of the range in seconds.

Examples

```
TRIGGER:DELAY:PATTERN:RANGE:LOWLIMIT 2.5E-9
```

sets the lower time limit of the range to 2.5 ns.

```
TRIGGER:DELAY:PATTERN:RANGE:LOWLIMIT?
```

might return 100.0E–9, indicating the lower time limit of the range is 100 ns.

TRIGger:DELay:PATtern:RANge:WHEn

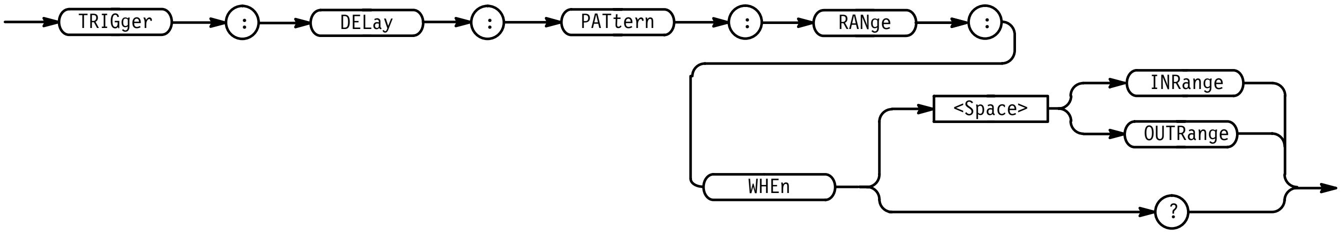
Sets or queries when the channel inputs must cease to meet the Boolean requirements of the Delay Pattern definition to trigger the logic scope.

Group Trigger

Syntax

```
TRIGger:DELay:PATtern:RANge:WHEn { INRange | OUTRange }
```

```
TRIGger:DELay:PATtern:RANge:WHEn?
```



Arguments `INRange` specifies that the trigger inputs must meet the Boolean requirements of the Delayed Trigger pattern definition for a period of time that ends within the user specified time range. The range is 2.0 ns to 1.0 second.

OUTRange specifies that the trigger inputs must meet the Boolean requirements of the Delayed Trigger pattern definition for a period of time that ends either before or after the user specified time range. The range is 2.0 ns to 1.0 second.

Examples TRIGGER:DELAY:PATTERN:RANGE:WHEN INRANGE specifies that the channel inputs must cease to meet the Boolean requirements of the delayed trigger pattern definition within the time range defined.

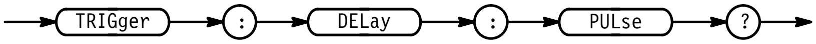
TRIGGER:DELAY:PATTERN:RANGE:WHEN INRANGE?
might return INRANGE, indicating the channel inputs must meet the delayed trigger pattern requirements within the specified time range.

TRIGger:DELay:PULse? (Query Only)

Returns the delayed pulse trigger parameters.

Group Trigger

Syntax TRIGger:DELay:PULse?



Examples TRIGGER:DELAY:PULSE?
might return TRIGGER:DELAY:PATTERN:CLASS WIDTH;TRIG WHEN WITHIN;SOURCE CH1;POLARITY POSITIVE as the current delayed pulse trigger parameters.

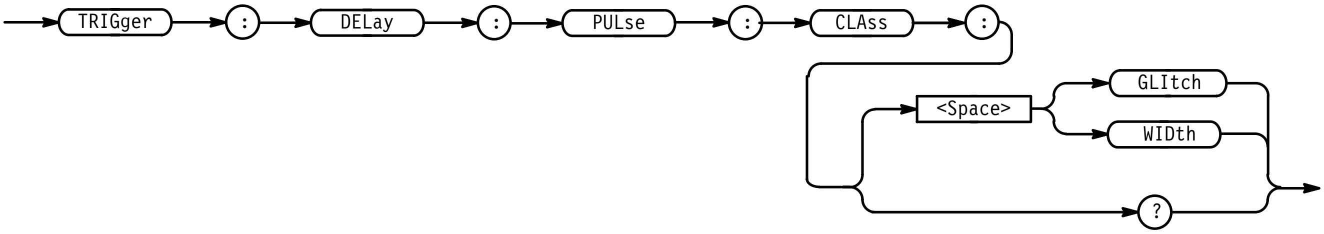
TRIGger:DELay:PULse:CLAss

Sets or queries the class used for delayed pulse triggering. This command is equivalent to selecting Glitch or Width from the Delayed Pulse Trigger Class menu.

Group Trigger

Syntax TRIGger:DELay:PULse:CLAss { GLItch | WIDth }

TRIGger:DELay:PULse:CLAss?



Arguments `GLItch` initiates a trigger when a pulse of the specified polarity and width is found; works in conjunction with TRIGger:DELay:PULse:GLItch:POLarity and TRIGger:DELay:PULse:GLItch:WIDth.

`WIDth` initiates a trigger when a pulse of the specified polarity that is either inside or outside the limits specified by TRIGger:DELay:PULse: WIDth:LOW-Limit and TRIGger:DELay:PULse:WIDth:HIGHLimit is found. The polarity is selected using the TRIGger:DELay:PULse:WIDth:POLarity command.

Examples `TRIGGER:DELay:PULSE:CLASS WIDTH`
specifies the width class for the delayed pulse trigger.

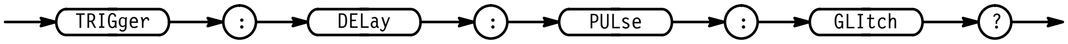
`TRIGGER:DELay:PULSE:CLASS WIDTH?`
might return `GLITCH`, indicating the logic scope will trigger when a pulse of the specified polarity and width is found.

TRIGger:DELay:PULse:GLItch? (Query Only)

Returns the delayed trigger parameters for Pulse Glitch triggers.

Group Trigger

Syntax `TRIGger:DELay:PULse:GLItch?`



Examples `TRIGGER:DELAY:PULSE:GLITCH?`
might return `:TRIGGER:DELAY:PULSE:CLASS GLITCH;SOURCE CH1; GLITCH:WIDTH 2.0E-9;POLARITY POSITIVE`.

TRIGger:DELay:PULse:GLItch:POLarity

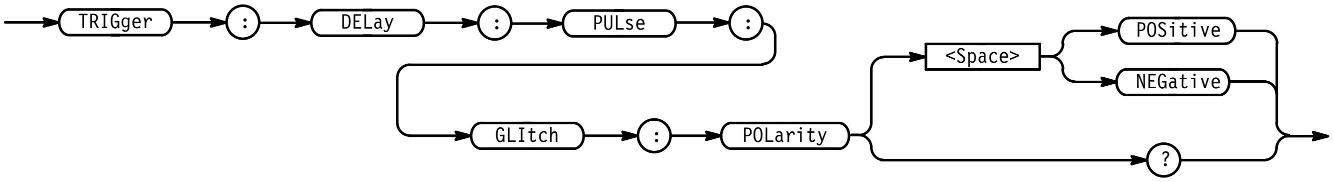
Sets or returns the transition direction for Glitch triggers. This command is equivalent to setting polarity for delayed pulse triggers.

Group Trigger

Syntax

```
TRIGger:DELay:PULse:GLItch:POLarity { POSitive | NEGative }
```

```
TRIGger:DELay:PULse:GLItch:POLarity?
```



Arguments POSITIVe specifies the narrower glitch portion of the waveform as the positive-going portion of the waveform cycle.

NEGative specifies the narrower glitch portion of the waveform as the negative-going portion of the waveform cycle.

Examples

```
TRIGGER:DELAY:PULSE:GLITCH:POLARITY POSITIVE
```

specifies that the polarity of the glitch is positive.

```
TRIGGER:DELAY:PULSE:GLITCH:POLARITY POSITIVE?
```

might return NEGATIVE as the polarity setting.

TRIGger:DELay:PULse:GLItch:WHEN

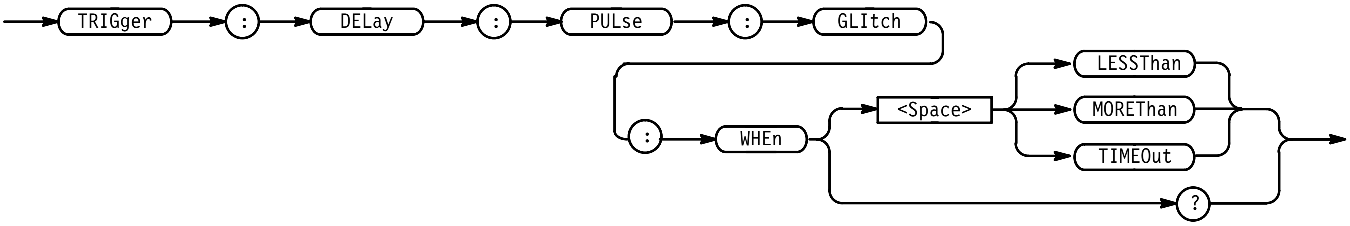
Sets or queries the delay trigger pulse glitch filter. Glitches are either filtered out or accepted using a Time or Time format.

Group Trigger

Syntax

```
TRIGger:DELay:PULse:GLItch:WHEn { LESSThan | MOREThan | TIMEOut }
```

```
TRIGger:DELay:PULse:GLItch:WHEn?
```



Arguments

MOREThan specifies that triggering occurs when the width of the glitch is more than the specified time.

LESSThan specifies that triggering occurs when the width of the glitch is less than the specified time.

TIMEOut specifies that triggering occurs when the glitch does not occur within the specified time. Triggering occurs at the point where the specified time elapses.

Examples

```
TRIGGER:DELAY:PULSE:GLITCH:WHEN MORETHAN
```

instructs the logic scope to trigger when the width of the glitch is greater than the time specified.

```
TRIGGER:DELAY:PULSE:GLITCH:WHEN?
```

might return TIMEOUT, indicating that triggering occurs when the glitch does not occur within the specified time.

TRIGger:DELay:PULse:GLItch:WIDth

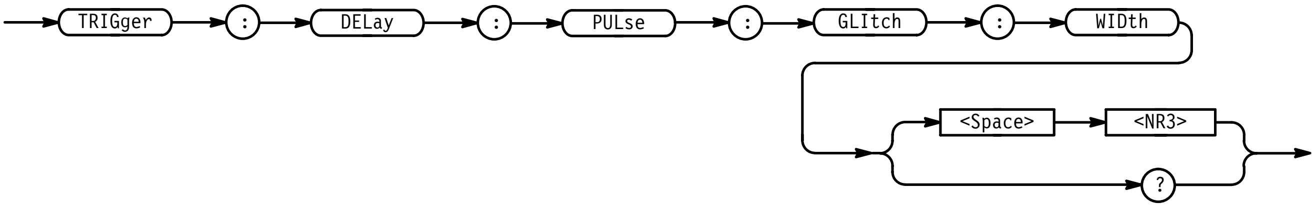
Sets or returns the width for the delayed pulse glitch trigger. This command is equivalent to selecting Polarity & Width in the Trigger menu.

Group Trigger

Syntax

```
TRIGger:DELay:PULse:GLItch:WIDth <NR3>
```

```
TRIGger:DELay:PULse:GLItch:WIDth?
```



Arguments <NR3> is the width of the glitch in seconds.

Examples `TRIGGER:DELAY:PULSE:GLITCH:WIDTH 15E-6`
sets the width of the glitch to 15 μs.

`TRIGGER:DELAY:PULSE:GLITCH:WIDTH?`
might return `500E-3`, indicating the glitch width is 0.5 seconds.

TRIGger:DELay:PULse:SOUrce

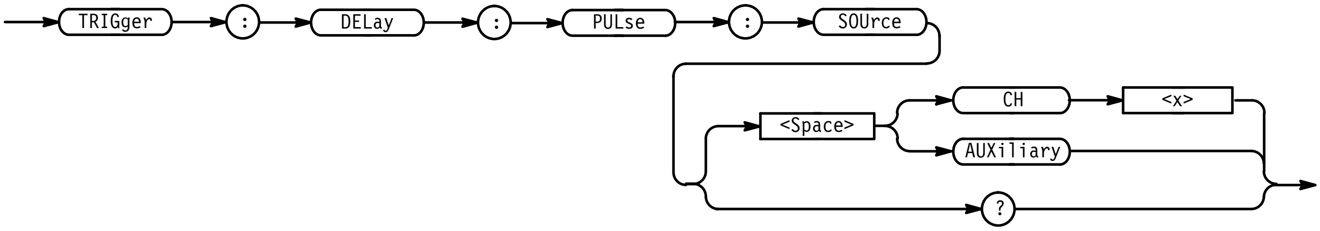
Sets or queries the source for the Delayed Pulse trigger. This command is equivalent to selecting the Delayed Trigger Source in the delayed pulse Source side menu.

Group Trigger

Syntax

```
TRIGger:DELay:PULse:SOUrce { CH<x> | AUXiliary }

TRIGger:DELay:PULse:SOUrce?
```



Arguments `CH<x>` specifies one of 16 input channels as the delayed pulse trigger source.

`AUXiliary` specifies the rear panel connector labeled AUX TRIGGER INPUT as the delayed pulse trigger source.

Examples `TRIGGER:DELAY:PULSE:SOURCE CH2`
selects Channel 2 as the source for the delayed pulse trigger.

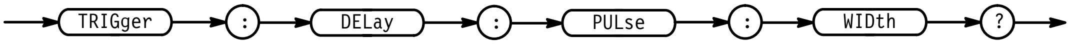
`TRIGGER:DELAY:PULSE:SOURCE?`
might return 8 as the delayed pulse trigger source.

TRIGger:DELay:PULse:WIDth? (Query Only)

Returns the width parameters for the Delayed Pulse width trigger.

Group Trigger

Syntax `TRIGger:DELay:PULse:Width?`



Examples `TRIGGER:DELAY:PULSE:WIDTH?`
might return `:TRIGGER:DELAY:PULSE:WIDTH:LOWLIMIT 2.0E−9;HIGHLIMIT 2.0E−9;WHEN WITHIN;POLARITY POSITIVE` as the current delayed pulse trigger parameters.

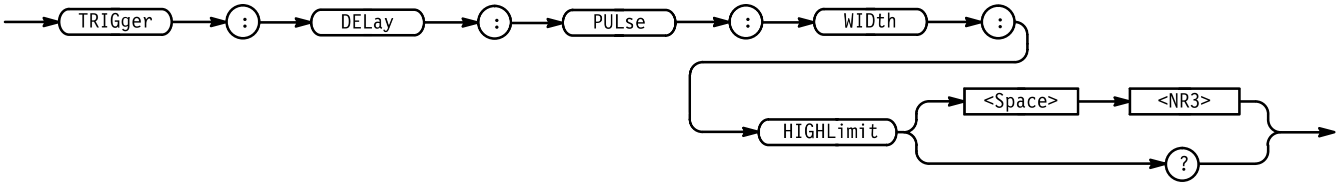
TRIGger:DELay:PULse:WIDth:HIGHLimit

Sets or queries the upper limit for the delayed pulse width trigger. Using this command is equivalent to setting the Upper Limit in the Pulse Width Trigger side menu.

Group Trigger

Syntax `TRIGger:DELay:PULse:WIDth:HIGHLimit <NR3>`

`TRIGger:DELay:PULse:WIDth:HIGHLimit?`



Arguments <NR3> is the upper limit in seconds.

Examples

`TRIGGER:DELAY:PULSE:WIDTH:HIGHLimit 2.0E-3`
sets the delayed pulse width to 2 ms.

`TRIGGER:DELAY:PULSE:WIDTH:HIGHLIMIT?`
might return `2.0E-9`, indicating the delayed pulse width is 2 ns.

TRIGger:DELay:PULse:WIDth:LOWLimit

Sets or queries the lower limit for the delayed pulse width trigger. Using this command is equivalent to setting the Lower Limit in the Pulse Width Trigger side menu.

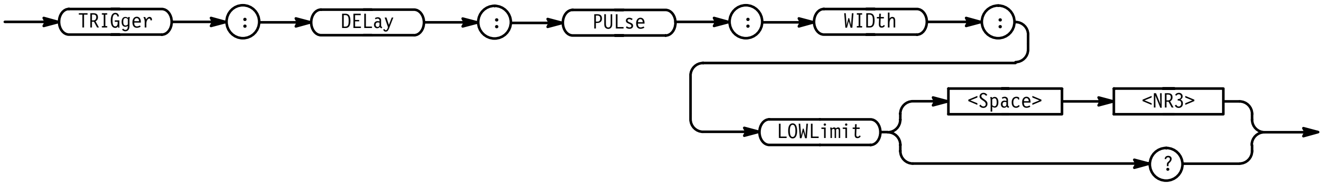
Group

Trigger

Syntax

```
TRIGger:DELay:PULse:WIDth:LOWLimit <NR3>
```

```
TRIGger:DELay:PULse:WIDth:LOWLimit?
```



Arguments

`<NR3>` is the lower limit in seconds.

Examples

`TRIGGER:DELAY:PULSE:WIDTH:LOWLIMIT 1.0E-3`
sets the delayed pulse width to 1 ms.

`TRIGGER:DELAY:PULSE:WIDTH:LOWLIMIT?`
might return `2.0E-9`, indicating the delayed pulse width is 2 ns.

TRIGger:DELay:PULse:WIDth:POLarity

Sets or queries the polarity for the delay pulse width trigger. Using this command is equivalent to selecting Polarity in the Pulse Width Polarity side menu.

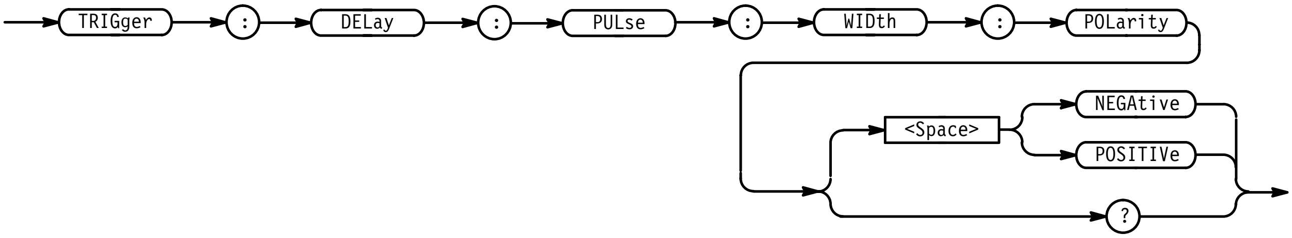
Group

Trigger

Syntax

```
TRIGger:DELay:PULse:WIDth:POLarity { NEGAtive | POSITIVe }
```

```
TRIGger:DELay:PULse:WIDth:POLarity?
```



Arguments

NEGAtive specifies a negative pulse.

POSITIVe specifies a positive pulse.

Examples

TRIGGER:DELAY:PULSE:WIDTH:POLARITY POSITIVE
specifies positive pulse width polarity.

TRIGGER:DELAY:PULSE:WIDTH:POLARITY?
might return 2.0E−6, indicating the delayed pulse width is 2 μs.

TRIGger:DELay:PULse:WIDth:WHEn

Sets or queries the delayed pulse width parameters controlling the logic scope triggers. The parameters define when the trigger event must complete to trigger the logic scope. This command is equivalent to selecting the condition in the Pulse Width Trig When side menu.

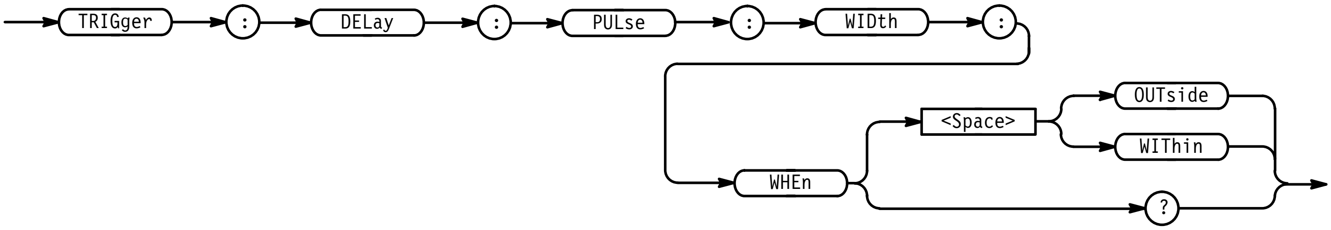
Group

Trigger

Syntax

```
TRIGger:DELay:PULse:WIDth:WHEn { OUTside | WIThin }

TRIGger:DELay: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:DELay:PULse:WIDth:HIGHLimit and TRIGger:DELay: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:DELay:PULse:WIDth:HIGHLimit and TRIGger:DELay:PULse:WIDth:LOWLimit commands respectively.

Examples `TRIGGER:DELAY:PULSE:WIDTH:WHEN OUTSIDE`
specifies a trigger when the duration of the pulse is greater than the high limit or less than the low limit.

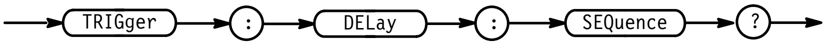
`TRIGGER:DELAY:PULSE:WIDTH:WHEN?`
returns either `OUTSIDE` or `WITHIN`, indicating the conditions for generating a pulse trigger.

TRIGger:DELay:SEQuence? (Query Only)

Returns delay sequence trigger parameters.

Group Trigger

Syntax `TRIGger:DELay:SEQuence?`



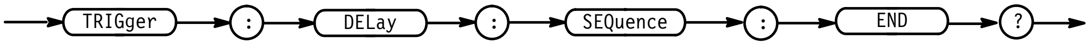
Examples `TRIGGER:DELAY:SEQUENCE?`
might return `:TRIGGER:DELAY:SEQUENCE:QUALIFIER LIMIT;LIMIT:WHEN MORETHAN;WIDTH 2.0E-9;:TRIGGER:DELAY:SEQUENCE:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E-9;LOWLIMIT 2.0E-9;:TRIGGER:DELAY:SEQUENCE:START:LOGIC AND;INPUT 65537,1;:TRIGGER:DELAY:SEQUENCE:END:LOGIC AND;INPUT 65537,2`.

TRIGger:DELay:SEQuence:END? (Query Only)

Returns the pattern definition of the end sequence for the delayed trigger.

Group Trigger

Syntax `TRIGger:DELay:SEQuence:END?`



Examples

`TRIGGER:DELAY:SEQUENCE:END?`
might return `:TRIGGER:DELAY:SEQUENCE:END:LOGIC AND;INPUT 65537,2`.

TRIGger:DELay:SEQuence:END:AUXiliary

Sets or queries the logic level expected at the rear panel connector labeled AUX TRIGGER INPUT as part of the end pattern definition for delayed sequence trigger.

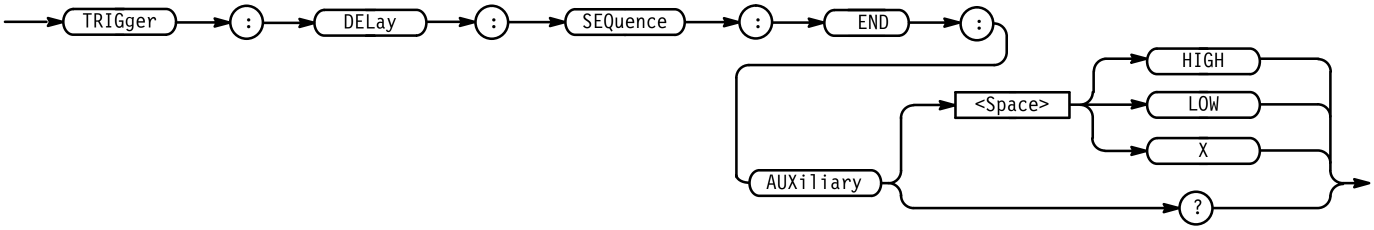
Group

Trigger

Syntax

```
TRIGger:DELay:SEQuence:END:AUXiliary { HIGH | LOW | X }
```

```
TRIGger:DELay:SEQuence:END:AUXiliary?
```



Arguments

`HIGH` specifies that the Aux Trigger Input must be High relative to the threshold level.

`LOW` specifies that the Aux Trigger Input must be Low relative to the threshold level.

`X` specifies that the Aux Trigger Input can be either High or Low relative to the threshold level.

Examples

`TRIGGER:DELAY:SEQUENCE:END:AUXILIARY HIGH`
specifies the Aux Trigger Input is High relative to the threshold level.

`TRIGGER:DELAY:SEQUENCE:END:AUXILIARY?`
might return `X`, indicating the Aux Trigger Input is either High or Low.

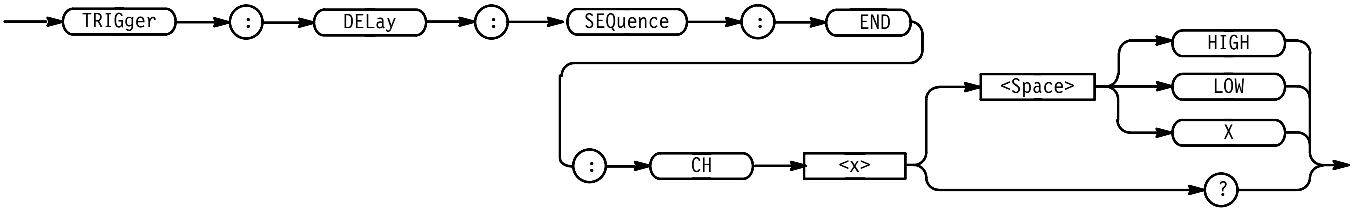
TRIGger:DELay:SEQuence:END:CH<x>

Sets or queries the logic level expected at the channel input as defined by `Ch<x>` (part of the end pattern definition for the delay sequence trigger).

Group Trigger

Syntax `TRIGger:DELay:SEQuence:END:CH<x> { HIGH | LOW | X } NOTE:` defines `CH<x>` pattern bit to High, Low, or Don’t Care.

`TRIGger:DELay:SEQuence:END:CH<x>?`



Arguments `HIGH` specifies that the selected channel must be High relative to the threshold level.

`LOW` specifies that the selected channel must be Low relative to the threshold level.

`X` specifies that the selected channel can be either High or Low relative to the threshold level.

Examples `TRIGGER:DELAY:PATTERN:CH14 HIGH`
specifies that channel 14 must be High to meet the Boolean requirements of the defined pattern for the delayed trigger.

`TRIGGER:DELAY:PATTERN:CH12?`
might return `X`, indicating the specified channel may be either High or Low.

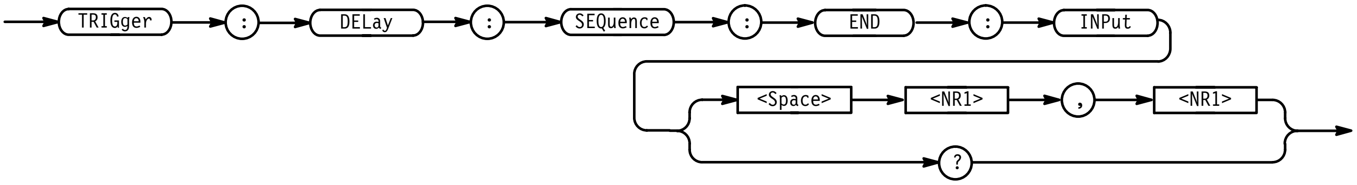
TRIGger:DELay:SEQuence:END:INPut

Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

Group Trigger

Syntax `TRIGger:DELay:SEQuence:END:INPut <NR1>,<NR1> NOTE: 1-high,0-low`

```
TRIGger:DELay:SEQuence:END:INPut?
```



Arguments

`<NR1>` is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

`<NR1>` is a value from 0 to 131071. The “Don’t Care” values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the “Don’t Care” status of channels 1 through 16. Bit 17 sets the “Don’t Care” of the Auxiliary Input. A bit value of zero will set the Boolean value to “Don’t Care” for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples

```
TRIGGER:DELAY:SEQUENCE:END:INPUT 65535, 65535
```

sets all 16 channels to trigger on High input values and sets the Auxiliary Input to “Don’t Care”.

```
TRIGGER:DELAY:SEQUENCE:END:INPUT 1, 1
```

sets channel 1 High and all other inputs to “Don’t Care”.

```
TRIGGER:DELAY:SEQUENCE:END:INPUT?
```

might return `3,3` indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to “Don’t Care”.

TRIGger:DELay:SEQuence:END:LOGIC

Sets or queries the combinatorial logic pattern applied to the end pattern for the delay sequence trigger input.

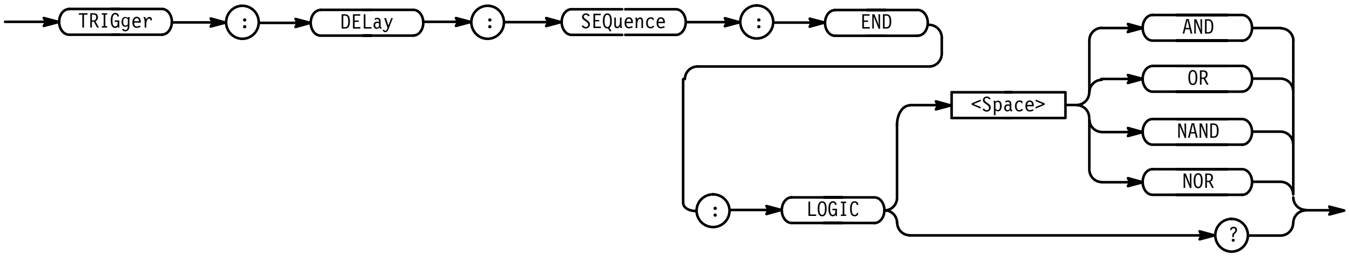
Group

Trigger

Syntax

```
TRIGger:DELay:SEQuence:END:LOGIC { AND | OR | NAND | NOR }
```

```
TRIGger:DELay:SEQuence:END:LOGIC?
```



Arguments

AND applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

OR applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

NAND applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

NOR applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

```
TRIGGER:DELAY:SEQUENCE:END:LOGIC AND
```

applies AND logic to the input channels.

```
TRIGGER:DELAY:SEQUENCE:END:LOGIC?
```

might return :TRIGGER:DELAY:SEQUENCE:END:LOGIC AND.

TRIGger:DELay:SEQuence:LIMIT? (Query Only)

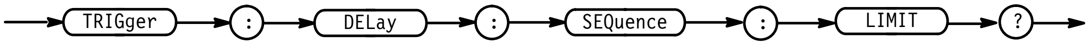
Returns the limit parameters of the delayed sequence trigger. The limit parameters define when the trigger inputs must cease to meet the Boolean requirements to trigger the logic scope.

Group

Trigger

Syntax

```
TRIGger:DELay:SEQuence:LIMIT?
```



Examples `TRIGGER:DELAY:SEQUENCE:LIMIT?`
might return `:TRIGGER:DELAY:SEQUENCE:LIMIT:WHEN MORETHAN;WIDTH 2.0E-9`.

TRIGger:DELay:SEQuence:LIMIT:WHEn

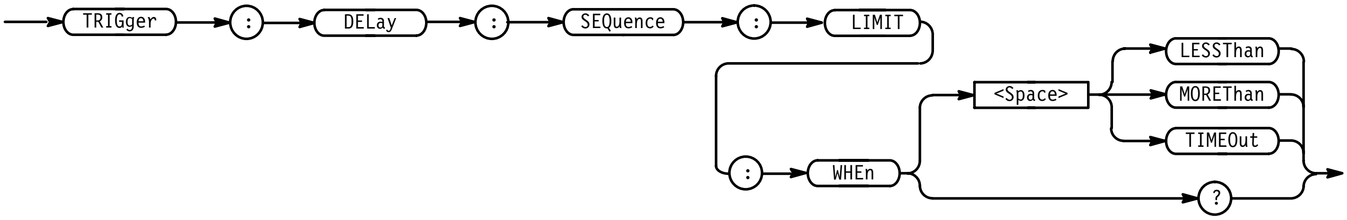
Sets or queries the definition of when the Boolean requirements must cease to meet the conditions of the defined pattern to trigger the logic scope.

Group Trigger

Syntax

```
TRIGger:DELay:SEQuence:LIMIT:WHEn { LESSThan | MOREThan |
TIMEOut }
```

```
TRIGger:DELay:SEQuence:LIMIT:WHEn?
```



Arguments `LESSThan` specifies that the logic scope triggers if the trigger inputs cease to meet the Boolean requirements before the specified time.

`MOREThan` specifies that the logic scope triggers if the trigger inputs cease to meet the Boolean requirements after the specified time.

`TIMEOut` specifies that the logic scope triggers after the specified amount of time elapses and the trigger inputs still meet the Boolean requirements.

Examples `TRIGGER:DELAY:SEQUENCE:LIMIT:WHEN LESSTHAN`
instructs the logic scope to trigger if the trigger inputs cease to meet the Boolean requirements before the specified time.

`TRIGGER:DELAY:SEQUENCE:LIMIT:WHEN?`
might return `MORETHAN`, indicating the logic scope will trigger if the trigger inputs fail to meet the specified requirements before the specified time.

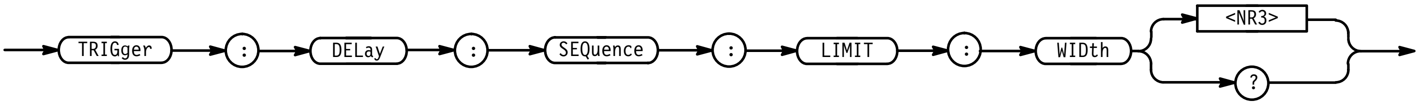
TRIGger:DELay:SEQuence:LIMIT:WIDTH

Sets or queries the time limit for the trigger width definitions of Less Than or More Than. This is equivalent to pressing Time in the delayed pattern time triggers side menu.

Group Trigger

Syntax `TRIGger:DELay:SEQuence:LIMIT:WIDTH <NR3>`

`TRIGger:DELay:SEQuence:LIMIT:WIDTH?`



Arguments `<NR3>` is the time limit from 2.0 ns to 1.0 second.

Examples `TRIGGER:DELAY:PATTERN:LIMIT:WIDTH 1.0E-3`
sets the trigger width time limit to 1.0 ms.

`TRIGGER:DELAY:PATTERN:LIMITWIDTH?`
might return `1.0 sec`, indicating the trigger width time limit.

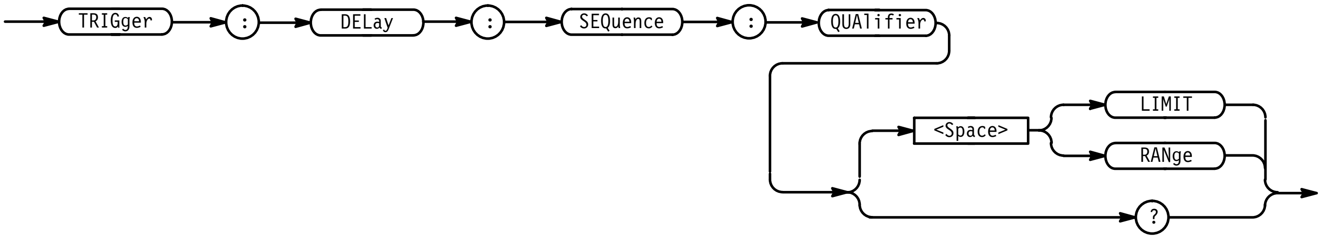
TRIGger:DELay:SEQuence:QUAlifier

Sets or queries the time qualifier used to define when the logic scope triggers. Using this command is equivalent to selecting the time qualifier Limit or Range for the delay sequence trigger.

Group Trigger

Syntax `TRIGger:DELay:SEQuence:QUAlifier { LIMIT | RANge }`

`TRIGger:DELay:SEQuence:QUAlifier?`



Arguments `LIMIT` specifies that there is a maximum time limit for the delayed trigger event to occur.

`RANge` specifies that there is a range of time for the delayed trigger event to occur.

Examples `TRIGGER:DELAY:SEQUENCE:QUALIFIER LIMIT`
specifies that there is a maximum time limit for the delayed trigger event.

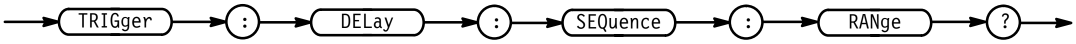
Examples `TRIGGER:DELAY:SEQUENCE:QUALIFIER?`
might return `RANGE`, indicating there is a range of time for the delayed trigger event.

TRIGger:DELay:SEQuence:RANge? (Query Only)

Returns the delayed sequence trigger range parameters.

Group Trigger

Syntax `TRIGger:DELay:SEQuence:RANge?`



Examples `TRIGGER:DELAY:SEQUENCE:RANGE?`
might return `:TRIGGER:DELAY:SEQUENCE:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E-9;LOWLIMIT 2.0E-9`.

TRIGger:DELay:SEQuence:RANge:HIGHLimit

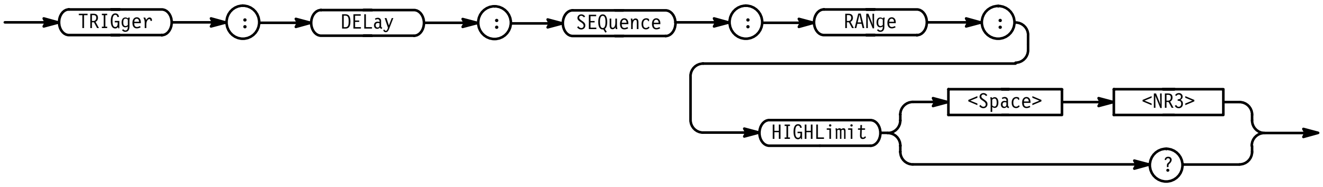
Sets or queries the value for the upper time limit of the range for the range-qualified delayed trigger. Using this command is equivalent to selecting upper limit for delayed sequence triggers with range selected as the time qualifier.

Group Trigger

Syntax

```
TRIGger:DELay:SEQuence:RANge:HIGHLimit <NR3>
```

```
TRIGger:DELay:SEQuence:RANge:HIGHLimit?
```



Arguments <NR3> is the upper time limit of the range in seconds.

Examples

```
TRIGGER:DELAY:SEQUENCE:RANGE:HIGHLIMIT 1.0E+0
```

sets the range upper time limit to 1 second.

```
TRIGGER:DELAY:SEQUENCE:RANGE:HIGHLIMIT?
```

might return 1, indicating the range upper time limit is set to 1 second.

TRIGger:DELay:SEQuence:RANge:LOWLimit

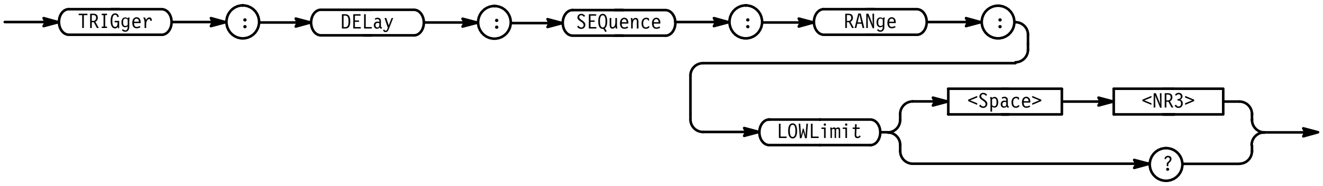
Sets or queries the value for the lower time limit of the range for the range-qualified delayed trigger. Using this command is equivalent to selecting lower limit for delayed sequence triggers with range selected as the time qualifier.

Group Trigger

Syntax

```
TRIGger:DELay:SEQuence:RANge:LOWLimit <NR3>
```

```
TRIGger:DELay:SEQuence:RANge:LOWLimit?
```



Arguments <NR3> is the lower time limit of the range in seconds.

Examples `TRIGGER:DELAY:SEQUENCE:RANGE:LOWLIMIT 1.0E+0`
sets the range lower time limit to 1 second.

`TRIGGER:DELAY:SEQUENCE:RANGE:LOWLIMIT?`
might return 1, indicating the range lower time limit is set to 1 second.

TRIGger:DELay:SEQuence:RANge:WHEn

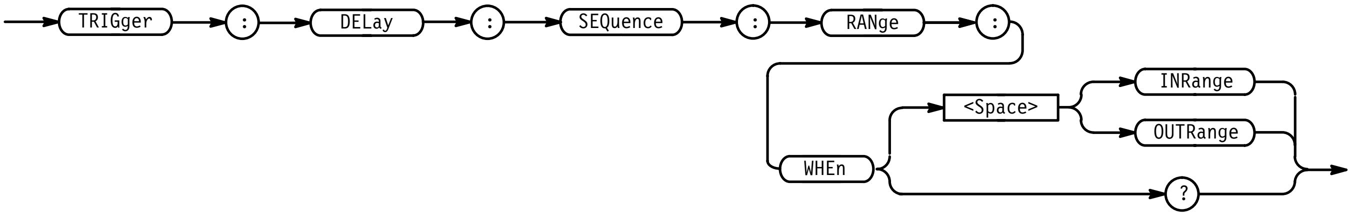
Sets or queries when the channel inputs must cease to meet the Boolean requirements of the pattern definition for the delayed sequence trigger.

Group Trigger

Syntax

```
TRIGger:DELay:SEQuence:RANge:WHEn { INRange | OUTRange }
```

```
TRIGger:DELay:SEQuence:RANge:WHEn?
```



Arguments INRange specifies that the trigger inputs must meet the Boolean requirements of the pattern definition for the delayed sequence trigger for a period of time that ends within the user specified time range.

OUTRange specifies that the trigger inputs must meet the Boolean requirements of the pattern definition for the delayed sequence trigger for a period of time that ends either before or after the user specified time range.

Examples

`TRIGGER:DELAY:SEQUENCE:RANGE:WHEN INRANGE`
specifies that the trigger inputs must meet the Inrange pattern and time definitions for trigger delay.

`TRIGGER:DELAY:SEQUENCE:RANGE:WHEN?`
might return `OUTRANGE`, indicating the trigger inputs are configured for the Outrange pattern and time definitions for trigger delay.

TRIGger:DELay:SEQuence:STARt? (Query Only)

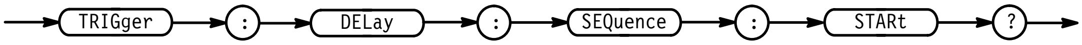
Returns the pattern definition of the start sequence for the delayed trigger.

Group

Trigger

Syntax

```
TRIGger:DELay:SEQuence:STARt?
```



Examples

`TRIGGER:DELAY:SEQUENCE:START?`
might return `:TRIGGER:DELAY:SEQUENCE:START:LOGIC AND;INPUT 65537,1`.

TRIGger:DELay:SEQuence:STARt:AUXiliary

Sets or returns the Boolean requirement (a logic level of level High, Low, or Don’t Care) for the rear panel connector labeled AUX TRIGGER INPUT for the delayed sequence trigger. The Boolean requirements of all trigger inputs define the logic start pattern recognized by the logic scope.

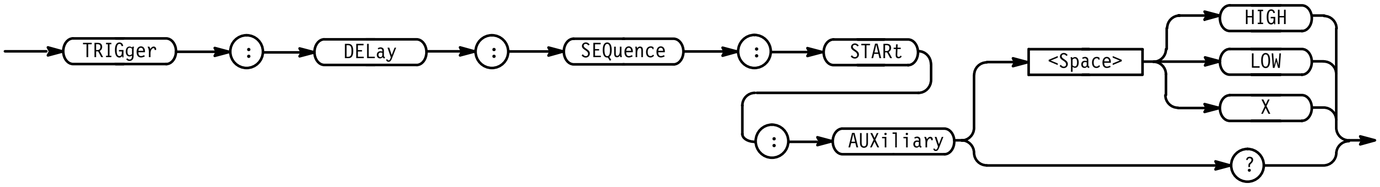
Group

Trigger

Syntax

```
TRIGger:DELay:SEQuence:STARt:AUXiliary { HIGH | LOW | X }
```

```
TRIGger:DELay:SEQuence:STARt:AUXiliary?
```



Arguments

HIGH specifies that the Aux Trigger Input must be High relative to the threshold level.

LOW specifies that the Aux Trigger Input must be Low relative to the threshold level.

<X> specifies that the Aux Trigger Input can be either High or Low relative to the threshold level.

Examples

```
TRIGGER:DELAY:SEQUENCE:START:AUXILIARY LOW
```

specifies that the Aux Trigger Input must be Low to meet the Boolean requirements of the defined pattern.

```
TRIGGER:DELAY:SEQUENCE:START:AUXILIARY?
```

might return X, indicating that the Aux Trigger Input can be either High or Low.

TRIGger:DELay:SEQuence:STARt:CH<x>

Sets or queries the Boolean requirement (a logic level of level High, Low, or Don’t Care) for the selected channel for the delayed sequence trigger. The Boolean requirements set for all the trigger inputs define the logic start pattern recognized by the logic scope.

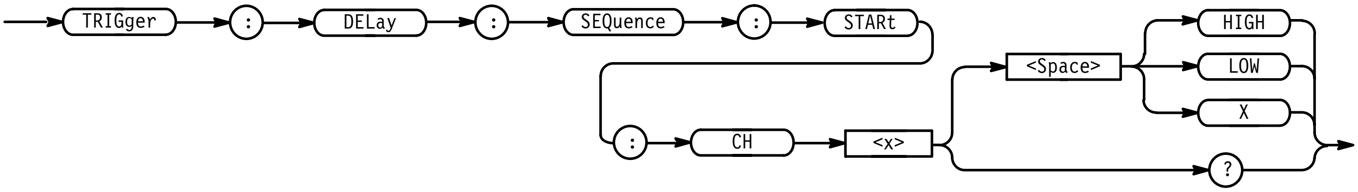
Group

Trigger

Syntax

```
TRIGger:DELay:SEQuence:STARt:CH<x> { HIGH | LOW | X }

TRIGger:DELay:SEQuence:STARt:CH<x>?
```



Arguments

HIGH specifies that the selected channel must be High relative to the threshold level.

LOW specifies that the selected channel must be Low relative to the threshold level.

X specifies that the selected channel can be either High or Low relative to the threshold level.

Examples

TRIGGER:DELAY:SEQUENCE:START:CH14 HIGH
specifies that channel 14 must be High to meet the Boolean requirements of the defined start pattern for the delayed sequence trigger.

TRIGGER:DELAY:SEQUENCE:START:CH11
might return LOW, indicating the delayed sequence trigger logic levels for channel 11.

TRIGger:DELay:SEQuence:STARt:INPut

Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

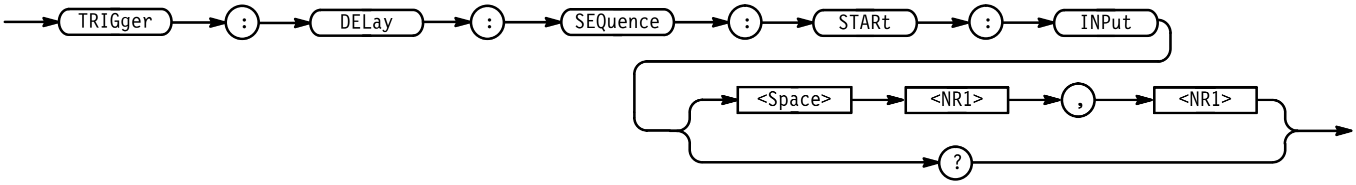
Group

Trigger

Syntax

```
TRIGger:DELay:SEQuence:STARt:INPut <NR1>,<NR1> NOTE: 1-high,0-low
```

```
TRIGger:DELay:SEQuence:STARt:INPut?
```



Arguments

<NR1> is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

<NR1> is a value from 0 to 131071. The “Don’t Care” values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted

as a 17-bit binary number. Bits 1 through 16 are used to set the “Don’t Care” status of channels 1 through 16. Bit 17 sets the “Don’t Care” of the Auxiliary Input. A bit value of zero will set the Boolean value to “Don’t Care” for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples

`TRIGGER:DELAY:SEQUENCE:START:INPUT 65535, 65535`
sets all 16 channels to trigger on High input values and sets the Auxiliary Input to “Don’t Care”.

`TRIGGER:DELAY:SEQUENCE:START:INPUT 1, 1`
sets channel 1 High and all other inputs to to “Don’t Care”.

`TRIGGER:DELAY:SEQUENCE:START:INPUT?`
might return `3,3` indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to “Don’t Care”.

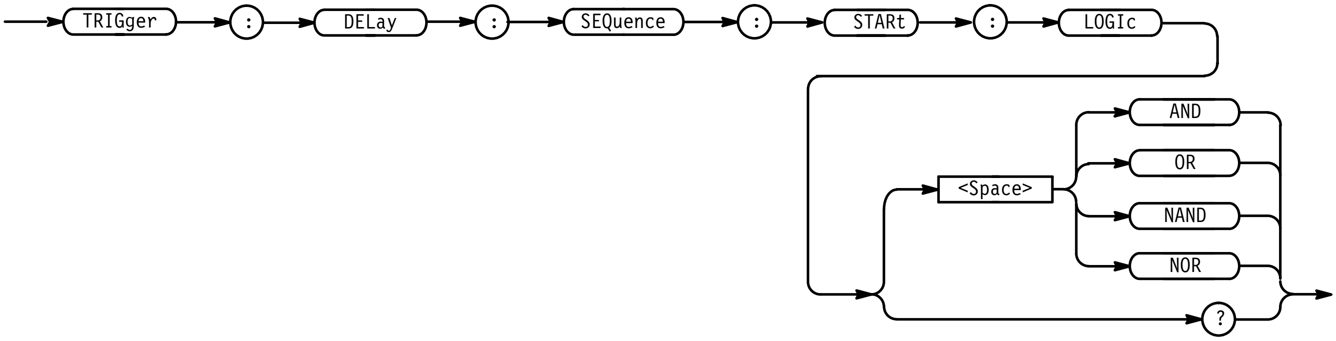
TRIGger:DELay:SEQuence:STARt:LOGIC

Sets or queries the combinatorial logic pattern to apply to the delay sequence trigger pattern.

Group Trigger

Syntax

```
TRIGger:DELay:SEQuence:STARt:LOGIC { AND | OR | NAND | NOR }
```



Arguments

`AND` applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

`OR` applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

NAND applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

NOR applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

```
TRIGGER:DELAY:SEQUENCE:START:LOGIC AND
```

defines AND logic as the delay sequence trigger pattern for the input channels.

```
TRIGGER:DELAY:SEQUENCE:START:LOGIC?
```

might return NOR, indicating NOR logic is the delay sequence trigger pattern for the input channels.

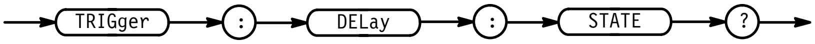
TRIGger:DELay:STATE? (Query Only)

Returns all delayed state trigger parameters.

Group Trigger

Syntax

```
TRIGger:DELay:STATE?
```



Examples

```
TRIGGER:DELAY:STATE?
```

might return :TRIGGER:DELAY:STATE:INPUT 65537,1;LOGIC AND;CLOCK CH1;SLOPE RISE.

TRIGger:DELay:STATE:AUXiliary

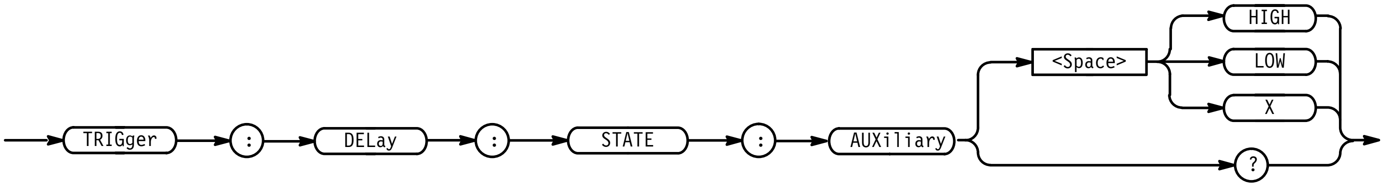
Sets or queries the Boolean requirement (logic levels of High, Low, or Don’t Care) for the rear panel connector labeled AUX TRIGGER INPUT. The Boolean requirements set for all the trigger inputs define the logic pattern that triggers the logic scope.

Group Trigger

Syntax

```
TRIGger:DELay:STATE:AUXiliary { HIGH | LOW | X }
```

TRIGger:DELay:STATE:AUXiliary?



Arguments

HIGH specifies that the Aux Trigger Input must be High relative to the threshold level.

LOW specifies that the Aux Trigger Input must be Low relative to the threshold level.

X specifies that the Aux Trigger Input can be either High or Low relative to the threshold level.

Examples

TRIGGER:DELAY:STATE:AUXILIARY LOW
specifies that the Aux Trigger Input must be Low to meet the Boolean requirements of the currently defined pattern for triggering the logic scope.

TRIGGER:DELAY:STATE:AUXILIARY?
might return X, indicating the Aux Trigger Input logic pattern may be either High or Low to trigger the logic scope.

TRIGger:DELay:STATE:CH<x>

Sets or queries the Boolean requirement (logic levels of High, Low, or Don’t Care) for the selected channel. The Boolean requirements set for all the trigger inputs define the logic pattern that triggers the logic scope.

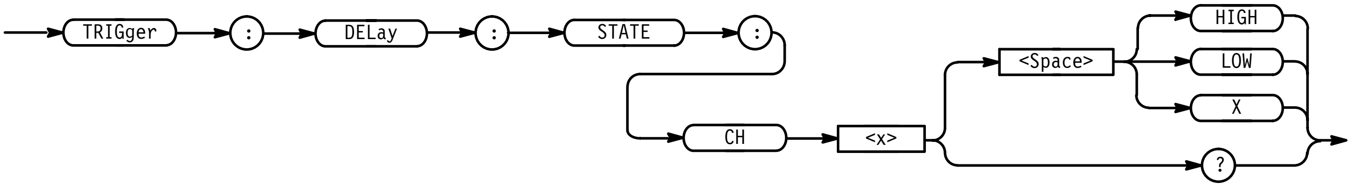
Group

Trigger

Syntax

TRIGger:DELay:STATE:CH<x> { HIGH | LOW | X } NOTE: defines CH<x> pattern bit to High, Low, or Don’t Care

TRIGger:DELay:STATE:CH<x>?



Arguments `HIGH` specifies that the selected channel must be High relative to the threshold level as part of the logic pattern definition that triggers the logic scope.

`LOW` specifies that the selected channel must be Low relative to the threshold level.

`X` specifies that the selected channel can be either High or Low relative to the threshold level.

Examples `TRIGGER:DELAY:STATE:CH14 HIGH`
specifies that channel 14 must be High to meet the Boolean requirements of the defined pattern for the delayed state trigger.

`TRIGGER:DELAY:STATE:CH11?`
might return `X`, indicating channel 11 may be either High or Low (relative to the threshold level) to meet the defined pattern for the delayed state trigger.

TRIGger:DELay:STATE:CLOCk

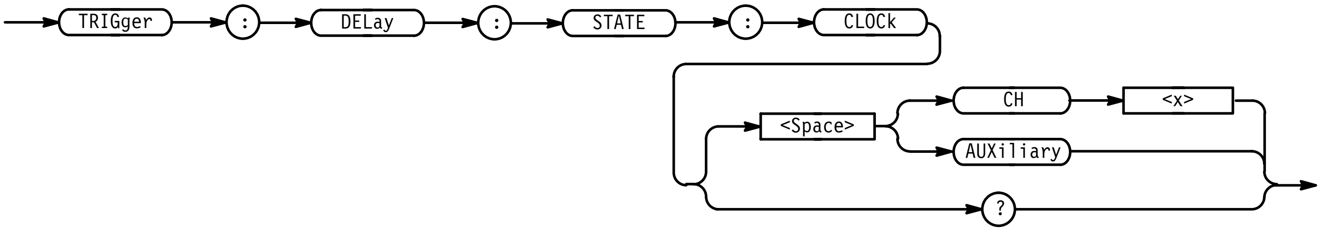
Sets or queries the channel to be used as the clock source for the delayed state trigger.

Group Trigger

Syntax

```
TRIGger:DELay:STATE:CLOCk { CH<x> | AUXiliary }
```

```
TRIGger:DELay:STATE:CLOCk?
```



Arguments `CH<x>` specifies one of the 16 input channels as the clock source.

`AUXiliary` specifies the rear panel connector labeled AUX TRIGGER INPUT as the clock source.

Examples

`TRIGGER:DELAY:STATE:CLOCK AUXILIARY`
designates the rear panel connector AUX TRIGGER INPUT as the clock source.

`TRIGGER:DELAY:STATE:CLOCK?`
might return `CH1` as the clock source for the delayed state trigger.

TRIGger:DELay:STATE:INPut

Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

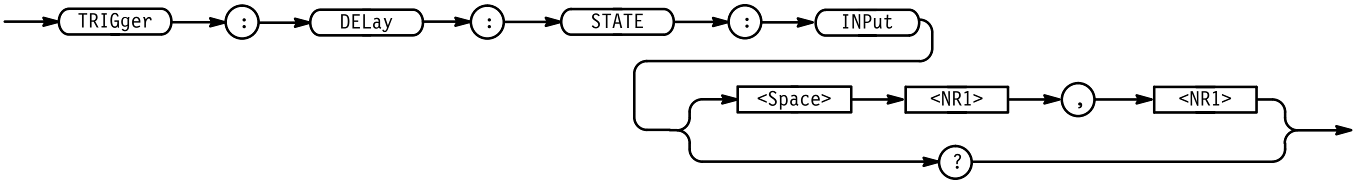
Group

Trigger

Syntax

```
TRIGger:DELay:STATE:INPut <NR1>,<NR1> NOTE: 1-high,0-low
```

```
TRIGger:DELay:STATE:INPut?
```



Arguments

`<NR1>` is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

`<NR1>` is a value from 0 to 131071. The “Don’t Care” values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the “Don’t Care” status of channels 1 through 16. Bit 17 sets the “Don’t Care” of the Auxiliary Input. A bit value of zero will set the Boolean value to “Don’t Care” for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples

`TRIGGER:DELAY:STATE:INPUT 65535, 65535`
sets all 16 channels to trigger on High input values and sets the Auxiliary Input to “Don’t Care”.

`TRIGGER:DELAY:STATE:INPUT 1, 1`
sets channel 1 High and all other inputs to to “Don’t Care”.

`TRIGGER:DELAY:STATE:INPUT?`
might return `3,3` indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to “Don’t Care”.

TRIGger:DELay:STATE:LOGIC

Sets or queries the logic function applied to the delay state trigger inputs.

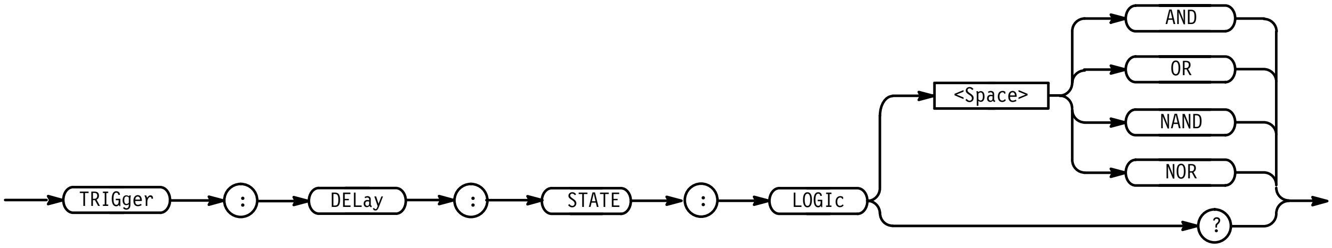
Group

Trigger

Syntax

```
TRIGger:DELay:STATE:LOGIC { AND | OR | NAND | NOR }
```

```
TRIGger:DELay:STATE:LOGIC?
```



Arguments

`AND` applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

`OR` applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

`NAND` applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

`NOR` applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

`TRIGGER:DELAY:STATE:LOGIC AND`
applies AND logic to the delay state trigger input channels.

`TRIGGER:DELAY:STATE:LOGIC?`
might return `OR`, indicating the logic applied to the delay state trigger inputs.

TRIGger:DELay:STATE:SLOpe

Sets or queries the transition edge (rising or falling) that triggers logic scope.

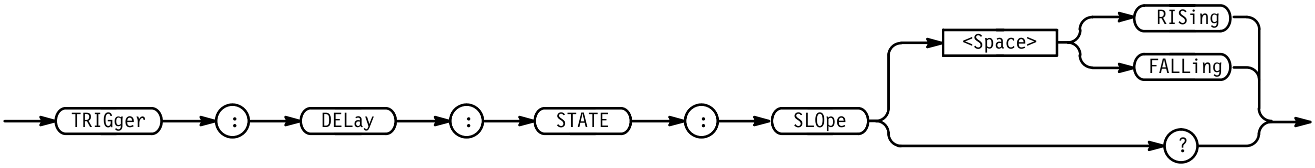
Group

Trigger

Syntax

```
TRIGger:DELay:STATE:SLOpe { RISing | FALLing }
```

```
TRIGger:DELay:STATE:SLOpe?
```



Arguments

`RISing` instructs the logic scope to trigger on the rising transition (positive) edge of a signal.

`FALLing` instructs the logic scope to trigger on the falling transition (negative) edge of a signal.

Examples

`TRIGGER:DELAY:STATE:SLOPE RISING`
instructs the logic scope to trigger on the rising edge of the input signal.

`TRIGGER:DELAY:STATE:SLOPE?`
might return `FALLING`, indicating the logic scope triggers on the falling edge of the input signal.

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.

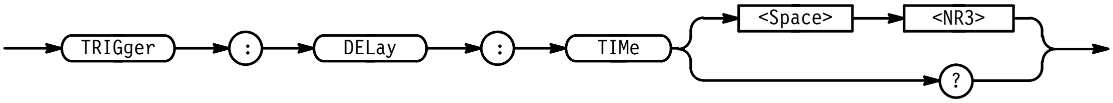
Group Trigger

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 horizontal delay time to 4 μs.

```
TRIGGER:DELAY:TIME?
```

might return 500.0E-3, indicating the horizontal delay is set to 0.5 seconds.

TRIGger:DELay:TYPe

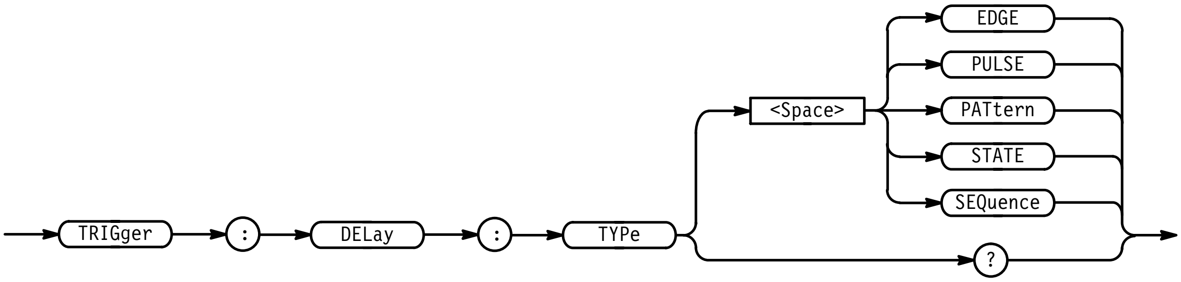
Sets or queries the type of delay trigger. Using this command is equivalent to setting Type in the Delayed Trigger menu.

Group Trigger

Syntax

```
TRIGger:DELay:TYPe { EDGE | PULSE | PATtern | STATE | SEQuence }
```

```
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 and is controlled by the TRIGger:DELay:EDGE commands.

`PULSE` specifies that a trigger occurs when a pulse is found meeting specified conditions related to its width.

`PATtern` specifies that a trigger occurs when a specified pattern is found at the logic scope input channels meeting specified Boolean and time qualifications.

`STATE` specifies that a trigger occurs when a specified pattern is found at the logic scope input channels *at the time the clock input changes state.* The pattern must meet specified Boolean conditions.

Examples `TRIGGER:DELAY:TYPE EDGE`
specifies the logic scope delay trigger type to be `EDGE`.

`TRIGGER:DELAY:TYPE?`
might return `PULSE`, indicating the delay trigger mode.

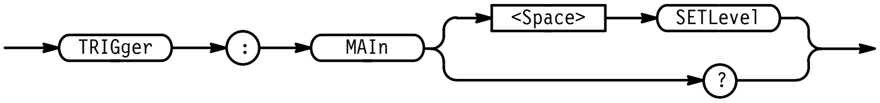
TRIGger:MAIn

Sets the main trigger level to 50% of the trigger signal or queries 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 minimum and maximum amplitudes of the trigger source input. Using this argument is equivalent to pressing the front panel SET LEVel TO 50% button.

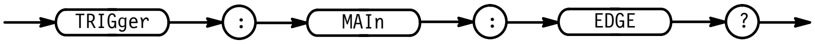
Examples `TRIGGER:MAIN SETLEVEL`
sets the main trigger level to 50% of the minimum and maximum values.

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`

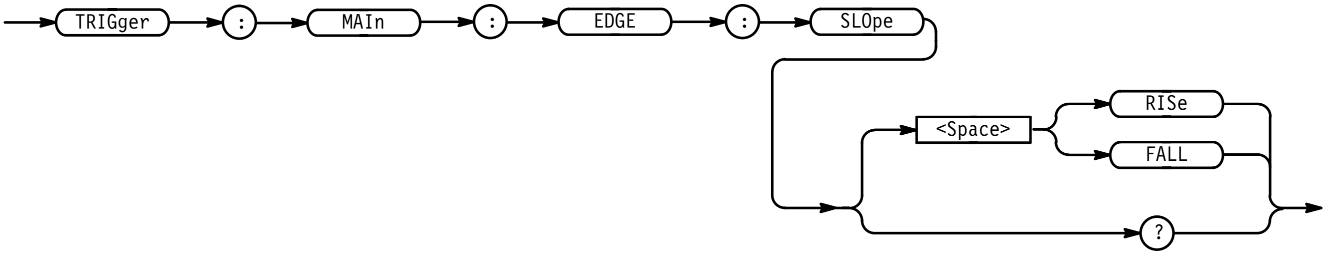
TRIGger:MAIn:EDGE:SLOpe

Selects either the rising or falling transition edge for the main edge trigger. Using this command is equivalent to setting Slope in the Trigger menu.

Group Trigger

Syntax `TRIGger:MAIn:EDGE:SLOpe { FALL | RISe }`

`TRIGger:MAIn:EDGE:SLOpe?`



Arguments

`RISe` instructs the logic scope to trigger on the rising transition (positive) edge of a signal.

`FALL` instructs the logic scope to trigger on the falling transition (negative) edge of a signal.

Examples

`TRIGGER:MAIN:EDGE:SLOPE RISE`
sets the main edge trigger to occur on the rising slope.

`TRIGGER:MAIN:EDGE:SLOPE?`
might return `FALL`, indicating the logic scope will trigger on the falling edge a signal.

TRIGger:MAIn:EDGE:SOUrce

Sets or queries the source for the main edge trigger. This command is equivalent to selecting the main trigger source in the Main Edge Source side menu.

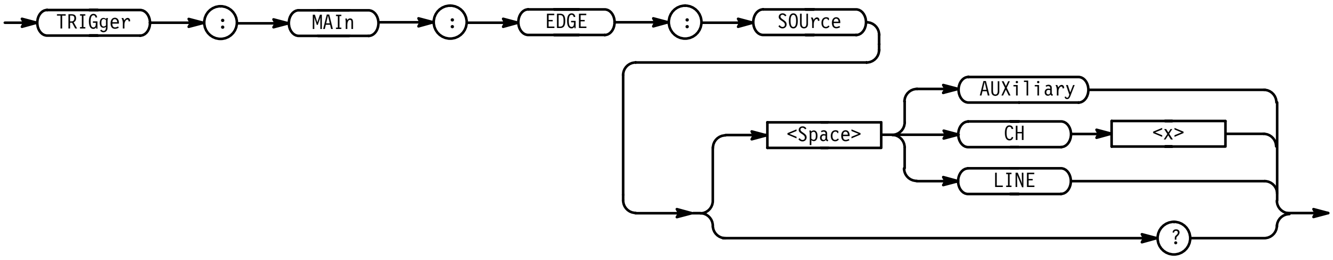
Group

Trigger

Syntax

```
TRIGger:MAIn:EDGE:SOUrce { AUXiliary | CH<x> | LINE }
```

```
TRIGger:MAIn:EDGE:SOUrce?
```



Arguments

`AUXiliary` specifies the rear panel connector labeled AUX TRIGGER INPUT as the main trigger source.

CH<x> specifies one of the 16 input channels as the main trigger source.

LINE specifies the AC line as the main trigger source.

Examples

```
TRIGGER:MAIN:EDGE:SOURCE LINE
```

specifies the AC line as the main edge trigger source.

```
TRIGGER:MAIN:EDGE:SOURCE?
```

might return CH12 for the main edge trigger source.

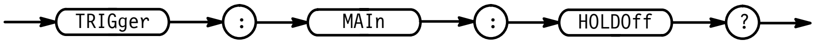
TRIGger:MAIn:HOLDOff? (Query Only)

Returns the main trigger holdoff value.

Group Trigger

Syntax

```
TRIGger:MAIn:HOLDOff?
```



Returns <NR1> from 0 to 100.

Examples

```
TRIGGER:MAIN:HOLDOFF?
```

might return :TRIGGER:MAIN:HOLDOFF:VALUE 0.

TRIGger:MAIn:HOLDOff:VALue

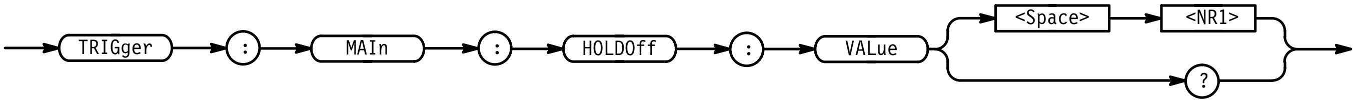
Sets or queries the main trigger holdoff value. Using this command is equivalent to setting Holdoff in the Mode & Holdoff side menu.

Group Trigger

Syntax

```
TRIGger:MAIn:HOLDOff:VALue <NR1>

TRIGger:MAIn:HOLDOff:VALue?
```



Arguments <NR1> is from 0 to 100, and is a percentage of the holdoff range.

Examples `TRIGGER:MAIN:HOLDOFF:VALUE 10`
sets the holdoff value to 10% of the holdoff range.

`TRIGGER:MAIN:HOLDOFF:VALUE?`
might return 0, indicating the main trigger holdoff value is 0%.

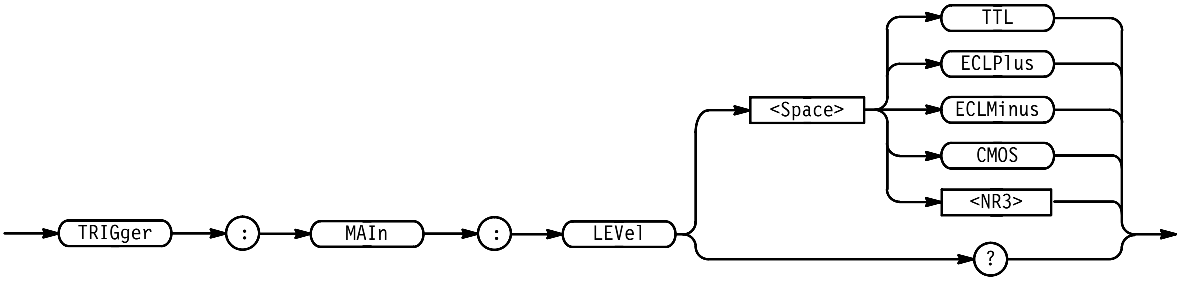
TRIGger:MAIn:LEVel

Set or queries the main trigger level. This command is equivalent to adjusting the front panel TRIGGER THRESHOLD knob.

Group Trigger

Syntax `TRIGger:MAIn:LEVel { TTL | ECLPlus | ECLMinus | CMOS | <NR3> }`

`TRIGger:MAIn:LEVel?`



Arguments `TTL` specifies a preset TTL level of 1.4 V.

`ECLPlus` specifies a preset level of 3.7 V.

`ECLminus` specifies a preset ECL level of –1.3 V.

`CMOS` specifies a preset level of 2.5 V.

<NR3> specify a new threshold in volts.

Examples `TRIGGER:MAIN:LEVEL CMOS`
sets the main trigger level to a preset level of 2.5 V.

`TRIGGER:MAIN:LEVEL?`
might return `1.4E+0`, indicating that the main edge trigger is set to 1.4 V.

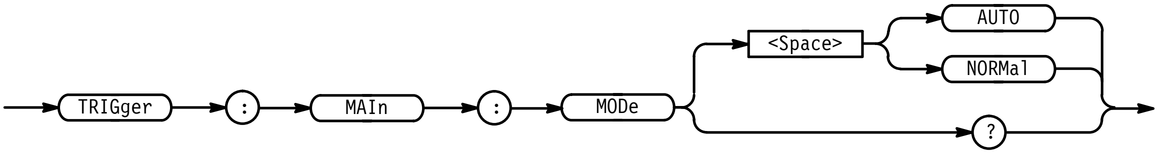
TRIGger:MAIn:MODe

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.

`NORMal` waits for a valid trigger event.

Examples `TRIGGER:MAIN:MODE AUTO`
specifies an automatically generated trigger event.

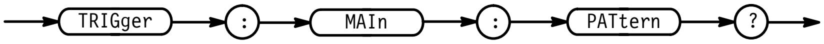
`TRIGGER:MAIN:MODE?`
might return `NORMal`, indicating the logic scope is awaiting a valid trigger event.

TRIGger:MAIn:PATtern (Query Only)

Returns all main pattern trigger parameters.

Group Trigger

Syntax `TRIGger:MAIn:PATtern?`



Examples

`TRIGGER:MAIN:PATTERN?`
might return `TRIGGER:MAIN:PATTERN:CLASS RANGE;TRIG WHEN WITHIN;LOGIC AND;` as the current main pattern trigger parameters.

TRIGger:MAIn:PATtern:AUXiliary

Sets or queries the Boolean requirement (a logic level of level of High, Low, or Don't Care) for the rear panel connector labeled AUX TRIGGER INPUT. The Boolean requirements for all inputs define the logic pattern that triggers the logic scope. Using this command is the same as setting the auxiliary bit in the Define Pattern menus found in the Pattern Trigger menu for the main trigger.

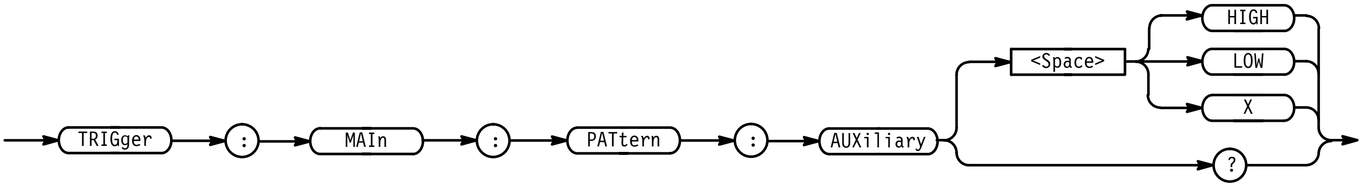
Group

Trigger

Syntax

```
TRIGger:MAIn:PATtern:AUXiliary { HIGH | LOW | X } NOTE: defines
AUXiliary pattern bit to H,L,X
```

```
TRIGger:MAIn:PATtern:AUXiliary?
```



Arguments

`HIGH` specifies that the AUX TRIGGER INPUT must be High relative to the threshold level.

`LOW` specifies that the AUX TRIGGER INPUT must be Low relative to the threshold level.

`X` specifies that the AUX TRIGGER INPUT can be either High or Low relative to the threshold level.

Examples

`TRIGGER:MAIN:PATTERN:AUXILIARY LOW`
specifies that the AUX TRIGGER INPUT must be Low to meet the Boolean requirements of the defined pattern.

`TRIGGER:MAIN:PATTERN:AUXILIARY?`
might return `X`, indicating the AUX TRIGGER INPUT may be either High or Low relative to the to the threshold level.

TRIGger:MAIn:PATtern:CH<x>

Sets or queries the Boolean requirement (a logic level of High, Low, or Don't Care) for the selected channel. The Boolean requirements of all inputs define the logic pattern that triggers the logic scope. This command is equivalent to setting the selected channel's bit in the Define Pattern menus found in the Pattern Trigger menu for the main trigger.

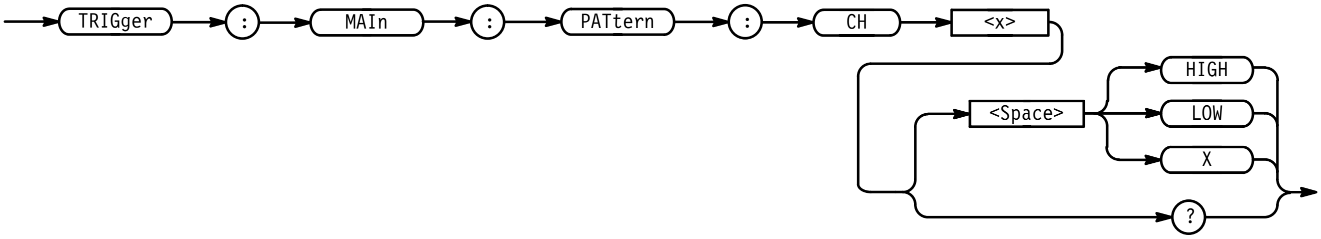
Group

Trigger

Syntax

`TRIGger:MAIn:PATtern:CH<x> { HIGH | LOW | X }` NOTE: defines `CH<x>` pattern bit to High, Low or Don't Care

`TRIGger:MAIn:PATtern:CH<x>?`



Arguments

`HIGH` specifies that the selected channel must be High relative to the threshold level.

`LOW` specifies that the selected channel must be Low relative to the threshold level.

`X` specifies that the selected channel can be either High or Low relative to the threshold level.

Examples

`TRIGGER:MAIN:PATTERN:CH14 HIGH`
specifies that channel 14 must be High to meet the Boolean requirements of the defined pattern for the main trigger.

`TRIGGER:MAIN:PATTERN:CH11`
might return `X`, indicating that channel 11 may be either High or Low relative to the threshold level, to meet the Boolean requirements of the defined pattern for the main trigger.

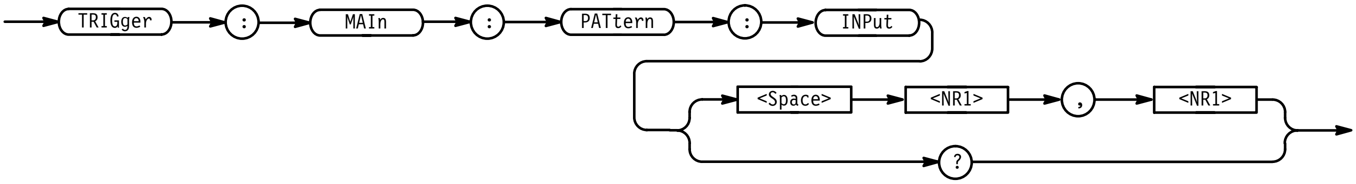
TRIGger:MAIn:PATtern:INPut

Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

Group Trigger

Syntax `TRIGger:MAIn:PATtern:INPut <NR1>,<NR1> NOTE: 1 High, 0 low`

`TRIGger:MAIn:PATtern:INPut?`



Arguments <NR1> is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

<NR1> is a value from 0 to 131071. The “Don’t Care” values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the “Don’t Care” status of channels 1 through 16. Bit 17 sets the “Don’t Care” of the Auxiliary Input. A bit value of zero will set the Boolean value to “Don’t Care” for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples `TRIGGER:MAIN:PATTERN:INPUT 65535, 65535`

sets all 16 channels to trigger on High input values and sets the Auxiliary Input to “Don’t Care”.

`TRIGGER:MAIN:PATTERN:INPUT 1, 1`

sets channel 1 High and all other inputs to to “Don’t Care”.

`TRIGGER:MAIN:PATTERN:INPUT?`

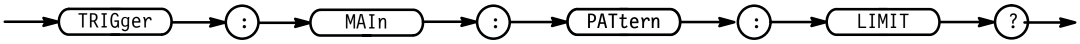
might return `3,3` indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to “Don’t Care”.

TRIGger:MAIn:PATtern:LIMIT? (Query Only)

Returns the main time-qualified trigger pattern parameters.

Group Trigger

Syntax `TRIGger:MAIn:PATtern:LIMIT?`



Examples `TRIGGER:MAIN:PATTERN:LIMIT?`
might return `:TRIGGER:MAIN:PATTERN:LIMIT:WHEN MORETHAN;WIDTH 2.0E-9`.

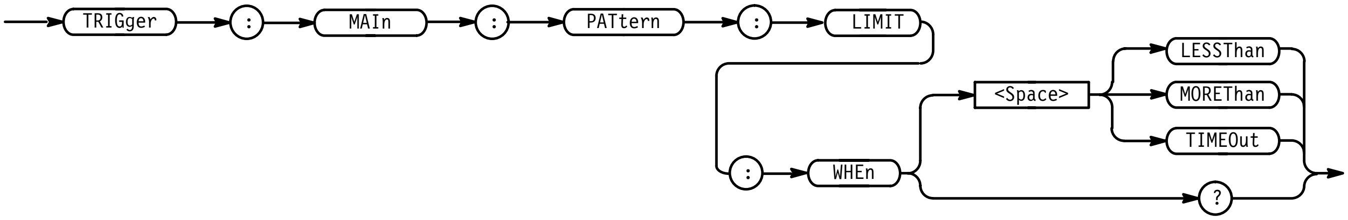
TRIGger:MAIn:PATtern:LIMIT:WHEn

Sets or queries the definition of when the Boolean requirements must cease to meet the conditions of the defined pattern to trigger the logic scope.

Group Trigger

Syntax `TRIGger:MAIn:PATtern:LIMIT:WHEn { LESSThan | MOREThan | TIMEOut }`

`TRIGger:MAIn:PATtern:LIMIT:WHEn?`



Arguments `LESSThan` instructs the logic scope to trigger if the trigger inputs cease to meet the Boolean requirements before the specified time.

`MOREThan` instructs the logic scope to trigger if the trigger inputs cease to meet the Boolean requirements after the specified time.

`TIMEOut` instructs the logic scope to trigger following the specified time lapse, if the trigger inputs still meet the Boolean requirements.

Examples

```
TRIGGER:MAIN:PATTERN:LIMIT:WHEN TIMEOUT
```

instructs the logic scope to trigger, after the specified time lapse, while the trigger inputs still meet the Boolean requirements.

```
TRIGGER:MAIN:PATTERN:LIMIT:WHEN?
```

might return `TIMEOUT`., indicating the logic scope will trigger following the specified time lapse, if the trigger inputs still meet the Boolean requirements.

TRIGger:MAIn:PATtern:LIMIT:WIDth

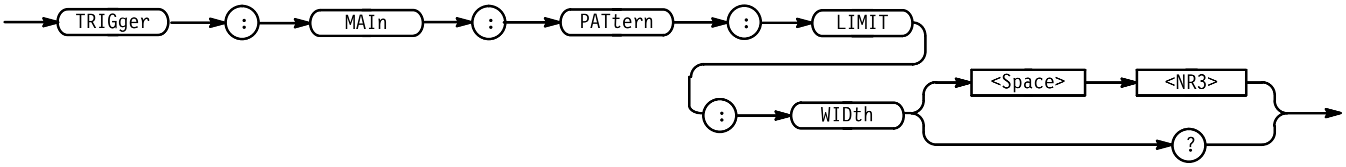
Sets or queries the time limit for the Trigger When definitions of Less Than, More Than, or Time Out. This is equivalent to pressing Time in the main pattern time triggers side menu.

Group Trigger

Syntax

```
TRIGger:MAIn:PATtern:LIMIT:WIDth <NR3>
```

```
TRIGger:MAIn:PATtern:LIMIT:WIDth?
```



Arguments `<NR3>` is the time limit from 2.0 ns to 1.0 second.

Examples

```
TRIGGER:MAIn:PATTERN:LIMIT:WIDTH 1.0E-3
```

sets the time limit width to 1.0 ms.

```
TRIGGER:MAIN:PATTERN:LIMIT:WIDTH?
```

might return `10 ns` as the time limit width.

TRIGger:MAIn:PATtern:LOGIC

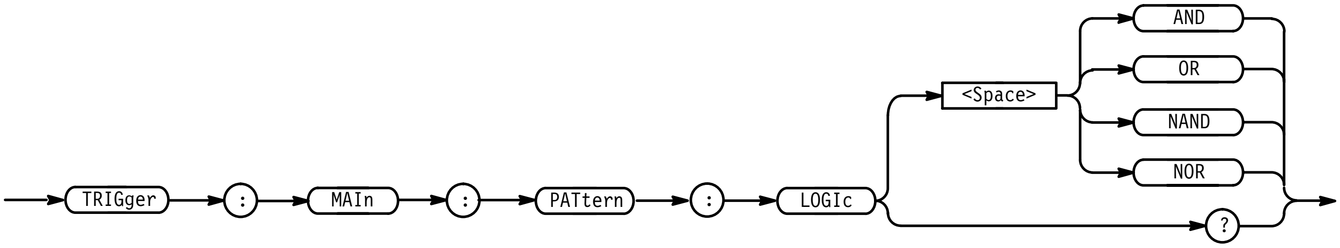
Sets or queries the combinatorial logic pattern to apply to the main trigger pattern.

Group Trigger

Syntax

```
TRIGger:MAIn:PATtern:LOGIC { AND | OR | NAND | NOR }
```

```
TRIGger:MAIn:PATtern:LOGIC?
```



Arguments

AND applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

OR applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

NAND applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

NOR applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

```
TRIGGER:MAIN:PATTERN:LOGIC AND
```
defines AND logic as the input channel trigger pattern for the main trigger.

```
TRIGGER:MAIN:PATTERN:LOGIC?
```
might return OR, indicating that the input channel main trigger pattern is set to OR.

TRIGger:MAIn:PATtern:QUAlifier

Sets or queries the Time Qualifier used to define when the logic scope triggers. Using this command is equivalent to selecting the time qualifier Limit or Range for the main pattern trigger.

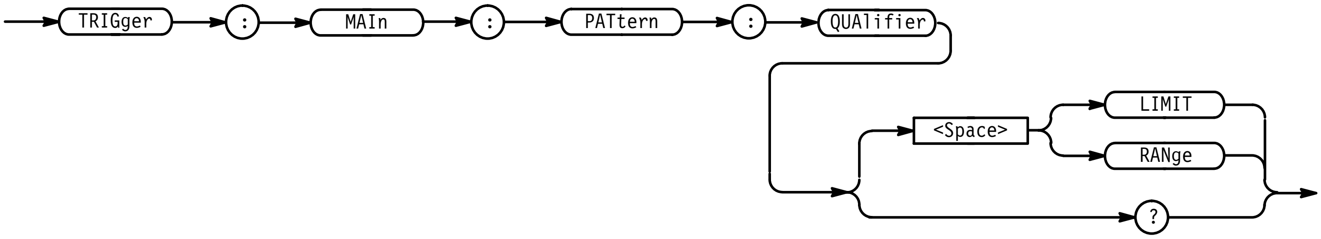
Group

Trigger

Syntax

```
TRIGger:MAIn:PATtern:QUAlifier { LIMIT | RANge }
```

```
TRIGger:MAIn:PATtern:QUAlifier?
```



Arguments `LIMIT` defines a maximum time limit for the main trigger event to occur.

`RANge` defines a range of time for the main trigger event to occur.

Examples `TRIGGER:MAIN:PATTERN:QUALIFIER RANGE`
specifies that the main trigger event must occur either within or outside a specified amount of time.

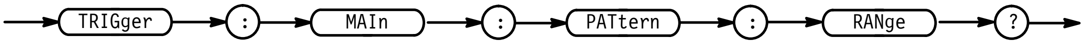
`TRIGGER:MAIN:PATTERN:QUALIFIER?`
might return `LIMIT`, indicating there is a maximum time limit for the main trigger event.

TRIGger:MAIn:PATtern:RANge? (Query Only)

Returns the main pattern trigger range parameters.

Group Trigger

Syntax `TRIGger:MAIn:PATtern:RANge?`



Examples `TRIGGER:MAIN:PATTERN:RANGE?`
might return `:TRIGGER:MAIN:PATTERN:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E-9;LOWLIMIT 2.0E-9`.

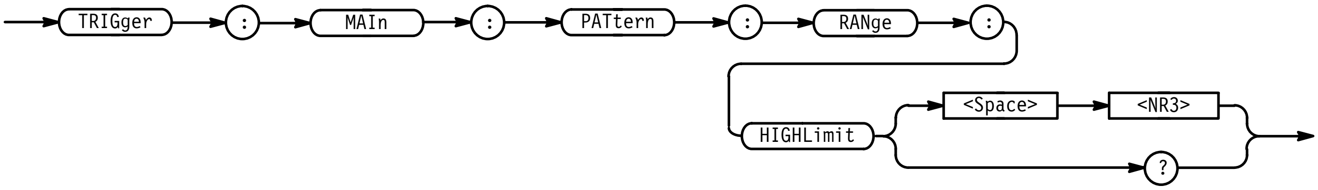
TRIGger:MAIn:PATtern:RANge:HIGHLimit

Sets or queries the value for the upper time limit of the range for the range-qualified main trigger. Using this command is equivalent to selecting upper limit for main pattern triggers with range selected as the time qualifier.

Group Trigger

Syntax `TRIGger:MAIn:PATtern:RANge:HIGHLimit <NR3>`

`TRIGger:MAIn:PATtern:RANge:HIGHLimit?`



Arguments <NR3> is the upper limit of the range in seconds.

Examples `TRIGGER:MAIN:PATTERN:RANGE:HIGHLIMIT 6.5E-9`
sets the upper limit of the range to 6.5 ns.

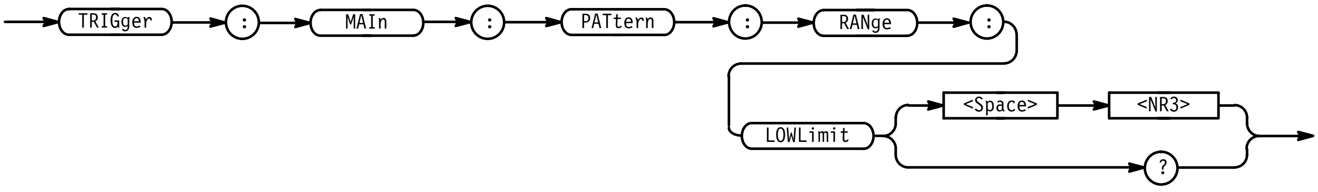
TRIGger:MAIn:PATtern:RANge:LOWLimit

Sets or queries the value for the lower time limit of the range for the range-qualified main trigger. This command is equivalent to selecting Lower Limit for main pattern triggers with range selected as the time qualifier.

Group Trigger

Syntax `TRIGger:MAIn:PATtern:RANge:LOWLimit <NR3>`

`TRIGger:MAIn:PATtern:RANge:LOWLimit?`



Arguments <NR3> is the lower limit of the range in seconds.

Examples `TRIGGER:MAIN:PATTERN:RANGE:LOWLIMIT 2.5E-9`
sets the upper limit of the range to 2.5 ns.

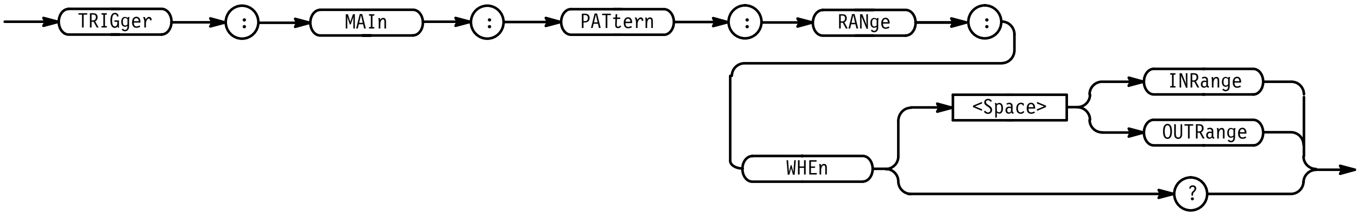
TRIGger:MAIn:PATtern:RANge:WHEn

Sets or queries when the channel inputs must cease to meet the Boolean requirements of the main pattern definition to trigger the logic scope.

Group Trigger

Syntax `TRIGger:MAIn:PATtern:RANge:WHEn { INRange | OUTRange }`

`TRIGger:MAIn:PATtern:RANge:WHEn?`



Arguments `INRange` specifies that the trigger inputs must meet the Boolean requirements of the main trigger pattern definition for a period of time that ends within the user specified time range.

`OUTRange` specifies that the trigger inputs must meet the Boolean requirements of the main trigger pattern definition for a period of time that ends either before or after the user specified time range.

Examples `TRIGGER:MAIN:PATTERN:RANGE:WHEN INRANGE`

specifies that the channel inputs must cease to meet the Boolean requirements of the main trigger pattern definition within the time range defined.

`TRIGGER:MAIN:PATTERN:RANGE:WHEN?`

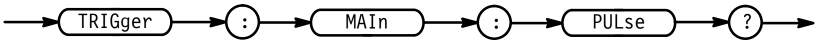
might return `OUTRANGE`, indicating the trigger inputs must meet the main trigger pattern definition for a period of time that ends either before or after the user specified time range.

TRIGger:MAIn:PULse? (Query 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;WHEN LESSTHAN;POLARITY POSITIVE;:TRIGGER: MAIN:PULSE:WIDTH:LOWLIMIT 2.0E-9;HIGHLIMIT 2.0E-9;WHEN WITHIN;POLARITY POSITIVE.`

TRIGger:MAIn:PULse:CLAss

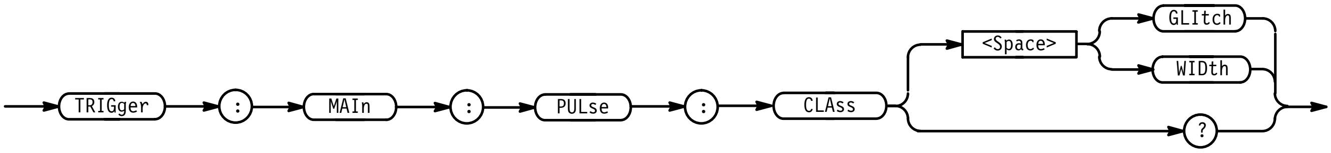
Sets or queries the class used for main pulse triggering. This command is equivalent to selecting Glitch or Width from the main pulse trigger class menu.

Group Trigger

Syntax

```
TRIGger:MAIn:PULse:CLAss { GLItch | 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.

`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:LOW-Limit 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 the Width class for the main pulse trigger.

```
TRIGGER:MAIN:PULSE:CLASS?
```

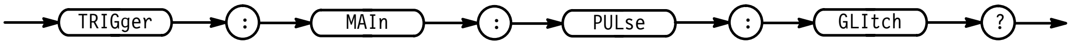
might return `GLITCH`, specifying main pulse triggering meeting the polarity and width criteria set by the TRIGger:MAIn:PULse:GLItch:POLarity and WIDth commands.

TRIGger:MAIn:PULse:GLItch? (Query Only)

Returns the main trigger parameters for pulse glitch triggers.

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;POLARITY POSITIVE`.

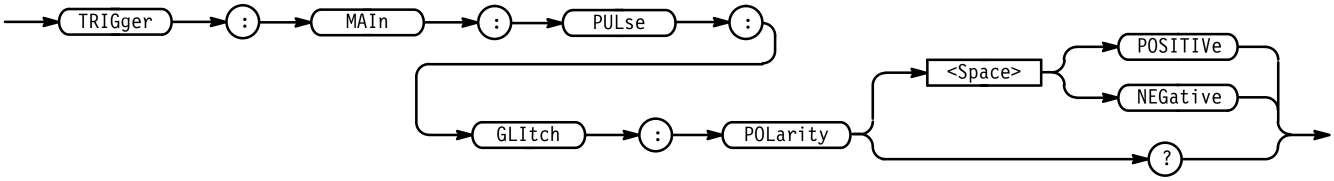
TRIGger:MAIn:PULse:GLItch:POLarity

Sets or queries the transition direction of the glitch that triggers the logic scope. This command is equivalent to setting polarity for main pulse triggers.

Group Trigger

Syntax `TRIGger:MAIn:PULse:GLItch:POLarity { POSITIVe | NEGative }`

`TRIGger:MAIn:PULse:GLItch:POLarity?`



Arguments `POSITIVe` specifies the narrower glitch portion of the waveform be the positive-going portion of the waveform cycle.

`NEGative` specifies the narrower glitch portion of the waveform be the negative-going portion of the waveform cycle.

Examples `TRIGGER:MAIN:PULSE:GLITCH:POLARITY POSITIVE`
specifies that the polarity of the narrower glitch portion of the waveform is positive.

`TRIGGER:MAIN:PULSE:GLITCH:POLARITY?`
might return `NEGATIVE`, specifying that the polarity of the narrower glitch portion of the waveform is negative.

TRIGger:MAIn:PULse:GLItch:WHEN

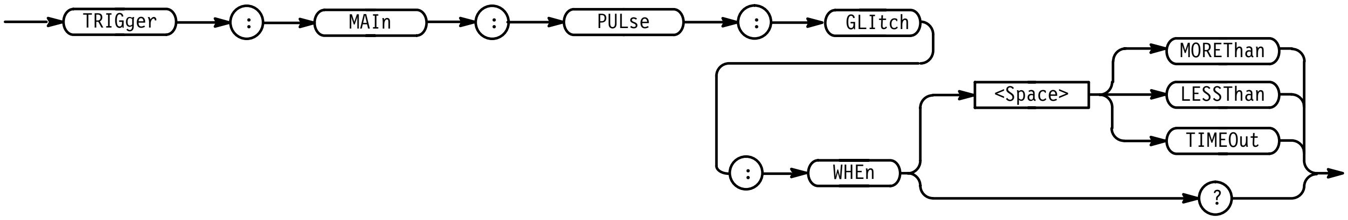
Sets or queries the main trigger pulse glitch filter. Glitches are either filtered out or accepted using a greater-than or less-than format.

Group Trigger

Syntax

```
TRIGger:MAIn:PULse:GLItch:WHEn { MOREThan | LESSThan | TIMEOut }
```

```
TRIGger:MAIn:PULse:GLItch:WHEn?
```



Arguments `MOREThan` specifies that triggering occurs when the width of the glitch is more than the specified time.

`LESSThan` specifies that triggering occurs when the width of the glitch is less than the specified time.

`TIMEOut` specifies that triggering occurs when the glitch does not occur within the specified time. Triggering occurs at the point the specified time elapses.

Examples `TRIGGER:MAIN:PULSE:GLITCH:WHEN TIMEOUT`
specifies that triggering occurs when the glitch does not occur within a specified time.

`TRIGGER:MAIN:PULSE:GLITCH:WHEN?`
might return `LESSTHAN`, indicating that triggering occurs when the width of the glitch is less than the specified time.

TRIGger:MAIn:PULse:GLItch:WIDth

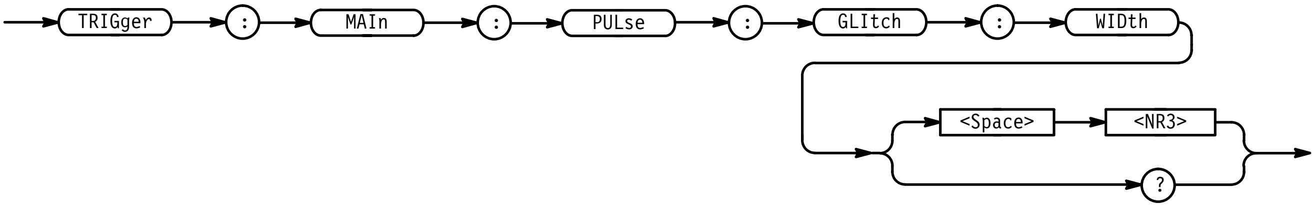
Sets or queries the width for the main pulse glitch trigger. This command is equivalent to selecting Polarity & Width in the Trigger menu.

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:SOUrce

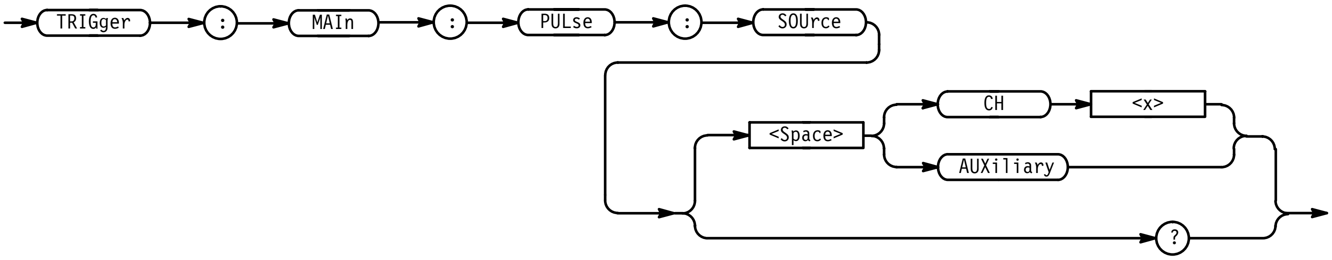
Sets or queries the source for the main pulse trigger. This command is equivalent to selecting the main trigger source in the main pulse Source side menu.

Group Trigger

Syntax

```
TRIGger:MAIn:PULse:SOUrce { CH<x> | AUxiliary }
```

```
TRIGger:MAIn:PULse:SOUrce?
```



Arguments CH<x> specifies one of the 16 input channels as the main pulse trigger source.

AUXiliary specifies the rear panel connector labeled AUX TRIGGER INPUT as the main pulse trigger source.

Examples `TRIGGER:MAIN:PULSE:SOURCE CH2`
selects Channel 2 as the source for the main pulse trigger.

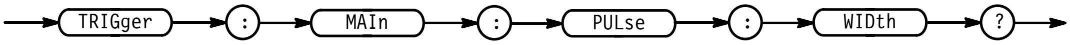
`TRIGGER:MAIN:PULSE:SOURCE?`
might return `AUXILIARY`, indicating that the rear panel connector labeled AUX TRIGGER INPUT is the main pulse trigger source.

TRIGger:MAIn:PULse:WIDth? (Query 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.

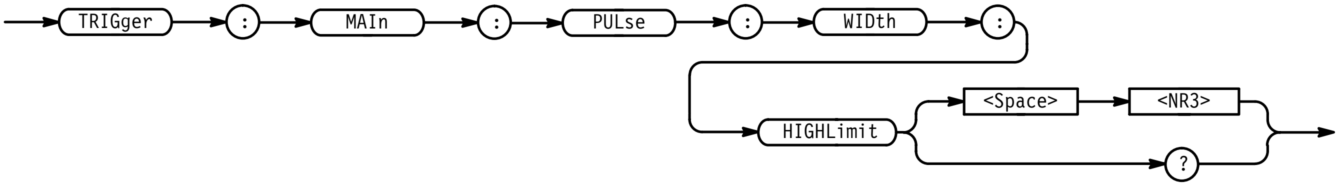
TRIGger:MAIn:PULse:WIDth:HIGHLimit

Sets or queries the upper limit for the main pulse width trigger. Using this command is equivalent to setting the Upper Limit in the Pulse Width Trigger 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.

Examples `TRIGGER:MAIN:PULSE:WIDTH:HIGHLIMIT 2.0E-6`
sets the upper limit for the main pulse width trigger to 2.0 μs.

`TRIGGER:MAIN:PULSE:WIDTH:HIGHLIMIT?`
might return `500E-3`, indicating the upper limit for the main pulse width trigger is 0.5 seconds.

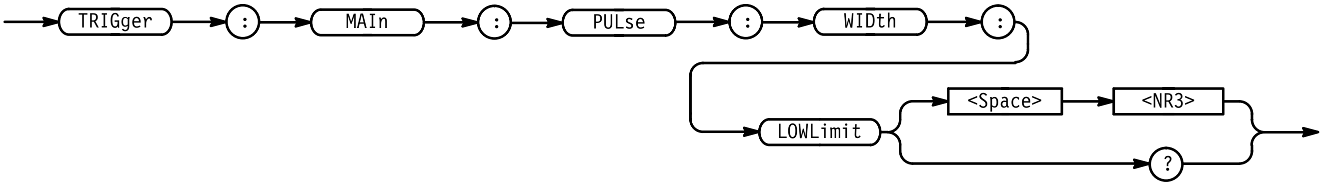
TRIGger:MAIn:PULse:WIDth:LOWLimit

Sets or queries the lower limit for the main pulse width trigger. Using this command is equivalent to setting the Lower Limit in the Pulse Width Trigger 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.

Examples `TRIGGER:MAIN:PULSE:WIDTH:LOWLIMIT 2.0E-6`
sets the lower limit for the main pulse width trigger to 2.0 μs.

`TRIGGER:MAIN:PULSE:WIDTH:LOWLIMIT?`
might return `100E-3`, indicating the lower limit for the main pulse width trigger is 0.1 seconds.

TRIGger:MAIn:PULse:WIDth:POLarity

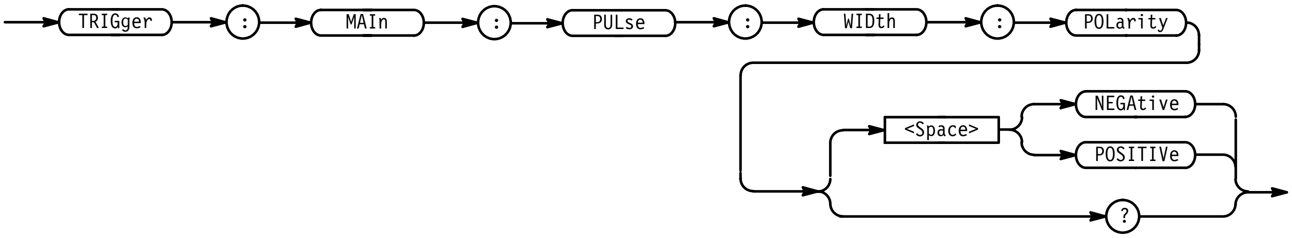
Sets or queries the polarity for the main pulse width trigger. Using this command is equivalent to selecting the polarity in the Pulse Width Polarity side menu.

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.

Examples

```
TRIGGER:MAIN:PULSE:WIDTH:POLARITY NEGATIVE
```

sets the main pulse width trigger polarity to negative.

```
TRIGGER:MAIN:PULSE:WIDTH:POLARITY?
```

might return POSITIVE, indicating the main pulse width trigger polarity is positive.

TRIGger:MAIn:PULse:WIDth:WHEn

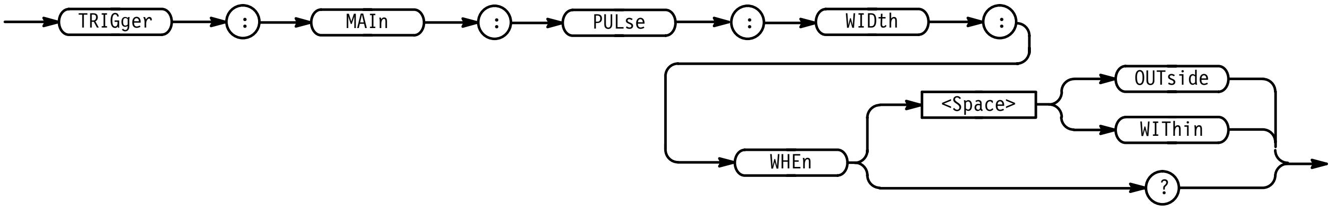
Sets or queries the main pulse width parameters of when the logic scope triggers. The parameters define when the trigger event must end to trigger the logic scope. This command is equivalent to selecting the condition of the Pulse Width Trigger When side menu.

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 WITHIN`

instructs the logic scope to trigger when the duration of the trigger pulse is within the specified high and low limits.

`TRIGGER:MAIN:PULSE:WIDTH:WHEN?`

returns either `OUTSIDE` or `WITHIN`, indicating the conditions for generating a pulse trigger.

TRIGger:MAIn:SEQuence? (Query Only)

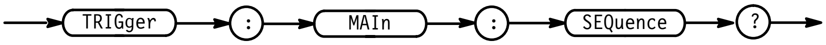
Returns the main sequence trigger parameters.

Group

Trigger

Syntax

`TRIGger:MAIn:SEQuence?`



Examples

`TRIGGER:MAIN:SEQUENCE?`

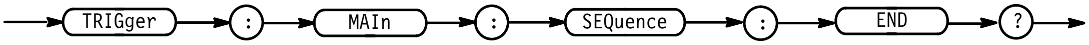
might return `:TRIGGER:MAIN:SEQUENCE:QUALIFIER LIMIT;LIMIT:WHEN MORETHAN;WIDTH 2.0E–9;:TRIGGER:MAIN:SEQUENCE:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E–9;LOWLIMIT 2.0E–9;:TRIGGER:MAIN:SEQUENCE:START:LOGIC AND;INPUT 65537,1;:TRIGGER:MAIN:SEQUENCE:END:LOGIC AND;INPUT 65537,2`.

TRIGger:MAIn:SEQuence:END? (Query Only)

Returns the pattern definition of the end sequence for the main trigger.

Group Trigger

Syntax `TRIGger:MAIn:SEQuence:END?`



Examples `TRIGGER:MAIN:SEQUENCE:END?`
might return `:TRIGGER:MAIN:SEQUENCE:END:LOGIC AND;INPUT 65537,2`.

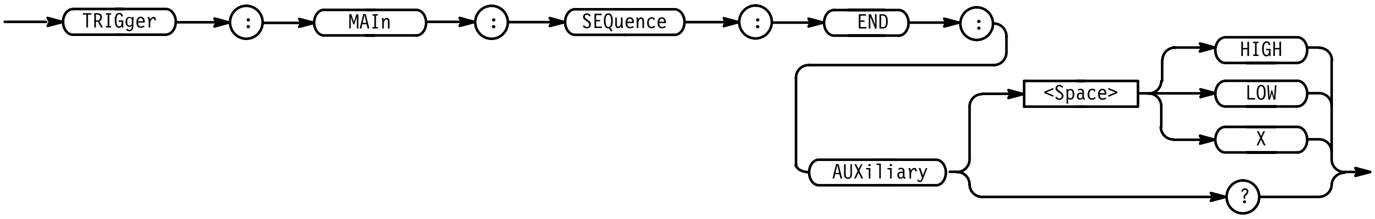
TRIGger:MAIn:SEQuence:END:AUXiliary

Sets or queries the logic level expected at the rear panel connector labeled AUX TRIGGER INPUT as part of the end pattern definition for main sequence trigger.

Group Trigger

Syntax `TRIGger:MAIn:SEQuence:END:AUXiliary { HIGH | LOW | X }`

`TRIGger:MAIn:SEQuence:END:AUXiliary?`



Arguments `HIGH` specifies that the AUX TRIGGER INPUT must be High relative to the threshold level.

`LOW` specifies that the AUX TRIGGER INPUT must be Low relative to the threshold level.

`X` specifies that the AUX TRIGGER INPUT can be either High or Low.

Examples

`TRIGGER:MAIN:SEQUENCE:END:AUXILIARY HIGH`
specifies that the AUX TRIGGER INPUT must be High, relative to the threshold level, as part of the end pattern definition for the main sequence trigger.

`TRIGGER:MAIN:SEQUENCE:END:AUXILIARY?`
might return 1, indicating that the end pattern definition for the main sequence trigger is High.

TRIGger:MAIn:SEQuence:END:CH<x>

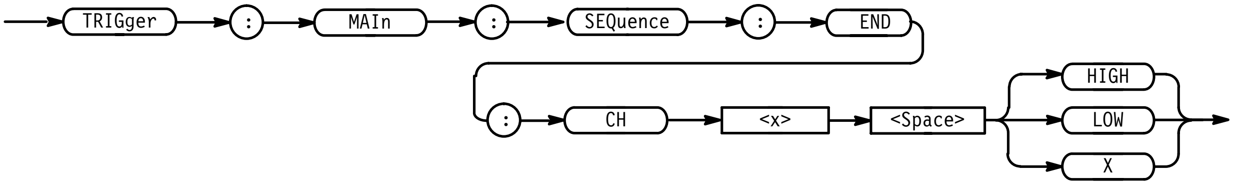
Sets or queries the logic level expected at the channel input defined by Ch<x> as part of the end pattern definition for the main sequence trigger.

Group

Trigger

Syntax

`TRIGger:MAIn:SEQuence:END:CH<x> { HIGH | LOW | X }` NOTE: defines `CH<x>` pattern bit to High, Low or Don't Care



Arguments

`HIGH` specifies that the selected channel must be High relative to the threshold level.

`LOW` specifies that the selected channel must be Low relative to the threshold level.

`X` specifies that the selected channel can be either High or Low relative to the threshold level.

Examples

`TRIGGER:MAIN:PATTERN:CH14 HIGH`
specifies that channel 14 must be High to meet the Boolean requirements of the defined pattern for the main trigger.

`TRIGGER:MAIN:PATTERN:CH11 HIGH?`
might return `LOW`, indicating channel 11 must be Low to meet the Boolean requirements of the defined pattern for the main trigger.

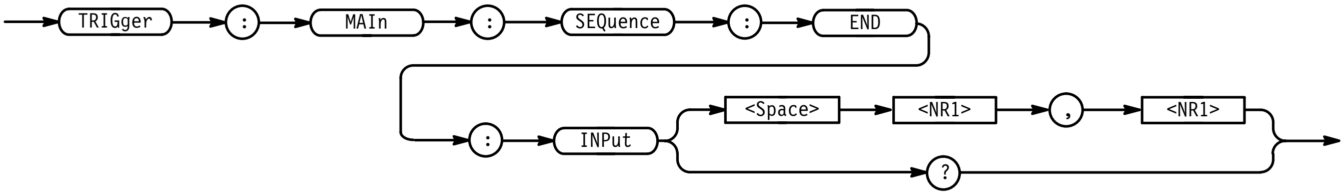
TRIGger:MAIn:SEQuence:END:INPut

Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

Group Trigger

Syntax `TRIGger:MAIn:SEQuence:END:INPut <NR1>,<NR1>` `NOTE: 1 High, 0 low`

`TRIGger:MAIn:SEQuence:END:INPut?`



Arguments `<NR1>` is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

`<NR1>` is a value from 0 to 131071. The “Don’t Care” values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the “Don’t Care” status of channels 1 through 16. Bit 17 sets the “Don’t Care” of the Auxiliary Input. A bit value of zero will set the Boolean value to “Don’t Care” for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples `TRIGGER:MAIN:SEQUENCE:END:INPUT 65535, 65535`
sets all 16 channels to trigger on High input values and sets the Auxiliary Input to “Don’t Care”.

`TRIGGER:MAIN:SEQUENCE:END:INPUT 1, 1`
sets channel 1 High and all other inputs to to “Don’t Care”.

`TRIGGER:MAIN:SEQUENCE:END:INPUT?`
might return `3,3` indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to “Don’t Care”.

TRIGger:MAIn:SEQuence:END:LOGIC

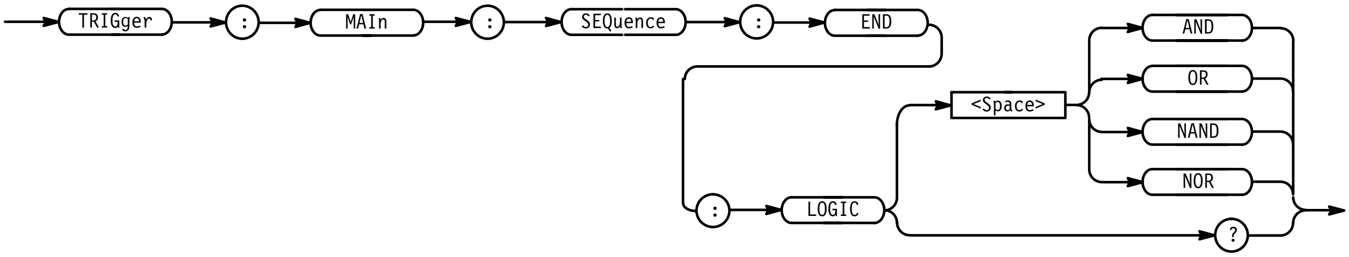
Sets or queries the combinatorial logic pattern applied to the end pattern for the main sequence trigger input.

Group Trigger

Syntax

```
TRIGger:MAIn:SEQuence:END:LOGIC { AND | OR | NAND | NOR }
```

```
TRIGger:MAIn:SEQuence:END:LOGIC?
```



Arguments

AND applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

OR applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

NAND applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

NOR applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

```
TRIGGER:MAIN:SEQUENCE:END:LOGIC AND
```

applies AND logic as the input channel trigger pattern for the main sequence trigger.

```
TRIGGER:MAIN:SEQUENCE:END:LOGIC?
```

might return OR, indicating that the input channel main sequence trigger pattern is set to OR.

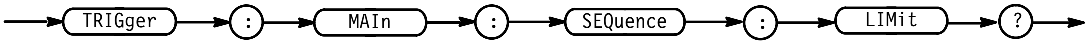
TRIGger:MAIn:SEQuence:LIMit? (Query Only)

Returns the limit parameters of the main sequence trigger. The limit parameters define when the trigger inputs must cease to meet the Boolean requirements to trigger the logic scope.

Group Trigger

Syntax

```
TRIGger:MAIn:SEQuence:LIMit?
```



Examples

```
TRIGGER:MAIN:SEQUENCE:LIMIT?
```

might return `:TRIGGER:MAIN:SEQUENCE:LIMIT:WHEN MORETHAN;WIDTH 2.0E−9`.

TRIGger:MAIn:SEQuence:LIMit:WHEn

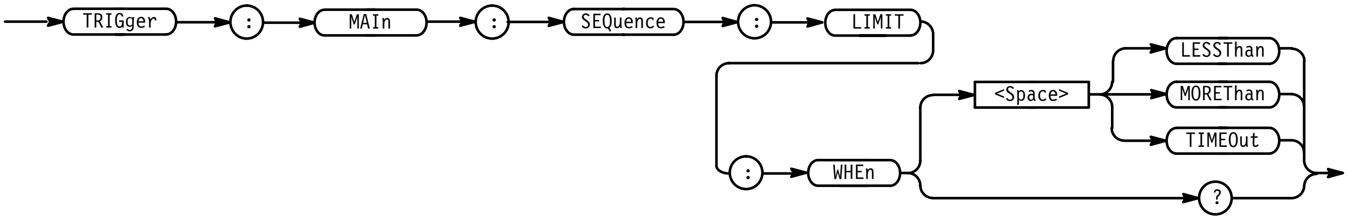
Sets or queries the definition of when the Boolean requirements must cease to meet the conditions of the defined pattern to trigger the logic scope.

Group Trigger

Syntax

```
TRIGger:MAIn:SEQuence:LIMIT:WHEn { LESSThan | MOREThan
| TIMEOut }
```

```
TRIGger:MAIn:SEQuence:LIMIT:WHEn?
```



Arguments `LESSThan` specifies that the logic scope triggers if the trigger inputs cease to meet the Boolean requirements before the specified time.

MOREThan specifies that the logic scope triggers if the trigger inputs cease to meet the Boolean requirements after the specified time.

TIMEOut specifies that the logic scope triggers after the specified amount of time elapses and the trigger inputs still meet the Boolean requirements.

Examples

```
TRIGGER:MAIN:SEQUENCE:LIMIT:WHEN LESSTHAN
```

the logic scope triggers if the trigger inputs cease to meet the Boolean requirements before the specified time.

```
TRIGGER:MAIN:SEQUENCE:LIMIT:WHEN?
```

might return MORETHAN, indicating the logic scope triggers if the trigger inputs cease to meet the Boolean requirements after the specified time.

TRIGger:MAIn:SEQuence:LIMIT:WIDth

Sets or queries the time limit for the Trigger Width definitions of Less Than, More Than, or Time Out. This is equivalent to pressing side menu Time for main sequence time triggers.

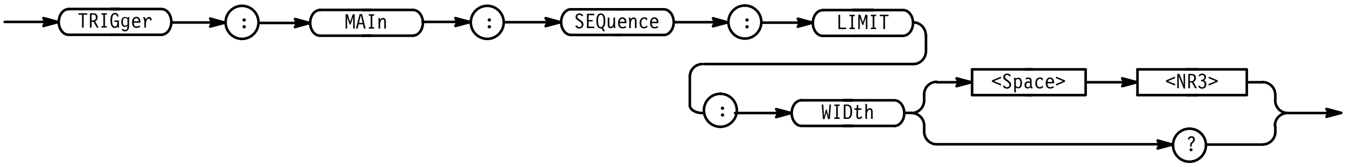
Group

Trigger

Syntax

```
TRIGger:MAIn:SEQuence:LIMIT:WIDth <NR3>
```

```
TRIGger:MAIn:SEQuence:LIMIT:WIDth?
```



Arguments

<NR3> is the time limit from 2.0 ns to 1.0 seconds.

Examples

```
TRIGGER:MAIN:SEQUENCE:LIMIT:WIDTH 1.0E-3
```

sets the trigger width time limit to 1.0 ms.

```
TRIGGER:MAIN:SEQUENCE:LIMIT:WIDTH?
```

might return 2.0E-9, indicating the trigger width time limit is 2 ns.

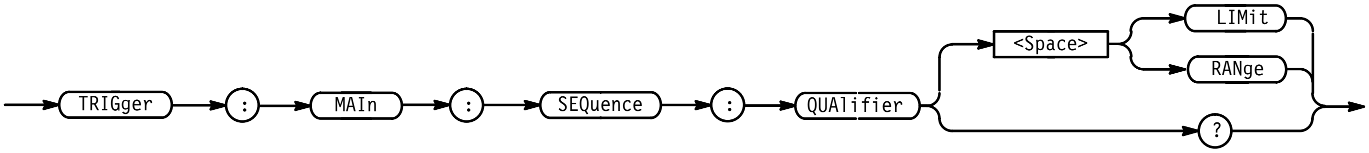
TRIGger:MAIn:SEQuence:QUAlifier

Sets or queries the Time Qualifier used to define when the logic scope triggers. Using this command is equivalent to selecting the time qualifier Limit or Range for the main sequence trigger.

Group Trigger

Syntax `TRIGger:MAIn:SEQuence:QUAlifier { LIMit | RANge }`

`TRIGger:MAIn:SEQuence:QUAlifier?`



Arguments `LIMIT` defines a maximum time limit for the main trigger event to occur.

`RANge` defines a range of time for the main trigger event to occur.

Examples `TRIGGER:MAIN:SEQUENCE:QUALIFIER LIMIT`
specifies there is a maximum time limit for the main trigger event.

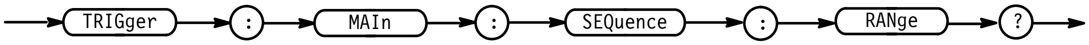
`TRIGGER:MAIN:SEQUENCE:QUALIFIER?`
might return `RANGE`, indicating the main trigger event must occur either within or outside a specified amount of time.

TRIGger:MAIn:SEQuence:RANge? (Query Only)

Returns the main sequence trigger range parameters.

Group Trigger

Syntax `TRIGger:MAIn:SEQuence:RANge?`



Examples `TRIGGER:MAIN:SEQUENCE:RANGE?`
might return `:TRIGGER:MAIN:SEQUENCE:RANGE:WHEN INSIDE;HIGHLIMIT 2.0E-9;LOWLIMIT 2.0E-9`.

TRIGger:MAIn:SEQuence:RANge:HIGHLimit

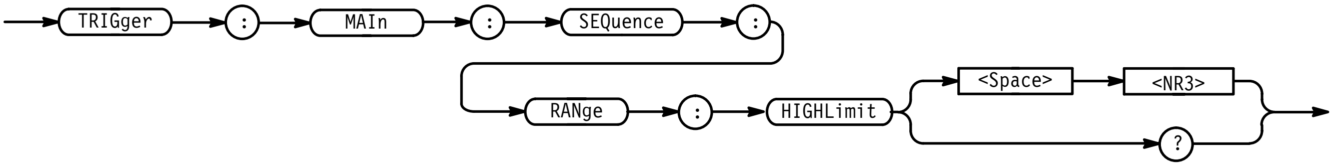
Sets or queries the value for the upper time limit of the range for the range-qualified main trigger. Using this command is equivalent to selecting Upper Limit for main sequence triggers with range selected as the time qualifier.

Group Trigger

Syntax

```
TRIGger:MAIn:SEQuence:RANge:HIGHLimit <NR3>
```

```
TRIGger:MAIn:SEQuence:RANge:HIGHLimit?
```



Arguments <NR3> is the upper limit of the range in seconds.

Examples

```
TRIGGER:MAIN:SEQUENCE:RANGE:HIGHLIMIT 1.0
```

sets the main sequence trigger upper limit range to 1 second.

```
TRIGGER:MAIN:SEQUENCE:RANGE:HIGHLIMIT?
```

might return 100.0E-9, indicating the main sequence trigger upper limit range is set to 100 ns.

TRIGger:MAIn:SEQuence:RANge:LOWLimit

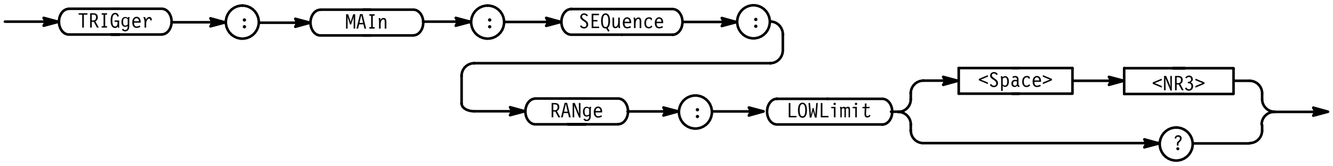
Sets or queries the value for the lower time limit of the range for the range-qualified main trigger. Using this command is equivalent to selecting Lower Limit for main sequence triggers with range selected as the time qualifier.

Group Trigger

Syntax

```
TRIGger:MAIn:SEQuence:RANge:LOWLimit <NR3>
```

```
TRIGger:MAIn:SEQuence:RANge:LOWLimit?
```



Arguments `<NR3>` is the lower limit of the range in seconds.

Examples `TRIGGER:MAIN:SEQUENCE:RANGE:LOWLIMIT 1.0`
sets the main sequence trigger lower limit range to 2 seconds.

`TRIGGER:MAIN:SEQUENCE:RANGE:LOWLIMIT?`
might return `100.0E–9`, indicating the main sequence trigger lower limit range is set to 100 ns.

TRIGger:MAIn:SEQuence:RANge:WHEn

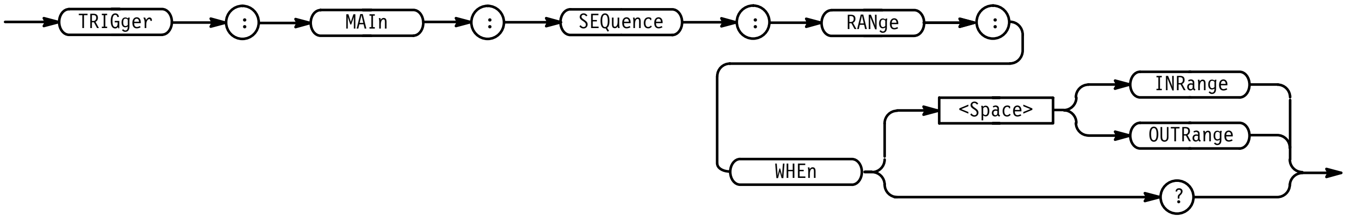
Sets or queries when the channel inputs must cease to meet the Boolean requirements of the pattern definition for the main sequence trigger.

Group Trigger

Syntax

```
TRIGger:MAIn:SEQuence:RANge:WHEn { INRange | OUTRange }
```

```
TRIGger:MAIn:SEQuence:RANge:WHEn?
```



Arguments `INRange` specifies that the trigger inputs must meet the Boolean requirements of the pattern definition for the main sequence trigger for a period of time ending within the user specified time range.

`OUTRange` specifies that the trigger inputs must meet the Boolean requirements of the pattern definition for the main sequence trigger for a period of time ending either before or after the user specified time range.

Examples

`TRIGGER:MAIN:SEQUENCE:RANGE:WHEN INRANGE`
specifies that the trigger inputs for the pattern definition of the main sequence trigger must end “within” the user specified time range.

`TRIGGER:MAIN:SEQUENCE:RANGE:WHEN?`
might return `OUTRANGE`, indicating the trigger inputs for the pattern definition of the main sequence trigger must end either before or after the user specified time range.

TRIGger:MAIn:SEQuence:STARt? (Query Only)

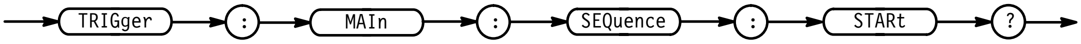
Returns the pattern definition of the start sequence for the main trigger.

Group

Trigger

Syntax

```
TRIGger:MAIn:SEQuence:STARt?
```



Examples

`TRIGGER:MAIN:SEQUENCE:START?`
might return `:TRIGGER:MAIN:SEQUENCE:START:LOGIC AND;INPUT 65537,1`.

TRIGger:MAIn:SEQuence:STARt:AUXiliary

Sets or queries the Boolean requirement (a logic level of High, Low, or Don’t Care) for the rear panel connector labeled AUX TRIGGER INPUT for the main sequence trigger. The Boolean requirements of all inputs define the start pattern recognized by the logic scope.

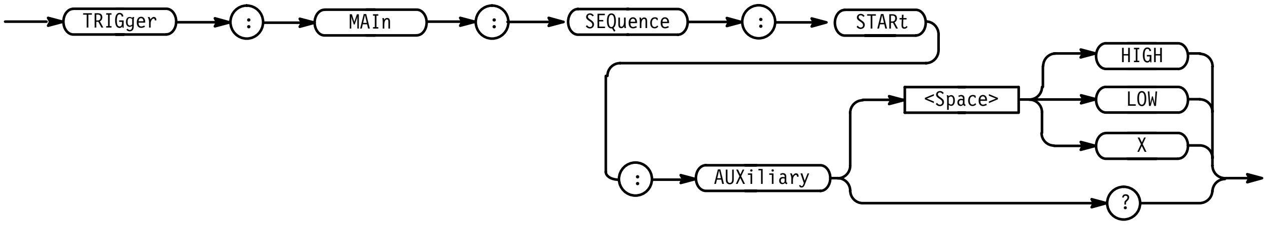
Group

Trigger

Syntax

```
TRIGger:MAIn:SEQuence:STARt:AUXiliary { HIGH | LOW | X }
```

```
TRIGger:MAIn:SEQuence:STARt:AUXiliary?
```



Arguments `HIGH` specifies that the AUX TRIGGER INPUT must be High relative to the threshold level.

`LOW` specifies that the AUX TRIGGER INPUT must be Low relative to the threshold level.

`X` specifies that the AUX TRIGGER INPUT can be either High or Low relative to the threshold level.

Examples `TRIGGER:MAIN:SEQUENCE:START:AUXILIARY LOW`
specifies that the AUX TRIGGER INPUT must be Low to meet the Boolean requirements of the defined pattern for the main sequence trigger.

`TRIGGER:MAIN:SEQUENCE:START:AUXILIARY?`
might return `HIGH`, indicating the AUX TRIGGER INPUT must be High to meet the Boolean requirements of the defined pattern for the main sequence trigger.

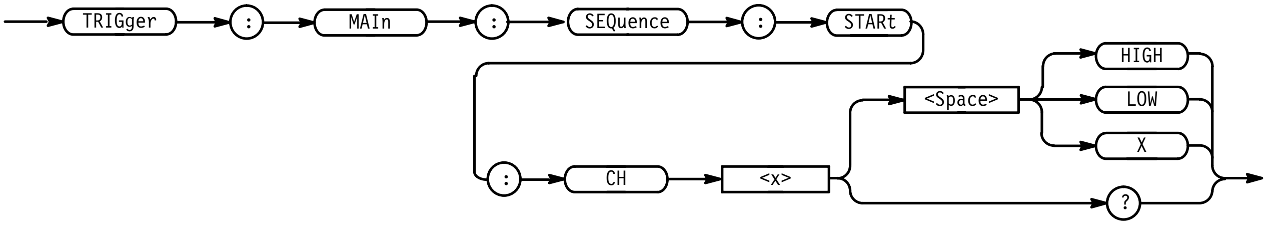
TRIGger:MAIn:SEQuence:STARt:CH<x>

Sets or queries the Boolean requirement (logic levels of High, Low, or Don’t Care) for the selected channel for the main sequence trigger. The Boolean requirements set for all the trigger inputs define the start logic pattern that the logic scope recognizes.

Group Trigger

Syntax `TRIGger:MAIn:SEQuence:STARt:CH<x> { HIGH | LOW | X }` NOTE: defines `CH<x>` pattern bit to High, Low or Don’t Care

`TRIGger:MAIn:SEQuence:STARt:CH<x>?`



Arguments HIGH specifies that the selected channel must be High relative to the threshold level.

LOW specifies that the selected channel must be Low relative to the threshold level.

X specifies that the selected channel can be either High or Low relative to the threshold level.

Examples TRIGGER:MAIN:SEQUENCE:START:CH14 HIGH

specifies that channel 14 must be High to meet the Boolean requirements of the defined pattern for the main sequence trigger.

TRIGGER:MAIN:SEQUENCE:START:CH9 HIGH

might return LOW, indicating that channel 9 must be Low to meet the Boolean requirements of the defined pattern for the main sequence trigger.

TRIGger:MAIn:SEQuence:STARt:INPut

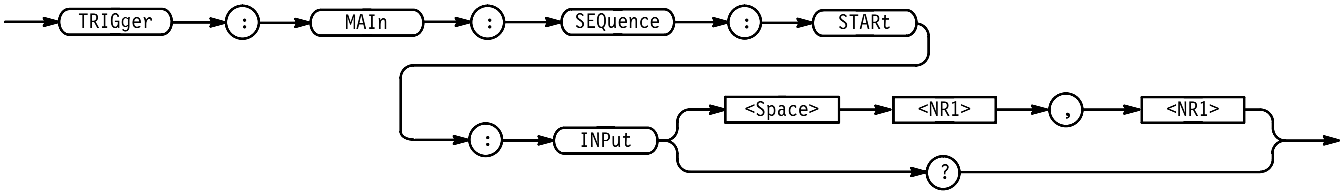
Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

Group Trigger

Syntax

```
TRIGger:MAIn:SEQuence:STARt:INPut <NR1>,<NR1>
```

```
TRIGger:MAIn:SEQuence:STARt:INPut?
```



Arguments

`<NR1>` is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

`<NR1>` is a value from 0 to 131071. The "Don't Care" values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the "Don't Care" status of channels 1 through 16. Bit 17 sets the "Don't Care" of the Auxiliary Input. A bit value of zero will set the Boolean value to "Don't Care" for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples

```
TRIGGER:MAIN:SEQUENCE:START:INPUT 65535, 65535
```

sets all 16 channels to trigger on High input values and sets the Auxiliary Input to "Don't Care".

```
TRIGGER:MAIN:SEQUENCE:START:INPUT 1, 1
```

sets channel 1 High and all other inputs to to "Don't Care".

```
TRIGGER:MAIN:SEQUENCE:START:INPUT?
```

might return `3,3`, indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to "Don't Care".

TRIGger:MAIn:SEQuence:STARt:LOGIC

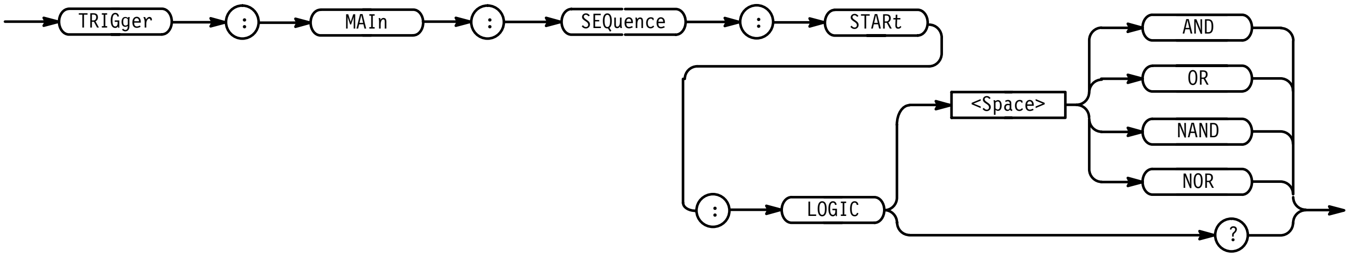
Sets or queries the combinatorial logic pattern applied to the main sequence trigger.

Group

Trigger

Syntax

```
TRIGger:MAIn:SEQuence:STARt:LOGIC { AND | OR | NAND | NOR }
```



Arguments

`AND` applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

`OR` applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

`NAND` applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

`NOR` applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

`TRIGGER:MAIN:SEQUENCE:START:LOGIC AND`
specifies AND logic as the input channel trigger pattern for the main sequence trigger.

`TRIGGER:MAIN:SEQUENCE:START:LOGIC?`
might return `NAND`, indicating NAND logic as the input channel trigger pattern for the main sequence trigger.

TRIGger:MAIn:STATE? (Query Only)

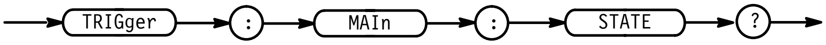
Returns all the main state trigger parameters.

Group

Trigger

Syntax

`TRIGger:MAIn:STATE?`



Examples `TRIGGER:MAIN:STATE?`
might return `:TRIGGER:MAIN:STATE:INPUT 65537,1;LOGIC AND;CLOCK CH1;SLOPE RISE`.

TRIGger:MAIn:STATE:AUXiliary

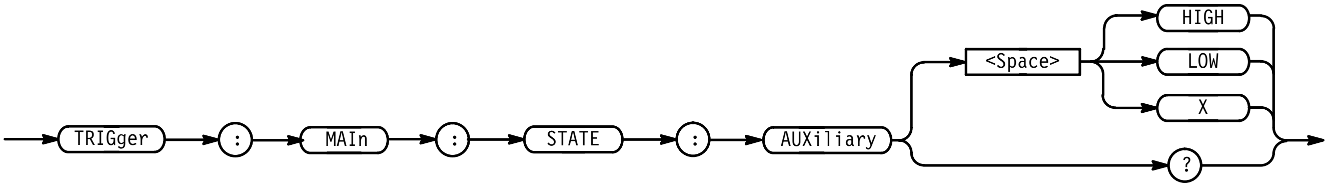
Sets or queries the Boolean requirement (logic levels of High, Low, or Don't Care) for the rear panel connector labeled AUX TRIGGER INPUT. The Boolean requirements set for all the trigger inputs define the logic pattern that triggers the logic scope.

Group Trigger

Syntax

```
TRIGger:MAIn:STATE:AUXiliary { HIGH | LOW | X }
```

```
TRIGger:MAIn:STATE:AUXiliary?
```



Arguments `HIGH` specifies that the AUX TRIGGER INPUT must be High relative to the threshold level.

`LOW` specifies that the AUX TRIGGER INPUT must be Low relative to the threshold level.

`X` specifies that the AUX TRIGGER INPUT can be either High or Low relative to the threshold level.

Examples `TRIGGER:MAIN:STATE:AUXILIARY LOW`
specifies that the AUX TRIGGER INPUT must be Low to meet the Boolean requirements of the defined pattern for the main state trigger.

`TRIGGER:MAIN:STATE:AUXILIARY?`
might return `HIGH`, indicating the AUX TRIGGER INPUT must be High to meet the Boolean requirements of the defined pattern for the main state trigger.

TRIGger:MAIn:STATE:CH<x>

Sets or queries the Boolean requirement (logic levels of High, Low, or Don't Care) for the selected channel. The Boolean requirements set for all the trigger inputs define the logic pattern that triggers the logic scope.

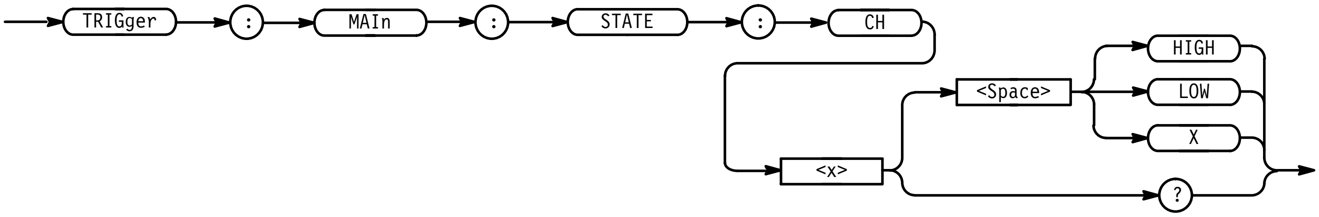
Group

Trigger

Syntax

`TRIGger:MAIn:STATE:CH<x> { HIGH | LOW | X }` NOTE: defines `CH<x>` pattern bit to High, Low or Don't Care

`TRIGger:MAIn:STATE:CH<x>?`



Arguments

`HIGH` specifies that the selected channel must be High relative to the threshold level.

`LOW` specifies that the selected channel must be Low relative to the threshold level.

`X` specifies that the selected channel can be either High or Low relative to the threshold level.

Examples

`TRIGGER:MAIN:STATE:CH14 HIGH`
specifies that channel 14 must be High to meet the Boolean requirements of the defined pattern for the main state trigger.

`TRIGGER:MAIN:STATE:CH11?`
might return `LOW`, indicating that channel 11 must be Low to meet the Boolean requirements of the defined pattern for the main state trigger.

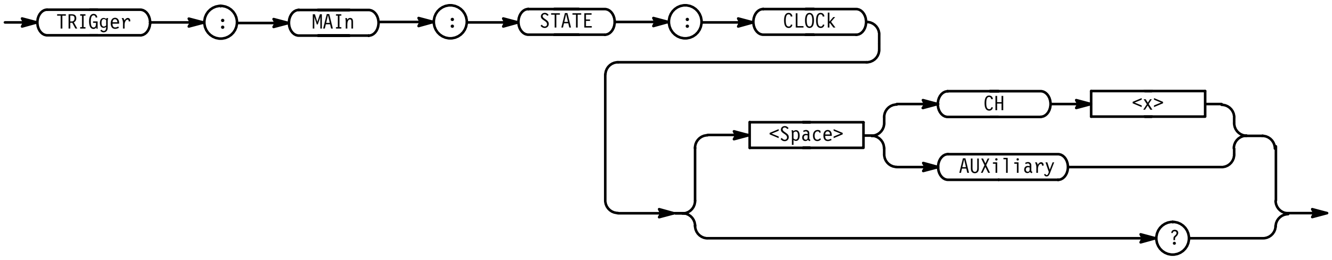
TRIGger:MAIn:STATE:CLOCk

Sets or queries the channel to be used as the clock source for the main state trigger.

Group Trigger

Syntax `TRIGger:MAIn:STATE:CLOCk { CH<x> | AUXiliary }`

`TRIGger:MAIn:STATE:CLOCk?`



Arguments `CH<x>` specifies one of the 16 input channels as the clock source.

`AUXiliary` specifies rear panel connector labeled AUX TRIGGER INPUT as the clock source.

Examples `TRIGGER:MAIN:STATE:CLOCK CH5`
specifies channel 5 as the clock source.

`TRIGGER:MAIN:STATE:CLOCK?`
might return `CH3`, identifying channel 3 as the clock source.

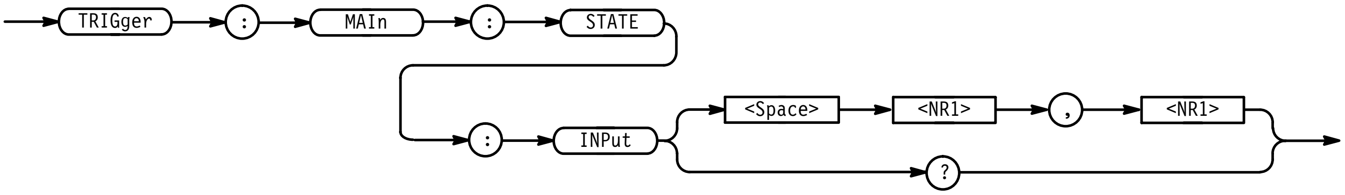
TRIGger:MAIn:STATE:INPut

Sets or queries the logic pattern input values for all 16 channels and the Auxiliary Trigger input.

Group Trigger

Syntax `TRIGger:MAIn:STATE:INPut <NR1>,<NR1> NOTE: 1 High, 0 low`

`TRIGger:MAIn:STATE:INPut?`



Arguments <NR1> is a value from 0 to 131071. The logic pattern input values of all channels and the Auxiliary Trigger input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the High or Low values of channels 1 through 16. Bit 17 is used to set the High or Low value of the Auxiliary Trigger input. A bit value of 1 for a channel indicates High, a bit value of zero indicates a Low. NOTE: Bit values are ignored if the matching bit in the second parameter is zero.

<NR1> is a value from 0 to 131071. The “Don’t Care” values of all channels and the Auxiliary Input are set according to this value. The parameter is interpreted as a 17-bit binary number. Bits 1 through 16 are used to set the “Don’t Care” status of channels 1 through 16. Bit 17 sets the “Don’t Care” of the Auxiliary Input. A bit value of zero will set the Boolean value to “Don’t Care” for that channel, regardless of the first parameter bit value. A bit value of 1 for a channel indicates the high or low value set by the previous parameter is used.

Examples

```
TRIGGER:MAIN:STATE:INPUT 65535, 65535
```

sets all 16 channels to trigger on High input values and sets the Auxiliary Input to “Don’t Care”.

```
TRIGGER:MAIN:STATE:INPUT 1, 1
```

sets channel 1 High and all other inputs to to “Don’t Care”.

```
TRIGGER:MAIN:STATE:INPUT?
```

might return `3,3`, indicating channels 1 and 2 will trigger when their input values are High. The other channels and the Auxiliary bits are set to “Don’t Care”.

TRIGger:MAIn:STATE:LOGIC

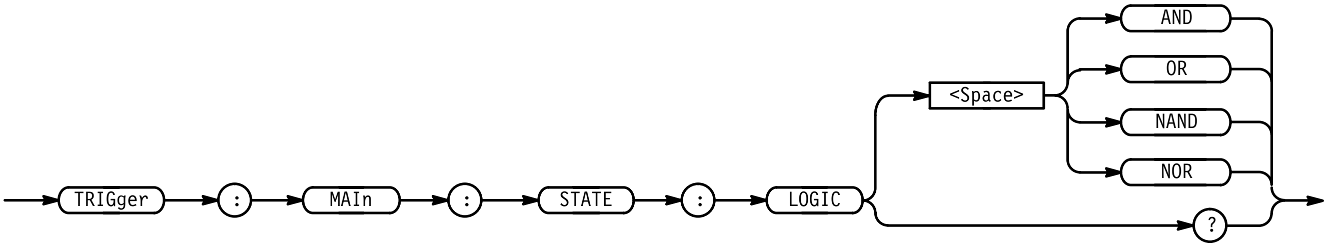
Sets or queries the logic function applied to the main state trigger inputs.

Group Trigger

Syntax

```
TRIGger:MAIn:STATE:LOGIC { AND | OR | NAND | NOR }

TRIGger:MAIn:STATE:LOGIC?
```



Arguments

AND applies AND logic to the input channels. AND is a logic (Boolean) function in which the output is true when and only when all the inputs are true.

OR applies OR logic to the input channels. OR is a logic (Boolean) function in which the output is true if any of the inputs are true. Otherwise the output is false.

NAND applies NAND logic to the input channels. NAND is a logic (Boolean) function in which the output of the AND function is complemented (true becomes false, and false becomes true).

NOR applies NOR logic to the input channels. NOR is a logic (Boolean) function in which the output of the OR function is complemented (true becomes false, and false becomes true).

Examples

```
TRIGGER:MAIN:STATE:LOGIC AND
```

specifies AND logic as the input channel trigger pattern for the main state trigger.

```
TRIGGER:MAIN:STATE:LOGIC?
```

might return NOR, indicating NOR logic as the input channel trigger pattern for the main state trigger.

TRIGger:MAIn:STATE:SLOpe

Sets or queries the transition edge (rising or falling) that triggers the logic scope.

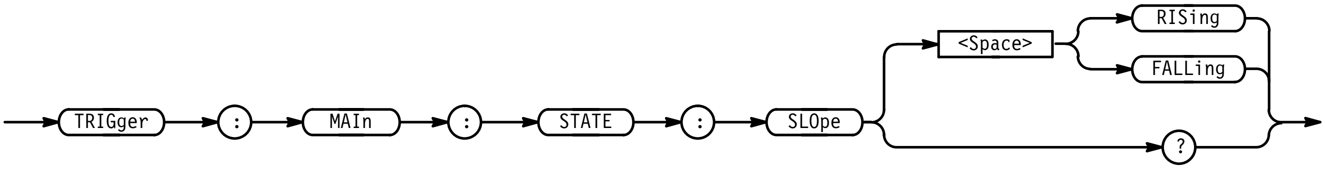
Group

Trigger

Syntax

```
TRIGger:MAIn:STATE:SLOpe { RISing | FALLing }
```

```
TRIGger:MAIn:STATE:SLOpe?
```



Arguments

`RISing` specifies to trigger on the rising transition (positive) edge of a signal.

`FALLing` specifies to trigger on the falling transition (negative) edge of a signal.

Examples

`TRIGGER:MAIN:STATE:SLOPE RISING`
instructs the logic scope to trigger on the rising transition edge of the signal.

`TRIGGER:MAIN:STATE:SLOPE?`
might return `FALLING`, indicating the logic scope is triggering on the falling transition edge of the signal.

TRIGger:MAIn:TYPe

Sets or queries the main trigger type. Using this command is equivalent to setting Type in the Trigger menu.

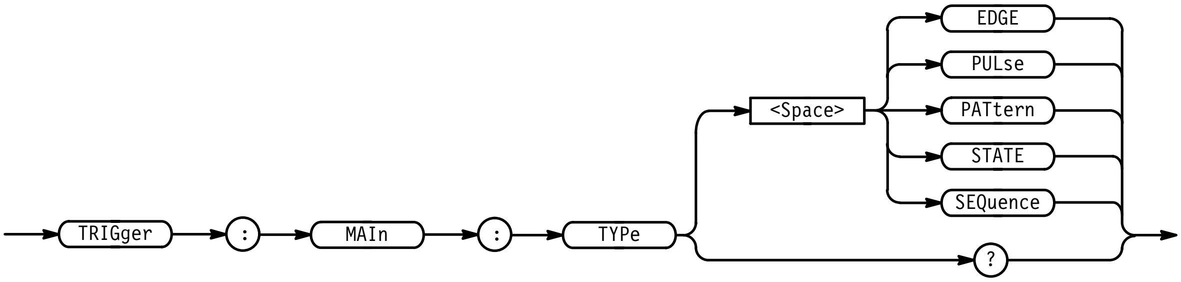
Group

Trigger

Syntax

```
TRIGger:MAIn:TYPe { EDGE | PULse | PATtern | STATE | SEQuence }
```

```
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. The conditions are controlled by the TRIGger:MAIn:EDGE commands.

`PULse` specifies that a trigger occurs when a pulse meets specified conditions related to its width.

`PATtern` specifies that triggers occur when a specified pattern at the input channels meet specified Boolean and time qualifications.

`STATE` specifies that a trigger occurs when specified patterns at the input channels meet specified Boolean conditions *at the time the clock input changes state.*

`SEQuence` specifies that a trigger occurs when a *sequence* of two patterns exist at the input channels meeting specified Boolean and time qualifications.

Examples

`TRIGGER:MAIN:TYPE EDGE`
specifies `EDGE` (the normal trigger condition) as the main trigger type.

`TRIGGER:MAIN:TYPE?`
might return `PULSE` indicating that the main trigger type is a pulse trigger.

TRIGger:STATE? (Query Only)

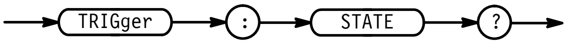
Returns the current state of the triggering system.

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 logic scope 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 recognized a trigger and is acquiring 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.

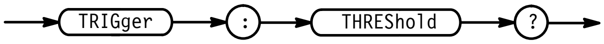
Examples `TRIGGER:STATE?`
might return `ARMED`, indicating that pretrigger data is being acquired.

TRIGger:THRESHold? (Query Only)

Returns the trigger thresholds for all channels; not effective for edge triggering.

Group Trigger

Syntax `TRIGger:THRESHold?`



Examples `TRIGGER:THRESHOLD?`
might return `:TRIGGER:THRESHOLD:CH1 0.0E+0;CH2 0.0E+0;CH3 0.0E+0;CH4 0.0E+0;CH5 0.0E+0;CH6 0.0E+0;CH7 0.0E+0;CH8 0.0E+0;CH9 0.0E+0;CH10 0.0E+0;CH11 0.0E+0;CH12 0.0E+0;CH13 0.0E+0;CH14 0.0E+0;CH15 0.0E+0;CH16 0.0E+0`.

TRIGger:THRESHold:CH<x>

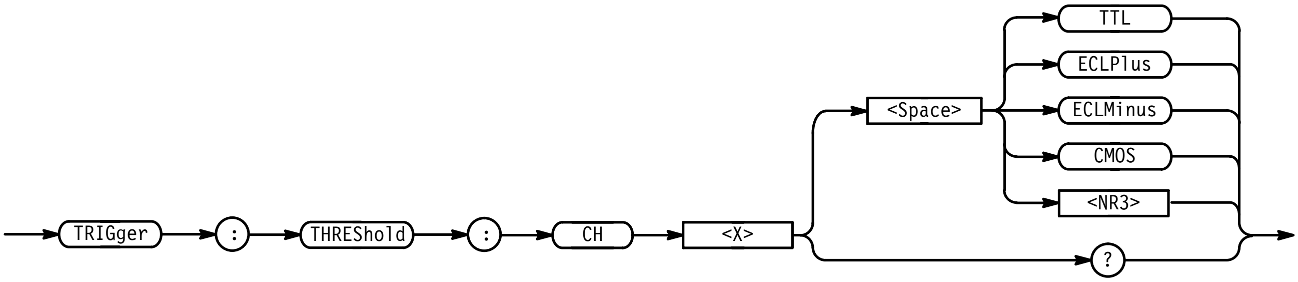
Sets or queries the trigger threshold for the channel input defined by `Ch<x>`.

Group Trigger

Syntax

```
TRIGger:THRESHold:CH<x> { TTL | ECLPlus | ECLMinus | CMOS |
<NR3> }
```

```
TRIGger:THRESHold:CH<x>?
```



Arguments TTL specifies a preset TTL level of 1.4 V.

ECLPlus specifies a preset level of 3.7 V.

ECLMinus specifies a preset ECL level of –1.3 V.

CMOS specifies a preset level of 2.5 V.

<NR3> specify a new threshold in volts.

Examples TRIGGER:THRESHOLD:CH4 TTL
specifies TTL as the preset trigger threshold for channel 4.

TRIGGER:THRESHOLD:CH7?
might return 2.5E+0, indicating a user defined trigger threshold of 2.5 volts for channel 7.

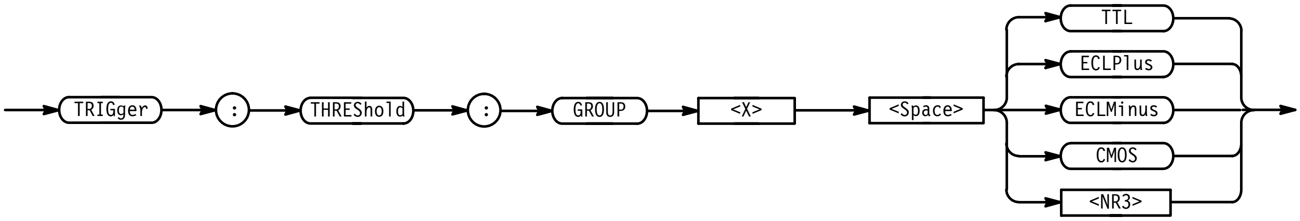
TRIGger:THRESHold:GROUP<x> (No Query Form)

Sets the trigger thresholds of the group specified by <x> for trigger types other than edge.

Group Trigger

Syntax

```
TRIGger:THRESHold:GROUP<x> { TTL | ECLPlus | ECLMinus | CMOS |
<NR3> }
```



Arguments TTL specifies a preset TTL level of 1.4 V.

ECLPlus specifies a preset level of 3.7 V.

ECLMinus specifies a preset ECL level of –1.3 V.

CMOS specifies a preset level of 2.5 V.

<NR3> specify a new threshold in volts.

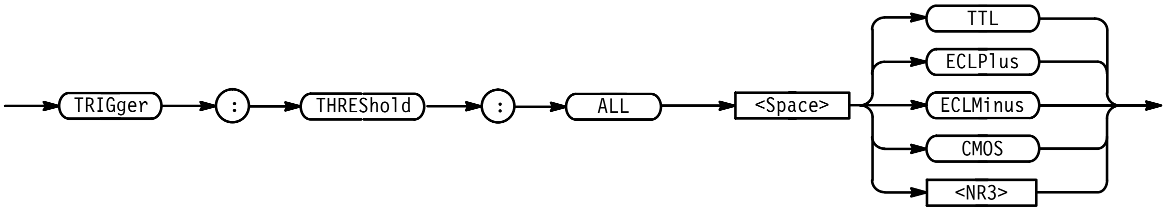
Examples TRIGGER:THRESHOLD:GROUP3 TTL
specifies TTL as the preset trigger threshold for group 3.

TRIGger:THRESHold:ALL (No Query Form)

Sets all nonedge trigger thresholds.

Group Trigger

Syntax TRIGger:THRESHold:ALL { TTL | ECLPlus | ECLMinus | CMOS | <NR3> }



Arguments TTL specifies a preset TTL level of 1.4 V.

ECLPlus specifies a preset level of 3.7 V.

ECLMinus specifies a preset ECL level of –1.3 V.

CMOS specifies a preset level of 2.5 V.

<NR3> specify a new threshold in volts.

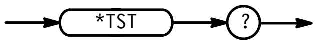
Examples TRIGGER:THRESHOLD:ALL ECL
specifies ECL as the preset trigger threshold for all nonedge triggers.

*TST? (Query Only)

Performs a GPIB interface self-test.

Group Miscellaneous

Syntax `*TST?`



Returns `<NR1>` and is always 0.

UNLock (No Query Form)

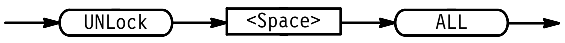
Unlocks the front panel. This command is equivalent to LOCk NONe.

***NOTE**. If the logic scope 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.

Examples `UNLOCK ALL`
unlocks all front-panel controls.

VERBose

Sets and queries the Verbose State controlling the length of keywords for query responses. Keywords may be both headers and arguments. This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk).

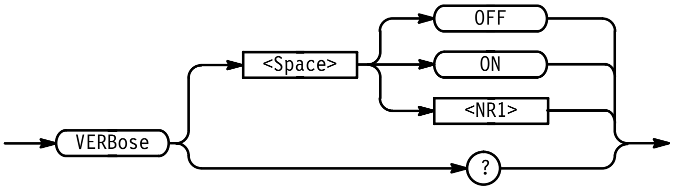
Group Miscellaneous

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 1, showing that the Verbose State is true.

*WAI (No Query Form)

Wait. Prevents the logic scope from executing further commands or queries until all pending operations are completed. This command allows synchronization of the logic scope with user application programs. Synchronization methods are described on page 3–7.

Group Status and Error

Related Commands BUSY?, *OPC

Syntax `*WAI`

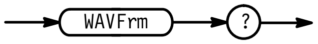
WAVFrm? (Query Only)

Returns WFMPre? and CURVe? data for waveform(s) 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?`



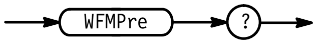
WFMPre? (Query Only)

Returns the waveform formatting data for the first ordered waveform as specified by the DATa:SOURce command. All channel and reference 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

Sets or returns the number of bits per binary waveform point (as specified by the DATa:SOUrce command) for the first ordered waveform. The WFMPre:BIT_Nr command is ignored on input.

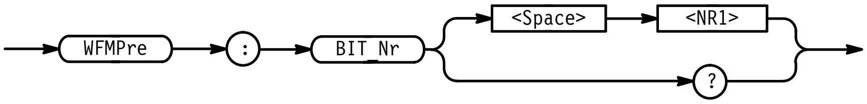
Group Waveform

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 16
```

specifies 16 bits per waveform point.

```
WFMPRE:BIT_NR?
```

might return 8, indicating 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.

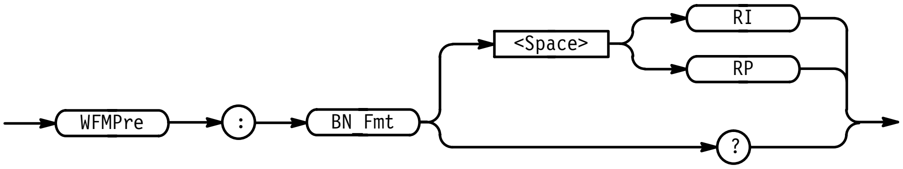
Group Waveform

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.

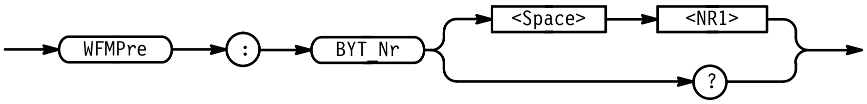
Group Waveform

Related Commands DATa:WIDth, WFMPre:BIT_Nr

Syntax

```
WFMPre:BYT_Nr <NR1>
```

```
WFMPre:BYT_Nr?
```



Arguments <NR1> is the number of bytes per point: 1 or 2.

Examples

```
WFMPRE:BYT_NR 2
```

specifies there are 2 bytes per waveform data point.

```
WFMPRE:BYT_NR?
```

might return 1, indicating there is one byte per waveform data point.

WFMPre:BYT_Or

Selects or queries which byte of the binary waveform data is transmitted first during a waveform data transfer, when DATa:WIDth (or WFMPre:BYT_Nr) is set to 2.

Group

Waveform

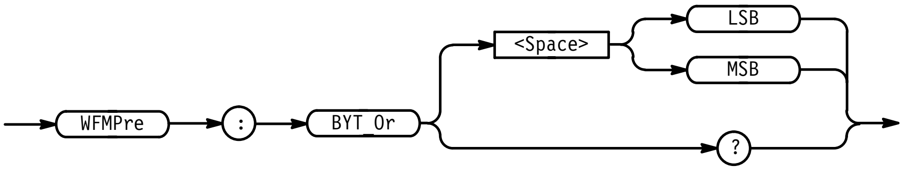
Related Commands

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 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 waveform encoding used when transferring data with the CURVe command.

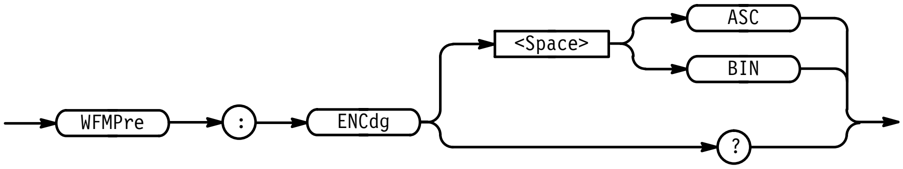
Group Waveform

Related Commands DATa:ENCdg, WFMPre:BYT_Or, WFMPre:BN_Fmt

Syntax

```
WFMPre:ENCdg { ASC | BIN }

WFMPre:ENCdg?
```



Arguments `ASC` specifies ASCII encoded curve data.

`BIN` specifies binary encoded curve data.

Examples

`WFMPRE:ENCDG ASC`
specifies that waveform data is in ASCII format.

`WFMPRE:ENCDG?`
might return `BIN`, indicating that waveform data is in binary format.

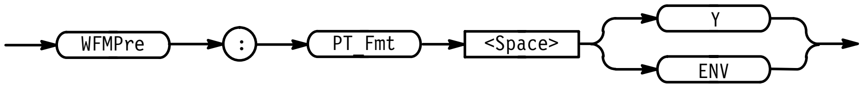
WFMPre:PT_Fmt (No Query Form)

Selects the waveform data point format for the ordered waveforms specified by the DATa:SOUrce command.

Group Waveform

Syntax

```
WFMPre:PT_Fmt { Y | ENV }
```



Arguments 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:

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

$$Y_n = YZEro + YMUlt\ (y_n - YOFf)$$

ENV specifies that the waveform is transmitted as maximum and minimum point pairs. Only *y* values are explicitly transmitted. Absolute coordinates are given by:

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

$$Y_{n_{max}} = YZEro + YMUlt\ (y_{n_{max}} - YOFf)$$

$$Y_{n_{min}} = YZEro + YMUlt\ (y_{n_{min}} - YOFf)$$

Examples

```
WFMPRE:PT_Fmt ENV
```

sets the waveform data point format to enveloped.

WFMPre:PT_Off (No Query Form)

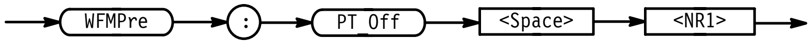
Specifies the trigger point within the waveform record for the reference waveform specified by the DATa:DESTination command.

Group Waveform

Related Commands HORizontal:TRIGger:POsition

Syntax

```
WFMPre:PT_Off <NR1>
```



Arguments <NR1> specifies the position of the trigger point relative to DATa:STARt, from 0 to the record length.

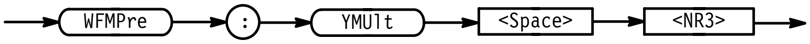
Examples `WFMPRE:PT_OFF 1`
specifies that the trigger point is the first point in the waveform record.

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 reference waveform vertical scale factor in YUNits (usually volts), per binary data point.

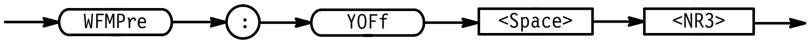
Examples `WFMPRE:YMULT 80.0E-3`
specifies that the reference waveform vertical scale factor is 2 volts/div.

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.

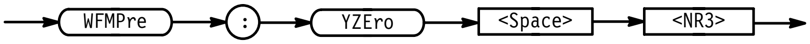
Examples `WFMPRE:YOFF 127.0`
specifies that the vertical offset is 127.

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 the offset in YUNits (usually volts).

Examples `WFMPRE:YZEro 1.0`
specifies that the vertical offset is 1.0 volts.

Table 2–25 lists additional WFMPre arguments included for compatibility purposes.

***NOTE**. These commands do not support a query form and all information is ignored.*

Table 2–25: 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

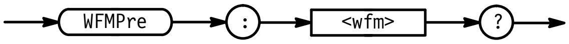
***NOTE**. When returning WFMPRE:<wfm> information from the logic scope, <wfm> specifies the waveform source (CH<x> or REF<x>). The source must also be set using the DAta:SOUrce command. When sending WFMPRE:<wfm> information to the scope, the <wfm> specification is ignored and the reference location 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 reference waveforms must be displayed before they can be queried. Querying an invalid reference waveform generates an execution error.

Group Waveform

Syntax `WFMPre:<wfm>?`



Mnemonics <wfm> = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

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 in the transmitted waveform record. This value is ignored on input.

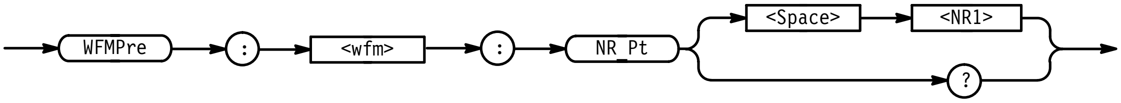
Group Waveform

Related Commands DATa:DESTination

Syntax

```
WFMPre:<wfm>:NR_Pt <NR1>
```

```
WFMPre:<wfm>:NR_Pt?
```



Mnemonics

<NR1> = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments <NR1> = 0 to the waveform record length. <NR1> = 0 indicates the waveform record is of an unspecified length.

Examples

```
WFMPRE:CH1:NR_Pt?
```

might return 2000, the number of data points in the waveform record transferred from channel 1.

WFMPre:<wfm>:PT_Fmt

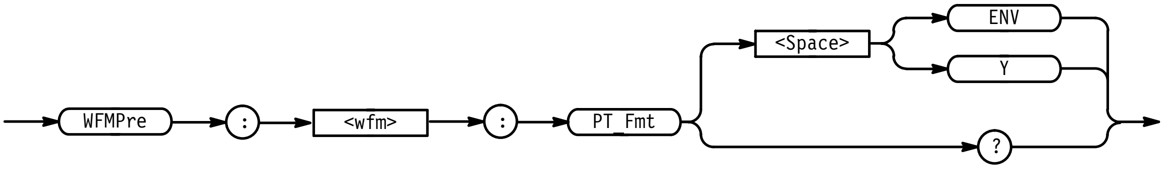
Selects or returns the data point format for the first ordered waveform as selected by the DATa:SOUrce command. A <wfm> input always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

Group Waveform

Related Commands DATa:DESTination

Syntax WFMPre:<wfm>:PT_Fmt { ENV | Y }

WFMPre:<wfm>:PT_Fmt?



GROUP works on selected channel value.

Mnemonics <wfm> = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

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:CH1:PT_FMT Y
specifies a normal waveform for channel one (one ASCII or binary data point is transmitted for each point in the waveform record).

WFMPRE:CH2:PT_FMT?
might return ENV, indicating that CH2 waveform data format is enveloped.

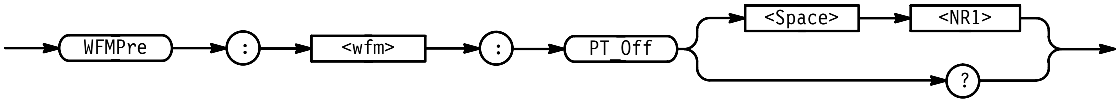
WFMPre:<wfm>:PT_Off

Selects or 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?`



Mnemonics `<wfm>` = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments `<NR1>` = 0 to the record length, and is the position of the trigger point relative to DATa:STARt when queried.

Examples `WFMPRE:REF1:PT_OFF 100`
specifies `100` as the trigger point within the waveform record.

`WFMPRE:CH1:PT_OFF?`
might return `250`, indicating the trigger position within the waveform record.

WFMPre:<wfm>:WFId

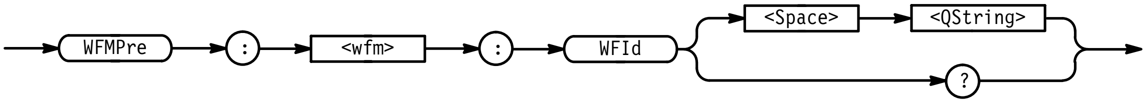
Sets or returns waveform or group information pertinent to input coupling, volts/division, time/division, acquisition mode, and record length.

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

Group Waveform

Syntax `WFMPre:<wfm>:WFId <QString>`

```
WFMPre:<wfm>:WFId?
```



Mnemonics <wfm> = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments <QString> is the waveform identifier string.

Examples `WFMPRE:GROUP1:WFID?`

might return `WFMPRE:GROUP1:WFID "GRP1, DC COUPLING, 1.000 VOLTS/DIV, 250.0 µS/DIV, 500 POINTS, SAMPLEMODE"` , the parameters for selected channel one.

WFMPre:<wfm>:XINcr

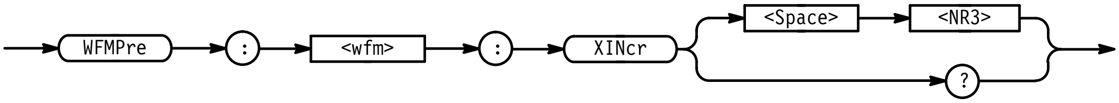
Sets or queries the horizontal sampling interval. A <wfm> input always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

Group Waveform

Syntax

```
WFMPre:<wfm>:XINcr <NR3>
```

```
WFMPre:<wfm>:XINcr?
```



Mnemonics <wfm> = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments

<NR3> is the horizontal sampling interval in seconds per division.

Examples

`WFMPRE:CH9:XINCR 50.0E-6`
sets the horizontal sampling interval of channel 9 to 2.5 msec/div.

`WFMPRE:GROUP4:XINCR?`
might return `5.0E-6`, indicating the horizontal sampling interval of group 4 is 250.0 µs/div.

WFMPre:<wfm>:XUNit

Sets or returns the horizontal (X-axis) units of the waveform data at the time of creation.

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

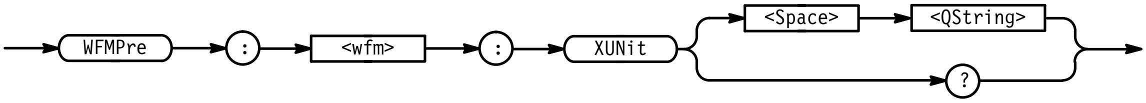
Group

Waveform

Syntax

`WFMPre:<wfm>:XUNit <QString>`

`WFMPre:<wfm>:XUNit?`



Mnemonics

<wfm> = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments

<QString> is the horizontal scale factor in “s” for seconds.

Examples

`WFMPRE:GROUP1:XUNIT “s”`
sets group 1 horizontal units to seconds.

`WFMPRE:CH1:XUNIT?`
might return `"s"`, indicating that the horizontal units for channel 1 are seconds.

WFMPre:<wfm>:YMUlt

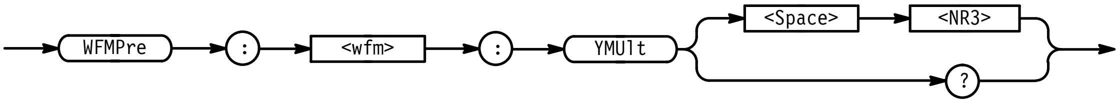
Sets or queries the vertical scale factor, in YUNit per unscaled data point value. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

Group Waveform

Syntax

```
WFMPre:<wfm>:YMUlt <NR3>
```

```
WFMPre:<wfm>:YMUlt?
```



Mnemonics <wfm> = CH<x>
REF y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments <NR3> is the vertical scale factor in YUNits (usually volts), per digitizing level.

Examples

```
WFMPRE:GROUP4:YMULT 80.0E-3
```

sets the group 4 vertical scale factor to 2.0 volts/div.

```
WFMPRE:CH5>:YMULT?
```

might return 40.0E-3, indicating the vertical scale factor for channel 5 is 1.0 volts/div.

WFMPre:<wfm>:YOFf

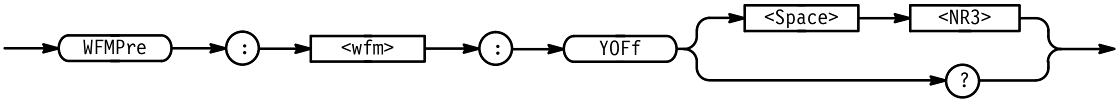
Sets or queries the vertical position of the waveform. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

Group Waveform

Syntax

```
WFMPre:<wfm>:YOFf <NR3>
```

```
WFMPre:<wfm>:YOFf?
```



Mnemonics `<wfm>` = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments `<NR3>` is the vertical position of the waveform in digitizing levels.

Examples `WFMPRE:CH2:YOFF`
sets the channel 2 waveform vertical offset position to 0.0.

`WFMPRE:GROUP2:YOFF?`
might return `127`, indicating the vertical offset position of group 2.

WFMPre:<wfm>:YUNit

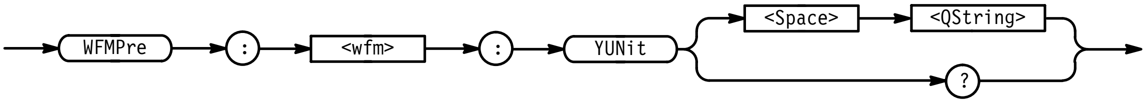
Sets or returns the vertical (Y-axis) units of the waveform data at the time of creation.

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

Group Waveform

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

`WFMPre:<wfm>:YUNit?`



Mnemonics `<wfm>` = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments <QString> is "V" for volts or "VV" for volts^2, and specifies the vertical units.

Examples `WFMPRE:GROUP3:YUNIT "V"`
sets the vertical units of group 3 to volts.

`WFMPRE:CH2:YUNIT?`
might return "V", indicating the units for the vertical component of the channel 2 waveform data are volts.

WFMPre:<wfm>:YZEro

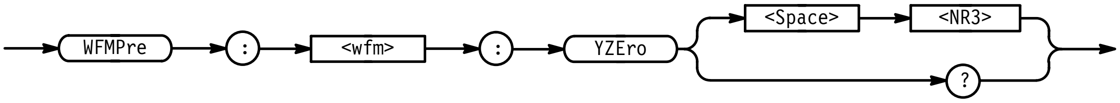
Sets or queries the vertical (Y-axis) offset voltage. On input <wfm> always defaults to the reference location specified by DATa:DESTination regardless of what is sent.

Group Waveform

Syntax

```
WFMPre:<wfm>:YZEro <NR3>

WFMPre:<wfm>:YZEro?
```



Mnemonics <wfm> = CH<x>
REF<y>
GROUP<z>
REF<y>CH<x>

GROUP works on the selected channel value.

Arguments <NR3> is the vertical offset in YUNits (usually volts).

Examples `WFMPRE:GROUP3:YZERO 2.0`
sets the vertical offset of group 3 to 2 .0 volts.

`WFMPRE:CH10:YZERO?`
might return 1.0E+0, indicating the vertical offset of channel 10 is 1.0 volts.

ZOOm

Resets the display to its normal state and 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.

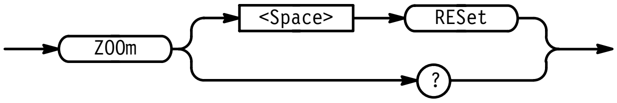
Group

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? (Query Only)

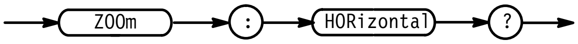
Returns the current horizontal positioning and scaling of the display.

Group

Zoom

Syntax

```
ZOOm:HORizontal?
```



Examples

```
ZOOM:HORIZONTAL?
```

might return `:ZOOM:HORIZONTAL:SCALE 1.0E+0;POSITION 500.0E-3;LOCK ALL`.

ZOOm:HORizontal:LOCk

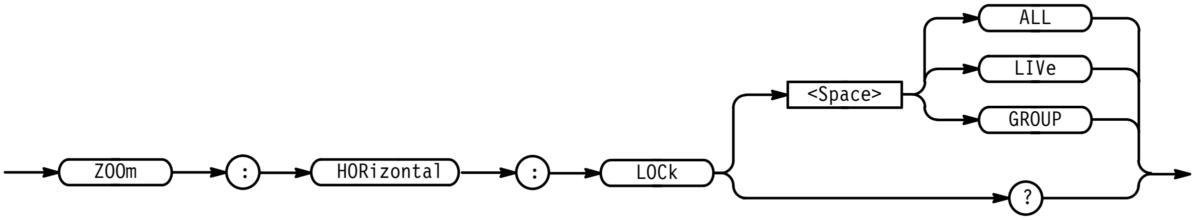
Specifies the waveforms that the horizontal zoom parameters affect. This is equivalent to setting Horizontal Lock in the Zoom side menu.

Group Zoom

Syntax

```
ZOOm:HORizontal:LOCk { ALL | LIVe | GROUP }
```

```
ZOOm:HORizontal:LOCk?
```



Arguments `ALL` specifies that all live (GROUP<x>) and Reference (Ref<y>) waveform groups will be horizontally positioned and scaled together.

`LIVe` specifies that all live groups (GROUP<x>) will be horizontally positioned and scaled together.

`GROUP` specifies that only the currently selected waveform group (live or reference) will be positioned and scaled using the horizontal zoom parameters.

Examples

`ZOOM:HORIZONTAL:LOCK LIVE`

specifies that all live groups (group<x> will be positioned and scaled together.

`ZOOM:HORIZONTAL:LOCK?`

might return `ALL`, indicating that all live and reference groups are horizontally positioned and scaled together.

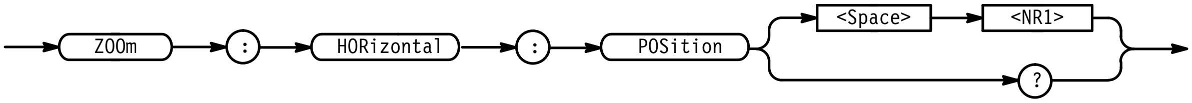
ZOOm:HORizontal:POSition

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 <NR1>`

`ZOOm:HORizontal:POSition?`



Arguments `<NR1>` is from 0 to 100, and is the percent of the waveform to the left of screen center.

Examples `ZOOM:HORIZONTAL:POSITION 50`
centers the waveform on the display.

`ZOOM:HORIZONTAL:POSITION?`
might return `100`, indicating 100% of the waveform is left of center.

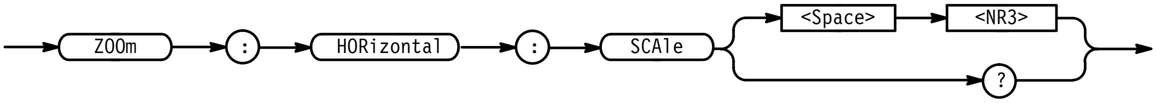
ZOOm:HORizontal:SCAle

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 2.0`
sets the horizontal scale factor to two times the horizontal scale.

`ZOOM:HORIZONTAL:SCALE?`
might return `1.00E+0`, indicating that the horizontal scale factor is 1.

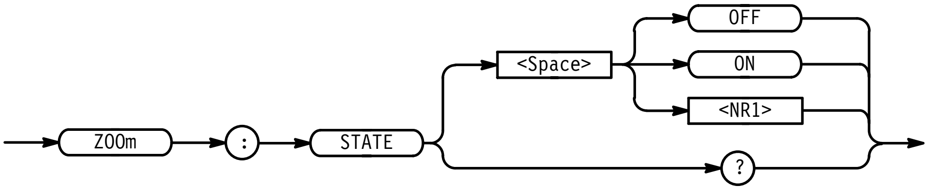
ZOOm:STATE

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 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.

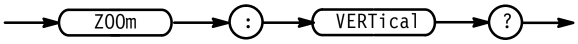
`ZOOM:STATE?`
returns either `0` or 1 depending on the state of Zoom mode.

ZOOm:VERTical? (Query Only)

Returns the current vertical positioning and scaling of the display.

Group Zoom

Syntax ZOOm:VERTical?



Examples `ZOOM:VERTICAL?`
might return `:ZOOM:VERTICAL:SCALE 1.0E+0;POSITION 0.0E+0`.

ZOOm:VERTical:POSition

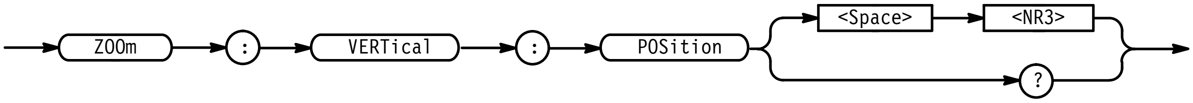
Sets or queries the vertical position of waveforms.

Group Zoom

Syntax

```
ZOOm:VERTical:POSition <NR3>

ZOOm:VERTical:POSition?
```



Arguments <NR3> is the vertical position in divisions.

Examples

`ZOOM:VERTICAL:POSITION 1.0`
sets the vertical position of waveforms to one division above the current vertical position.

`ZOOM:VERTICAL:POSITION?`
might return `:ZOOM:VERTICAL:POSITION 0`

ZOOm:VERTical:SCAle

Sets or queries the vertical expansion and compression factor.

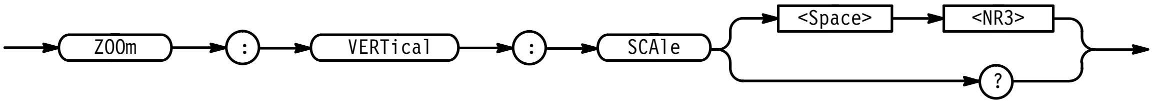
Group Zoom

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 2.0`
sets the vertical expansion to two times the vertical scale.

`ZOOM:VERTICAL:SCALE?`
might return `ZOOM:VERTICAL:SCALE 1.0E+0`, indicating the vertical scale factor is one.

Status and Events

The TLS 216 Logic Scope provides a status and event reporting system for the General Purpose Interface Bus (GPIB). This system informs you of certain significant events that occur within the logic scope.

The logic scope 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 and the program requests information from the logic scope.

Registers

The registers in the event handling system fall into two functional groups:

- Status Registers contain information about the status of the logic scope. 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 logic scope 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 logic scope sets the bit representing that type of event to a value of one. (You can disable bits so they ignore events and remain at zero. See the Enable Registers section on page 3–3.) Reading the status registers tells you what types of events have occurred.

The Standard Event Status Register (SESR) shown in Figure 3–1, records eight types of events that can occur within the logic scope. Use the *ESR? query to read the SESR register. Reading the register clears the bits of the register so 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 logic scope 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 logic scope was parsing a command or query. Command error messages are listed in Table 3–4 on page 3–13.
4	**EXE** (Execution Error). Shows that an error occurred while the logic scope was executing a command or query. Execution error messages are listed in Table 3–5 on page 3–15.
3	**DDE** (Device Error). Shows that a device error occurred. Device error messages are listed in Table 3–6 on page 3–18.
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 3–2, records whether output is available in the Output Queue, whether the logic scope 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.

7	6	5	4	3	2	1	0
—	RQS 6 MSS	ESB	MAV	—	—	—	—

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 logic scope 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

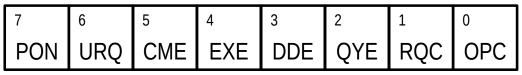
DESER, ESER, and SRER allow you to select which events are reported to the Status Registers and the Event Queue. Each Enable Register acts as a filter to a Status Register (the DESER also acts as a filter to the Event Queue) and can prevent information from being recorded in the register or queue.

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 3–3. This register controls which types of events are reported to the SESR and the Event Queue. The bits in the DESER correspond to those in the SESR, as described earlier.

Use the DESE command to enable and disable the bits in the DESER. Use the DESE? query to read the DESER.



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 3–4. It controls which types of events are summarized by the Event Status Bit (ESB) in the SBR.

Use the *ESE command to set the bits in the ESER. 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 3–5. 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 nonvolatile 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 logic scope status and event reporting system contains two queues: the Output Queue and the Event Queue.

The Output Queue

The logic scope stores query responses in the Output Queue. It empties this queue each time it receives a new command or query message after an End Of Message (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 logic 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? query, and empties the SESR.

Reading the SESR erases any events that were summarized by previous *ESR? reads but not read from the Event Queue. Events that follow an *ESR? read are put in the Event Queue but are not available until *ESR? is used again.

Event Handling Sequence

Figure 3–6 shows how to use the status and event handling system. In the explanation that follows, numbers in parentheses refer to numbers in Figure 3–6.

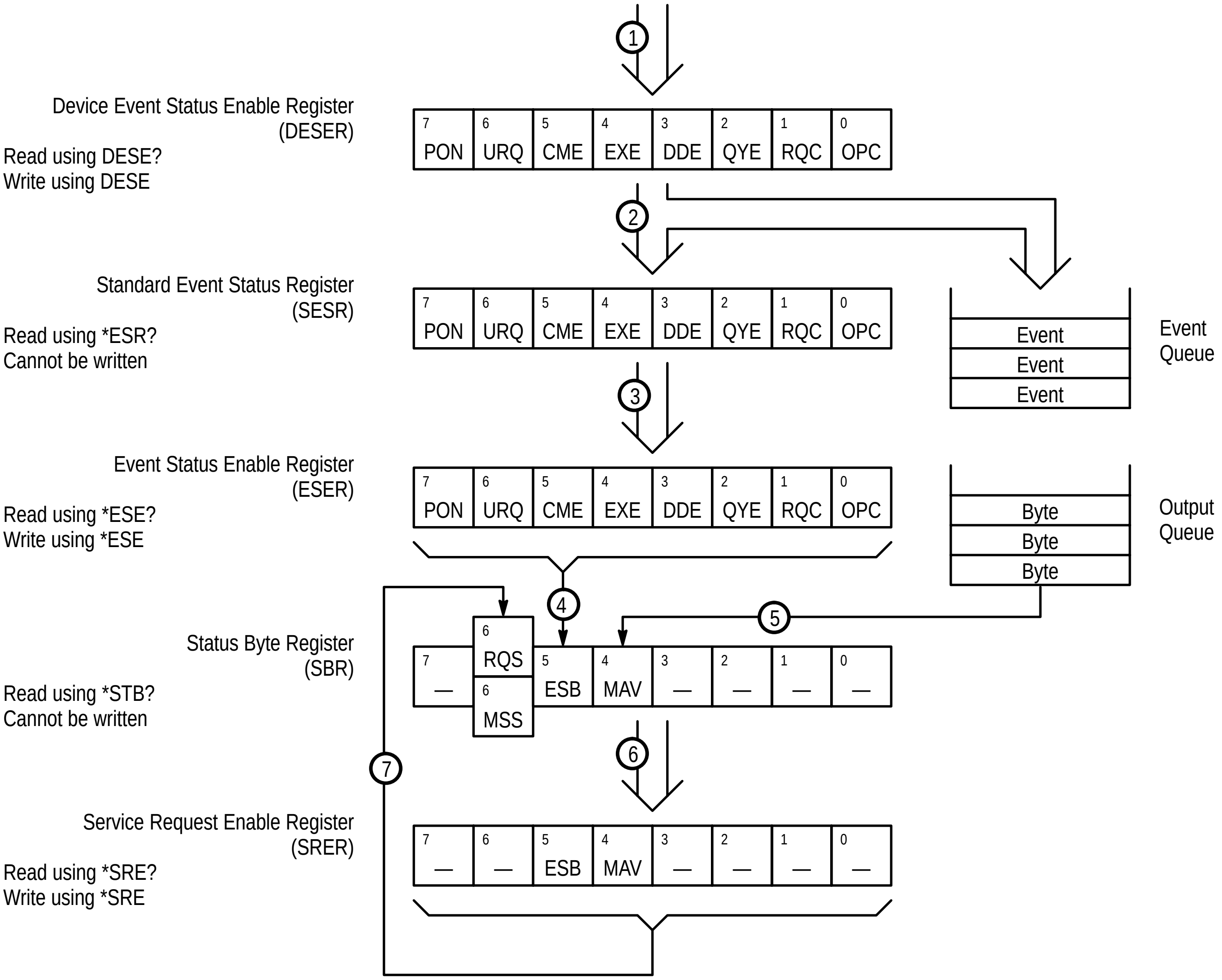

Device Event Status Enable Register (DESER)
Read using DESE?
Write using DESE
Standard Event Status Register (SESR)
Read using *ESR?
Cannot be written
Event Status Enable Register (ESER)
Read using *ESE?
Write using *ESE
Status Byte Register (SBR)
Read using *STB?
Cannot be written
Service Request Enable Register (SRER)
Read using *SRE?
Write using *SRE
7 PON 6 URQ 5 CME 4 EXE 3 DDE 2 QYE 1 RQC 0 OPC
7 — 6 RQS 6 MSS 5 ESB 4 MAV 3 — 2 — 1 — 0 —
7 — 6 — 5 ESB 4 MAV 3 — 2 — 1 — 0 —
Event
Event
Event
Event Queue
Byte
Byte
Byte
Output Queue
1
2
3
4
5
6
7

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 logic scope, some commands initiate processes requiring additional 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 logic scope 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 (the first operation must be completed before the next one is initiated). The status and event reporting system of the logic scope provides this capability.

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:GROUP1 ON
GROUP1:FIRST 1

/* Take amplitude measurement on acquired data */
MEASUREMENT:IMMED:VALUE?
```

The acquisition of the waveform requires extended processing time. It may not finish before the logic scope takes an amplitude measurement (See Figure 3–7). This can result in an incorrect amplitude value.

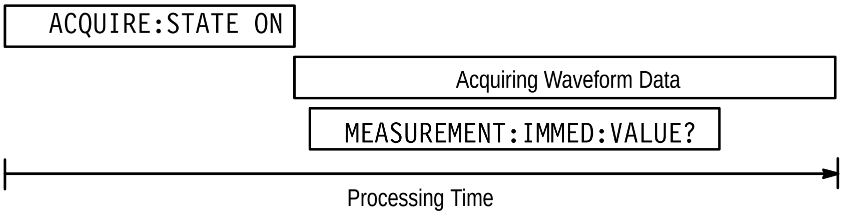


Figure 3–7: Command Processing Without Using Synchronization

To ensure the logic scope completes waveform acquisition before taking the measurement on the acquired data, you can synchronize the program. Figure 3–8 shows the desired processing sequence.

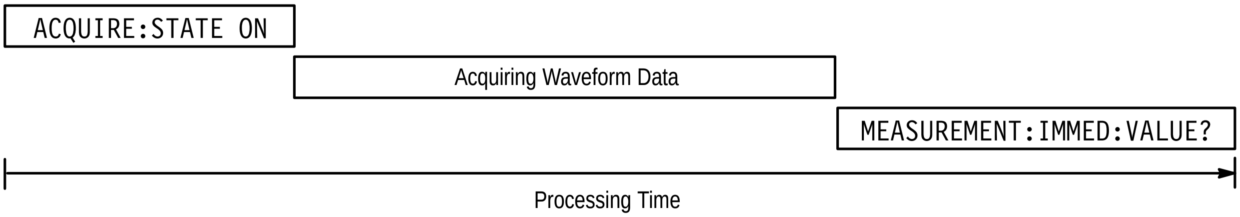


Figure 3–8: Processing Sequence With Synchronization

You can use four commands to synchronize the operation of the logic scope 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:GROUP1 ON
GROUP1:FIRST 1
HORIZONTAL:RECORDLENGTH 500
ACQUIRE:MODE NORMAL 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 logic scope 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 logic scope input buffer, but the commands will not be processed by the logic scope 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 determine whether the logic scope 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:GROUP1 ON
GROUP1:FIRST 1
HORIZONTAL:RECORDLENGTH 500
ACQUIRE:MODE NORMAL 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:GROUP1 ON
GROUP1:FIRST 1
HORIZONTAL:RECORDLENGTH 500
ACQUIRE:MODE NORMAL 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:GROUP1 ON
GROUP1:FIRST 1
HORIZONTAL:RECORDLENGTH 500
```

```
ACQUIRE:MODE NORMAL 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

The Controller 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 time-out 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:GROUP1 ON
GROUP1:FIRST 1
HORIZONTAL:RECORDLENGTH 500
ACQUIRE:MODE NORMAL SAMPLE
ACQUIRE:STOPAFTER SEQUENCE

/* Acquire waveform data */
ACQUIRE:STATE ON

/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1

/* Wait until the acquisition is complete before taking the measurement */
*OPC?
Wait for read from Output Queue.

/* Take amplitude measurement on acquired data */
MEASUREMENT:IMMED:VALUE?
```

This is the simplest approach. It requires no status handling or loops. However, you must set the controller time-out for longer than the acquisition operation.

Messages

Tables 3–3 through 3–9 list all the programming interface messages the logic scope generates in response to commands and queries.

For most messages, a secondary message from the logic scope gives more detail about the cause of the error or the meaning of the message. This message is part of the message string, and is separated from the main message by a semicolon.

Each message is the result of an event. Each type of event sets a specific bit in the SESR, and is controlled by the equivalent bit in the DESER. Thus, each message is associated with a specific SESR bit. In the message tables that follow, the associated SESR bit is specified in the table title, with exceptions noted with the error message text.

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

Table 3–3: No Event Messages

Code	Message
0	No events to report – queue empty
1	No events to report – new events pending *ESR?

Table 3–4 shows the error messages generated by improper command syntax. Check that the command is properly formed and that it follows the rules in the Command Syntax chapter starting on page 2–1.

Table 3–4: Command Error Messages — CME Bit 5

Code	Message
100	Command error
101	Invalid character
102	Syntax error
103	Invalid separator
104	Data type error
105	GET not allowed
106	Invalid program data separator
108	Parameter not allowed
109	Missing parameter
110	Command header error
111	Header separator error

Table 3–4: Command Error Messages — CME Bit 5 (Cont.)

Code	Message
112	Program mnemonic too long
113	Undefined header
118	Query not allowed
120	Numeric data error
121	Invalid character in number
123	Exponent too large
124	Too many digits
128	Numeric data not allowed
130	Suffix error
131	Invalid suffix
134	Suffix too long
138	Suffix not allowed
140	Character data error
141	Invalid character data
144	Character data too long
148	Character data not allowed
150	String data error
151	Invalid string data
152	String data too long
158	String data not allowed
160	Block data error
161	Invalid block data
168	Block data not allowed
170	Expression error
171	Invalid expression
178	Expression data not allowed
180	Alias error
181	Invalid outside alias definition
183	Invalid inside alias definition
184	Command in alias requires more/fewer parameters

Table 3–5 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
202	Settings lost due to rtl
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
250	Mass storage error
251	Missing mass storage
252	Missing media
253	Corrupt media
254	Media full
255	Directory full
256	File name not found
257	File name error
258	Media protected
260	Expression error
261	Math error in expression
2200	Measurement error, Measurement system error
2201	Measurement error, Zero period
2202	Measurement error, No period found
2203	Measurement error, No period, second waveform
2204	Measurement error, Low signal amplitude
2205	Measurement error, Low amplitude, second waveform
2206	Measurement error, Invalid gate
2207	Measurement error, Measurement overflow
2208	Measurement error, Waveform does not cross Mid Ref
2209	Measurement error, No second Mid Ref crossing

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
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
2250	Calibration error, Strobe delay meas failed
2251	Calibration error, HLUT entry out of bounds
2252	Calibration error, HLUT corrections not working

Table 3–5: Execution Error Messages — EXE Bit 4 (Cont.)

Code	Message
2253	Calibration error, Excessive amplitude on DC measurement
2254	Calibration error, Excessive offset on DC measurement
2255	Calibration error, Incorrect forward gain at sampler
2256	Calibration error, VLUT underrange cannot be calculated
2257	Calibration error, VLUT overrange cannot be calculated
2260	Calibration error
2261	Calibration error, VCO coarse DAC out of bounds
2262	Calibration error, VCO coarse DAC nonfunctional
2263	Calibration error, VCO coarse DAC did not converge
2264	Calibration error, VCO fine DAC nonfunctional
2265	Calibration error, VCO fine DAC did not converge
2266	Calibration error, VCO tcgain DAC out of bounds
2267	Calibration error, VCO frequency measurement failed
2268	Calibration error, Diagnostic A/D timed out
2269	Calibration error, VCO wrap delay out of bounds
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
2302	Cursor
2311	Group requested has not been selected, or has been deleted

Table 3–6 lists the device errors that can occur during logic scope operation. These errors may indicate the logic scope 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 3–7 lists the system event messages. These messages are generated whenever certain system conditions occur.

Table 3–7: System Event Messages

Code	Message
400	Query event
401	Power on (PON bit 7 set)
402	Operation complete (OPC bit 0 set)
403	User request (URQ bit 6 set)
404	Power fail (DDE bit 3 set)
405	Request control
410	Query INTERRUPTED (QYE bit 2 set)
420	Query UNTERMINATED (QYE bit 2 set)
430	Query DEADLOCKED (QYE bit 2 set)
440	Query UNTERMINATED after indefinite response (QYE bit 2 set)
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)
454	Right menu button #5 pushed (URQ bit 6 set)
460	Bottom menu button #1 pushed (URQ bit 6 set)

Table 3–7: System Event Messages (Cont.)

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

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

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
547	Measurement warning, Clipping positive/negative
548	Measurement warning, Clipping positive
549	Measurement warning, Clipping negative

Table 3–9 shows internal errors that indicate an internal fault in the logic scope.

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 logic scope from the GPIB interface. The disks that come with this manual contain listings for these programs written in 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.)

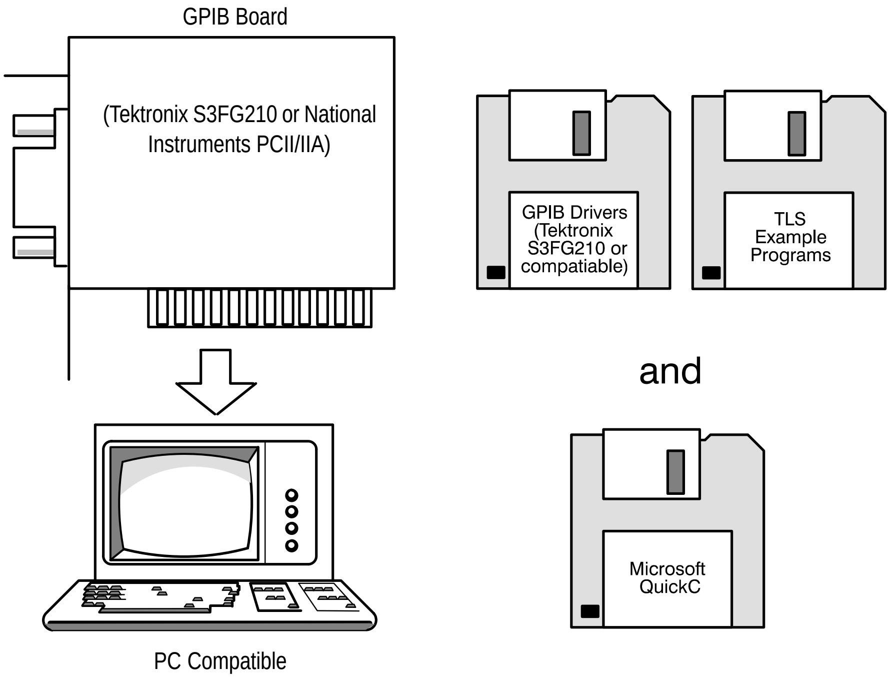

GPIB Board
(Tektronix S3FG210 or National Instruments PCII/IIA)
PC Compatible
GPIB Drivers (Tektronix S3FG210 or compatiable)
TLS Example Programs
and
Microsoft QuickC

Figure 4–1: Equipment Needed to Run the Example Programs

All example programs assume that the GPIB system recognizes the logic scope as DEV1 and the PC (controller) as GPIB0. You can use the `IBCONF.EXE` program to assign these names.

The example software includes the following programs:

- `MEAS` — measures waveform parameters from a logic scope.
- `COMM` — shows communication between a controller and a logic scope.
- `GETGRP` — reads a waveform from a logic scope and stores it in a file.

- CURSOR — uses cursors to measure waveform parameters.
- TL — a talker-listener program.

Compiling and Executing the Example Programs

The example programs disk contains programs written in Microsoft QuickC 2.5. The disk contains the following files.

- *Executable files* of the programs, which are stored in the directory *Programs.*
- *Source files* of the programs, which are in the directory *Sources.*
- A *Readme file* in each directory, which explains how to proceed.
- *Make files* for linking and compiling source files. These sample Make files are also stored in the Sources directory. Their file names have the suffix .MAK.

If you wish to develop code, you will need to use files that come with the GPIB system. Specifically, the QuickC programs use the files 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.*

Running Your Example Programs

To run an example program from the Programs directory, perform the following:

1. Install the Tektronix S3FG210, or National Instruments GPIB-PCII/IIA, GPIB board and drivers in your PC.
2. Identify the GPIB device as DEV1. You can use the IBCONF.EXE program to do this task.
3. Copy the files from the TLS Example Programs disk to your hard disk. To create a special directory (assumed to be named “examples”) on your hard disk (assumed to be drive C) and to copy the example programs (assumed to be in drive B), make drive C your current directory and type:

```
mkdir examples
cd examples
copy B:\quick-c\*.* .
```

4. Run the program by typing the program name.

To run `meas.exe`, type:	`meas`
To run `comm.exe`, type:	`comm`
To run `getgrp.exe`, type:	`getgrp`
To run `cursor.exe`, type:	`cursor`
To run `tl.exe`, type:	`tl`

Compiling And Linking Your Example QuickC Programs

To make an executable for any example from the Sources directory, perform the following steps:

1. Install QuickC. Select the LARGE memory model. Be sure to set up your path so DOS can access the QuickC directory.

2. Install the Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB board and drivers to your hard disk (assumed to be drive C).

3. Identify the GPIB device as DEV1. You can use the `IBCONF.EXE` program to do this task.

4. Copy the files from the examples disk to your hard disk. For example, to create a special directory (assumed to be named “examples”) on your hard disk (assumed to be drive C) and to copy the example programs (assumed to be in drive B), make drive C your current drive and type:

```
mkdir examples
cd examples
copy B:\quick-c\*.* .
```

5. Copy the files `DECL.H` and `MCIB.OBJ` from your Tektronix S3FG210 (National Instruments GPIB-PCII/IIA) GPIB drivers directory to the directory you created in step 4. For example, to copy the GPIB drivers to the current directory (assumed to be in C:\examples from step 4), type:

```
copy \gpib-pc\decl.h .
copy \gpib-pc\mcib.obj .
```

6. To compile and link your TLS sample C programs, type the command associated with the example program you wish to compile and link:

To compile and link `meas.c`, type:	`nmake meas.mak`
To compile and link `comm.c`, type:	`nmake comm.mak`
To compile and link `getgrp.c`, type:	`nmake getgrp.mak`
To compile and link cursor.c, type:	`nmake cursor.mak`

- To compile and link tl.c, type: `nmake tl.mak`

7. Run the program by typing the program name.

   - To run meas, type: `meas`
   - To run comm, type: `comm`
   - To run getgrp, type: `getgrp`
   - To run cursor, type: `cursor`
   - To run tl, type: `tl`

Appendix A: Character Charts

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

Table A–1: Logic Scope 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	⏚ 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: 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** 55 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
AUTO
AUTOSet
AUXiliary
AVErage
BACKWards
BASe
BAud
BANdwidth
BAUd
BELl
BIN
BIT_Nr
BMP
BMPCOLOR
BN_Fmt
BOLd
BOTTOM1
BOTTOM2
BOTTOM3
BOTTOM4
BOTTOM5
BOTTOM6
BOTTOM7
BOX
BURst
BUSY
BY
BYCONTents
BYT_Nr
BYT_Or
CATalog
CARea
CENtronics
CHKsm0
CH1
CH2
CH3
CH4
CLAss
CLEar
CLEARMenu
CLEARSNapshot
CLEARSpool
CLOCK
CMEan
COLLision
COLOr
COMpare
CONTRast
CONTROl
COPy
COUNt
COUPling
CPU
CRMs
CROSSHair
CRVchk
CURRent
CURSor
CURSOR1
CURSOR2
CURVe
CUSTom
CWD
DATa
DATE
DC
DEFAult
DEFIne
DELay
DELAYEd
DELEte
DELTa
DESE
DESKJet
DESTination
DIAg
DIR
DIRection
DISplay
DOTs
DPU411
DPU412
ECL
EDGE
EDGE1
EDGE2
EITher
ENCdg
ENV
ENVelope
EPSColor
EPSImage
EPSMono
EPSOn
EVEN
EVENT
EVENTS
EVENTSTime
EVMsg
EVQty
EXECute
FACtory
FALL
FALSe
FASTframe
FIELD
FIELDrate
FIELDS
FIELDS
FIFty
FILEName
FILESystem
FILTer
FITtoscreen
FLAg
FLEXformat
FORCe
FORMat
FORWards
FPAnel
FRAme
FREE
FREESpace
FREQuency
FULl
FUNCtion
GATing
GLItch
GND
GPIb
GRAticule
GRId
HARDCopy
HARDFlagging
HBArs
HDELTA
HDR
HDTV
HEADer
HERtz
HFRej
HIGH
HIGHLimit
HIRes
HIStogram
HOLdoff
HORizontal
HPGl
HPOS1
HPOS2
HUNdred
ID
IMMed
INDependent
IMPedance
INFInite
INFPersist
INIT
INPut
INTENSIFied
INTENSITy
INTERLeaf
IRE

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
MKDir
MODe
MONo
MORELimit
MOREThan
MSB
NAMe
NANd
NDUty
NEGAtive
NEGSynchwidth
NEWpass
NOISErej
NONe
NOR
NORMal
NOVershoot
NR_Pt
NTSc
NUMACq
NUMAVg
NUMEnv
NUMERic
NWIdth
ODD
OFF
OFFSet
ON
OR
OPTion
OVERAll
OVERWrite
PAIred
PAL
PALEtte
PARIty
PASSWord
PATtern
PCX
PCXCOLOR
PDUTy
PEAKdetect
PERCent
PERIod
PERSistence
PHAse
PK2pk
POLarity
PORT
PORTRait
POSition
POSITION1
POSITION2
POSITIVe
POVershoot
PRInt
PRObe
PROCessing
PT_Fmt
PT_Off
PULse
PWIdth
RATE1
RATE2
RATE3
RATE4
RECAll
RECOrdlength
REF1
REF2
REF3
REF4
REFLevel
REFSelect
REGular
REJect
REM
REName
REPEt
RESet
RESETAll
RESUlt
RI
RIBinary
RIGHT1
RIGHT2
RIGHT3
RIGHT4
RIGHT5
RISE
RLE
RMDir
RMS
RP
RPBinary
RS232
RUN
RUNSAfter
RUNSTop
RUNT
SAMple
SAVe
SCAle
SCAN
SCROLLBAR
SCROLLTEXT
SECAm
SECdiv
SECOnds
SELect
SEQuence
SET
SETLevel
SETUp
SHORT
SHOW
SINX
SLOpe
SNAp
SNAPShot
SOFTFlagging
SOUrce
SOURCE1
SOURCE2
SPECTral
SRIbinary
SRPbinary
STANdard
STARt
STATE
STOP
STOPAfter
STOPBits
STORe
STYle
SYNc
TARget
TEKSecure
TEMPErature
TEMPLate
TEXt
THInkjet
THReshold
TIFf
TIMe
TITLe
TOLerance
TRACk
TRIGAfter
TRIGBar
TRIGger
TRIGT
TRUe
TTL
TWEnty
TYPe
UNIts
UNLock
VALue
VARpersist
VBArs
VDELTA
VECtors
VERBose
VERTical
VIDeo
VOLts
V1STArttime
V1STOptime
V2STArttime
V2STOptime
WAVEform
WAVFrm
WFId
WFMPre
WHEn
WIDth
X
XINcr
XMUlt
XOFf
XUNit
XY
XZEro
Y
YMUlt
YOFf
YT
YUNit
YZEro
ZMUlt
ZOFf
ZONe
ZOOm
ZUNit
ZZEro

Appendix C: Internal Specifications

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

GPIB Function Subsets

The logic scope supports many GPIB function subsets, as listed below. Some of the listings describe subsets that the logic scope does not support.

- SH1 (Source Handshake). The logic scope can transmit multiline messages across the GPIB.
- AH1 (Acceptor Handshake). The logic scope can receive multiline messages across the GPIB.
- T5 (Talker). The logic scope 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 logic scope has talk-only capability for hardcopy operation.
- L4 (Listener). The logic scope becomes a listener when its listen address is sent with the ATN (Attention) line asserted. The logic scope does not have listen-only capability.
- SR1 (Service Request). The logic scope asserts an SRQ (Service Request) line to notify the controller when it requires service.
- RL1 (Remote/Local). The logic scope responds to both the GTL (Go To Local) and LLO (Local Lock Out) interface messages.
- PP0 (Parallel Poll). The logic scope has no parallel poll capability. It does not respond to the following interface messages: PPC, PPD, PPE, and PPU. The logic scope does not send out a status message when the ATN (Attention) and EOI (End or Identify) lines are asserted simultaneously.
- DC1 (Device Clear). The logic scope 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 logic scope responds to the GET (Group Execute Trigger) interface message.
- C0 (Controller). The logic scope cannot control other devices.

- E2 (Electrical). The logic scope uses tristate buffers to provide optimal high-speed data transfer.

Interface Messages

Table C–1 shows the standard interface messages that are supported by the logic scope.

Table C–1: Logic Scope 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

Recalling the factory setup establishes the initialization settings shown in Table D–1 to provide you with a known default setup for the logic scope.

Table D–1: Factory Initialization Defaults

Control	Changed by Factory Init to
Acquire mode	Sample
Acquire stop after	RUN/STOP button only
Acquire # of averages	16
Acquire # of envelopes	10
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 mode	Independent
Cursor function	Off
Cursor time units	Seconds
Delayed edge trigger level	0 V
Delayed edge trigger slope	Rising
Delayed edge trigger source	Channel 1
Delay trigger type	Edge
Delay time	16 ns
Delay events, triggerable after main	2
Delayed, delay by ...	Delay by Time
Delayed, time base mode	Delayed Runs After Main
Display clock	No Change
Display color – collision contrast	Off
Display color – map group colors	Reset to Factory Default Colors
Display color – map reference colors	Reset to Factory Default Colors
Display color – palette	Normal
Display color – palette colors	The colors of each palette are reset to factory hue, saturation, and lightness (HLS) values

Table D–1: Factory Initialization Defaults (Cont.)

Control	Changed by Factory Init to
Display color – persistence palette	Temperature
Display format	YT
Display graticule type	Full
Display intensity – text/graticule	60% (100%)
Display intensity – waveform	80% (100%)
Display interpolation filter	Sin(x)/x
Display style	Vectors
Display trigger bar style	Short
Display trigger “T”	On
Display variable persistence	500 ms
Edge trigger threshold (Main and Delayed)	0.0 V
Edge trigger slope (Main and Delayed)	Rising
Edge trigger source (Main and Delayed)	Channel 1
Group definition	First, last, and selected channel = Channel 1
Group selection	Group 1, containing Channel 1 only, is on; all other groups off
Group display mode	Analog
Group display thresholds	Vhigh min = 2.00 V; Vlow, max = 800 mV
Group display height	1.0 Divs
Horizontal – delay trigger position	50%
Horizontal – delay trigger record length	500 points (10 divs)
Horizontal – delay time/division	50 μs
Horizontal – main trigger position	50%
Horizontal – main trigger record length	500 points (10 divs)
Horizontal – main time/division	500 μs
Horizontal – time base	Main only
Main trigger holdoff	0%
Main trigger mode	Auto
Main trigger type	Edge
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)

Table D–1: Factory Initialization Defaults (Cont.)

Control	Changed by Factory Init to
Measure Low Ref	10% and 0 V (units)
Measure Mid Ref	50% and 0 V (units)
Measure Mid2 Ref	50% and 0 V (units)
Measure Setup or Hold Clock Source	Channel 1 (Ch1)
Measure Setup or Hold Clock Edge	Rising and forward searching
Pattern, Sequence, and State trigger patterns (Main and Delayed)	Ch 1 = H (HIGH); all others = X (don't care)
Pattern and Sequence trigger time qualification type (Main and Delayed)	Time Limit
Pattern and Sequence trigger time qualifier, time-limit qualified (Main and Delayed)	TRUE for > Time
Pattern and Sequence trigger time setting, time-limit qualified (Main and Delayed)	2.0 ns
Pattern and Sequence trigger upper limit time, time-range qualified (Main and Delayed)	2.0 ns
Pattern and Sequence trigger lower limit time, time-range qualified (Main and Delayed)	2.0 ns
Pattern and Sequence trigger time qualifier, time range qualified (Main and Delayed)	In Range
Pattern, Sequence, and State trigger logic definition	AND
Pattern, Sequence, and State trigger thresholds (Main and Delayed)	Channels 1-16 = 0V; AUX = 1.5 V
Pulse glitch trigger polarity (Main and Delayed)	Positive
Pulse glitch trigger when ... (Main and Delayed)	True < Time
Pulse trigger class (Main and Delayed)	Glitch
Pulse trigger glitch width (Main and Delayed)	2.0 ns
Pulse trigger glitch width (Main and Delayed)	Upper limit = 2.0 ns
Pulse trigger threshold (Main and Delayed)	0.0 V
Pulse trigger source — Glitch and width (Main and Delayed)	Channel 1 (Ch1)
Pulse width trigger when ... (Main and Delayed)	Within limits
Pulse width upper limit (Main and Delayed)	2.0 ns
Pulse width lower limit (Main and Delayed)	2.0 ns
Pulse width trigger polarity (Main and Delayed)	Positive

Table D–1: Factory Initialization Defaults (Cont.)

Control	Changed by Factory Init to
Saved setups	No change
Saved waveforms	No change
Vertical bandwidth (all groups)	Full
Vertical offset (all groups)	0 V
Vertical position (all groups)	0 divs.
Vertical volts/div. (all groups)	100 mV/div.
Zoom horizontal (all groups)	1.0X
Zoom horizontal lock	All
Zoom horizontal position (all groups)	50% = 0.5 (the middle of the display)
Zoom state	Off
Zoom vertical (all groups)	1.0X
Zoom vertical position (all groups)	0 divs.

Glossary

ASCII

Acronym for the American Standard Code for Information Interchange. Controllers transmit commands to the logic scope using ASCII character encoding.

Address

A 7-bit code that identifies an instrument on the communication bus. The logic scope 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 logic scope.

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).

Real-time sampling

A sampling mode where the logic scope samples fast enough to completely fill a waveform record from a single trigger event. Use real-time sampling to capture single-shot or transient events.

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.

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