MSO2000B, DPO2000B, MSO2000 and DPO2000 Series Oscilloscopes

Programmer Manual

077-0738-00

Tektronix

MSO2000B, DPO2000B, MSO2000 and DPO2000 Series Oscilloscopes

Programmer Manual

Revision A

www.tektronix.com

077-0738-00

Tektronix

Copyright © Tektronix. All rights reserved. Licensed software products are owned by Tektronix or its subsidiaries or suppliers, and are protected by national copyright laws and international treaty provisions.

Tektronix products are covered by U.S. and foreign patents, issued and pending. Information in this publication supersedes that in all previously published material. Specifications and price change privileges reserved.

TEKTRONIX and TEK are registered trademarks of Tektronix, Inc.

Contacting Tektronix

Tektronix, Inc.
14150 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Table of Contents

Getting Started 1-1
Setting Up Remote Communications Hardware 1-2
Connecting via Ethernet 1-2
Connecting via USB 1-3
Connecting via GPIB 1-4
Setting Up Remote Communications Software 1-5
Using TekVISA 1-6
Using Tektronix e*Scope Software 1-7
Documentation 1-9
Command Syntax 2-1
Command and Query Structure 2-1
Clearing the oscilloscope 2-4
Command Entry 2-4
Constructed Mnemonics 2-6
Argument Types 2-7
Command Groups 2-11
Acquisition Command Group 2-11
Alias Command Group 2-12
Bus Command Group 2-13
Calibration and Diagnostic Command Group 2-16
Cursor Command Group 2-17
Display Command Group 2-18
Ethernet Command Group 2-19
File System Command Group 2-20
FilterVu Command Group 2-21
Hard Copy Command Group 2-22
Horizontal Command Group 2-23
Mark Command Group 2-23
Math Command Group 2-25
Measurement Command Group 2-26
Miscellaneous Command Group 2-29
PictBridge Command Group 2-30
Save and Recall Command Group 2-31
Search Command Group 2-33
Status and Error Command Group 2-37
Trigger Command Group 2-38
Vertical Command Group 2-47
Waveform Transfer Command Group 2-50
Zoom Command Group 2-57

Commands Listed in Alphabetical Order 2-59
Status and Events 3-1
Registers 3-1
Queues 3-4
Event Handling Sequence 3-5
Synchronization Methods 3-7
Messages 3-12
Appendix A: Character Set A-1
Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments B-1
Appendix C: Reserved Words C-1
Appendix D: Application Module-enabled Commands D-1
Appendix E: Search and Trigger Command Sequence Examples E-1
Example 1: Single Threshold Edge Search E-1
Example 2: Single Threshold Edge Trigger E-1
Example 3: Dual Threshold Runt Search E-2
Example 4: Single Threshold Logic Search on Three Waveforms E-2
Index

Getting Started

This manual explains the use of commands for remotely controlling your oscilloscope. With this information, you can write computer programs to perform functions, such as setting the front-panel controls, taking measurements, performing statistical calculations, and exporting data for use in other programs.

This manual describes commands for the following models:

Table 1-1:

Model	Bandwidth	Number of Analog Channels	Sample Rate	Record Length, all ch.	Wfm. Capture Rate
MSO2024B, MSO2024	200 MHz	4	1 GS/s	1 M pts.	5,000
MSO2022B	200 MHz	2	1 GS/s	1 M pts.	5,000
MSO2014B, MSO2014	100 MHz	4	1 GS/s	1 M pts.	5,000
MSO2012B, MSO2012	100 MHz	2	1 GS/s	1 M pts.	5,000
MSO2004B	70 MHz	4	1 GS/s	1 M pts.	5,000
MSO2002B	70 MHz	2	1 GS/s	1 M pts.	5,000
DPO2024B, DPO2024	200 MHz	4	1 GS/s	1 M pts.	5,000
DPO2022B	200 MHz	2	1 GS/s	1 M pts.	5,000
DPO2014B, DPO2014	100 MHz	4	1 GS/s	1 M pts.	5,000
DPO2012B, DPO2012	100 MHz	2	1 GS/s	1 M pts.	5,000
DPO2004B	70 MHz	4	1 GS/s	1 M pts.	5,000
DPO2002B	70 MHz	2	1 GS/s	1 M pts.	5,000

New in the Programmer Manual

The following major changes were made to this version of the programmer manual (077-0738-00):

Added 6 new oscilloscope models:

MSO2002B – 70 MHz, 2 channel

MSO2004B – 70 MHz, 4 channel

MSO2022B – 200 MHz, 2 channel

DPO2002B – 70 MHz, 2 channel

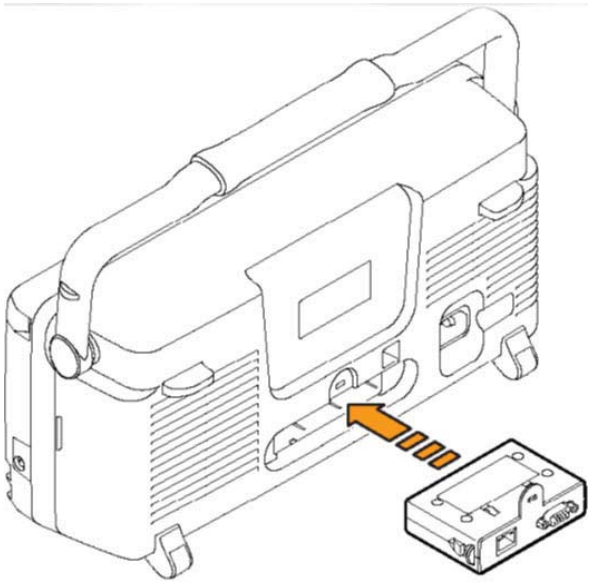
DPO2004B – 70 MHz, 4 channel

DPO2022B – 200 MHz, 2 channel

Setting Up Remote Communications Hardware

You can remotely communicate between your oscilloscope and PC via Ethernet, USB, or GPIB cables.

NOTE. *In order to communicate via an Ethernet cable, you need to install an optional DPO2CONN Connectivity Module into the back of the instrument. This module includes both Ethernet and VGA video monitor ports.*



Connecting via Ethernet

If your PC is connected to a local area network, you can use an Ethernet cable to connect your oscilloscope to the same network, and then use software to remotely control the oscilloscope via the PC. First, you'll need to acquire an optional DPO2CONN Connectivity Module, which provides Ethernet and video out ports for your oscilloscope (search on www.tektronix.com). Then, simply plug one end of the Ethernet cable into the Ethernet port (RJ-45 connector), and the other end into your network connection.

NOTE. *You can connect an MSO/DPO2000B oscilloscope only to a 10/100 Base-T local area network.*

To view or change the Ethernet settings on your oscilloscope, do the following:

1. On the front panel, push **Utility**.
2. Push **Utility Page**.
3. Select **I/O** with the Multipurpose knob.
4. Push **Ethernet Network Settings**.
5. If you are on a DHCP Ethernet network and using a through cable, on the side menu set DHCP/BOOTP to **On**.
6. If you are using a cross-over cable, set DHCP/BOOTP to **Off**, and set a hard-coded TCPIP address.

Connecting via USB

You can connect your oscilloscope directly to a PC by using a high-speed USB cable, and then use software to remotely control the oscilloscope via the PC. Simply plug one end of the cable into the USB 2.0 high-speed device port on the rear panel of your oscilloscope, and the other end into a USB port on your computer.

This port requires that the cable connected from the port to the host computer meets the USB 2.0 specification for high speed connections. Typically, such cables should be 3 feet or shorter in length, but this is determined by the quality of the cable and, with higher quality cables, this length can be extended. (It is also dependent upon the drive capability of the host USB port to which the instrument is connected.) The use of high quality short cables is recommended to avoid USB connection problems.

Once the USB cable is connected, the system automatically configures itself. To verify that the USB is enabled:

1. On the front panel, push **Utility**.
2. Push **Utility Page**.
3. Select **I/O** with the Multipurpose knob.
4. Push **USB**, and verify that USB is enabled.
5. If USB is not enabled, push **Enabled** on the side menu.

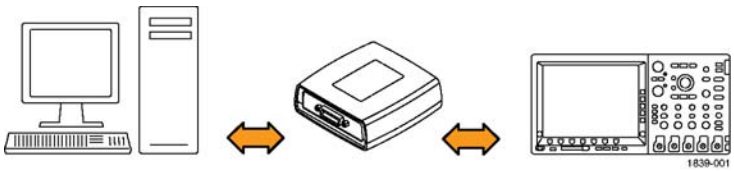
After connection, the host, with appropriate software, can list the oscilloscope as a USB device with the following parameters: (See Table 1-2.)

Table 1-2: USB Device Parameters

Parameter	Value
Manufacturer ID	0x0699 (decimal 1689)
Product ID	0x399 DPO2002B (decimal 921) 0x39A MSO2002B (decimal 922) 0x39B DPO2004B (decimal 923) 0x39C MSO2004B (decimal 924) 0x39D DPO2012B (decimal 925) 0x39E MSO2012B (decimal 926) 0x39F DPO2014B (decimal 927) 0x3A0 MSO2014B (decimal 928) 0x3A1 DPO2022B (decimal 929) 0x3A2 MSO2022B (decimal 930) 0x3A3 DPO2024B (decimal 931) 0x3A4 MSO2024B (decimal 932) 0x0372 DPO2012 0x0373 DPO2014 0x0374 DPO2024 0x0376 MSO2012 0x0377 MSO2014 0x0378 MSO2024
Serial number	Serial number
Manufacturer description	“Tektronix”
Interface description	“USBTMC-USB488”

Connecting via GPIB

The oscilloscope has a USB 2.0 high-speed device port to control the oscilloscope through USBTMC or GPIB with a TEK-USB-488 Adapter. The USBTMC protocol allows USB devices to communicate using IEEE488 style messages. This lets you run your GPIB software applications on USB hardware.



To use GPIB, start by connecting an appropriate USB cable to the USB 2.0 high-speed device port on the rear panel of your oscilloscope. Connect the other end to the TEK-USB-488 Adapter host port. Then connect a GPIB cable from the TEK-USB-488 Adapter to your PC.

Supply power to the Adapter in either of these two ways:

1. Use the optional 5 V_{DC} power adapter connected to the 5 V_{DC} power input on the Adapter.
2. Use an appropriate USB cable connected to a powered USB host port on your PC and the Device port on the TEK-USB-488 Adapter.

Before setting up the oscilloscope for remote communication using the electronic (physical) GPIB interface, you should familiarize yourself with the following GPIB requirements:

- A unique device address must be assigned to each device on the bus. No two devices can share the same device address.
- No more than 15 devices can be connected to any one line.
- Only one device should be connected for every 6 feet (2 meters) of cable used.
- No more than 65 feet (20 meters) of cable should be used to connect devices to a bus.
- At least two-thirds of the devices on the network should be powered on while using the network.
- Connect the devices on the network in a star or linear configuration. Do not use loop or parallel configurations.

To function correctly, your oscilloscope must have a unique device address. The default setting for the GPIB configuration is GPIB Address 1.

To change the GPIB address settings, do the following:

1. On the front panel, push **Utility**.
2. Push **Utility Page**.
3. Select **I/O** with the Multipurpose knob.
4. Push **GPIB**.
5. Enter the GPIB address on the side menu, using the multipurpose knob. This will set the GPIB address on an attached TEK-USB-488 Adapter

The oscilloscope is now set up for bidirectional communication with your PC.

Setting Up Remote Communications Software

Connect your oscilloscope directly to a computer to let the PC analyze your data, collect screen images, or to control the oscilloscope using a program of your own creation. You can connect using TekVISA drivers, or connect directly from any computer's web browser using Tektronix e*Scope Web-enabled tools.

NOTE. *The CD that your oscilloscope shipped with contains additional tools for efficient connectivity between your oscilloscope and your computer. These include toolbars that speed connectivity with Microsoft Excel and Word.*

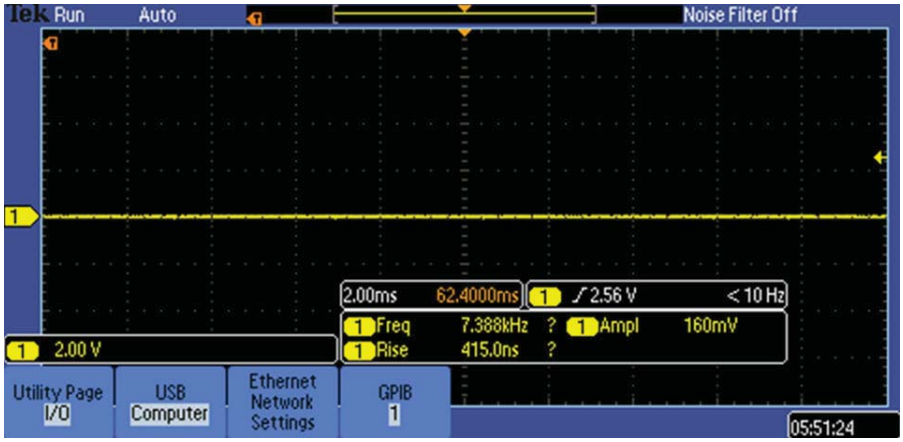
Using TekVISA

TekVISA lets you use your MS-Windows computer to acquire data from your oscilloscope for use in an analysis package that runs on your PC, such as Microsoft Excel, National Instruments LabVIEW, Tektronix OpenChoice Desktop software, or your own custom software. You can use a common communications connection, such as USB, Ethernet, or GPIB, to connect the computer to the oscilloscope. The TekVISA drivers are automatically installed by installing the OpenChoice Desktop software that came with your instrument's CD. You can also download the OpenChoice Desktop software from www.tektronix.com\downloads.

NOTE. *TekVISA cannot run if any other version of VISA drivers is installed.*

To set up communications between your oscilloscope and a computer running TekVISA drivers:

1. Install the Tektronix OpenChoice Desktop software package, either from the CD that came with your instrument or from the Tektronix website. This will automatically install the TekVISA drivers.
2. Connect the oscilloscope to your computer with the appropriate USB, Ethernet or GPIB cable. Cycle the power on the oscilloscope.
3. Push **Utility.**
4. Push **Utility Page.**
5. Turn multipurpose knob **a** and select **I/O.**



6. If you are using USB, the system sets itself up automatically for you, if USB is enabled. Check **USB** on the lower menu to be sure that USB is enabled. If it is not enabled, push **USB.** Then push **Connect to Computer** on the side menu.
7. To use Ethernet, push **Ethernet Network Settings** on the lower menu. Use the side menu buttons to adjust your network settings, as needed. For more information, see the e*Scope setup information below.
8. If you are using GPIB, push **GPIB**. Enter the GPIB address on the side menu, using multipurpose knob **a.** This will set the GPIB address on an attached TEK-USB-488 Adapter.
9. Run the application software on your computer.

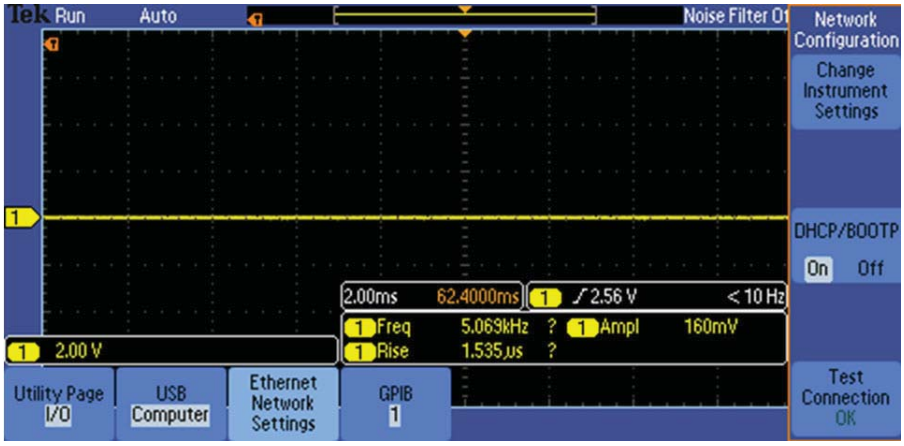
Using Tektronix e*Scope Software

Your oscilloscope contains a pre-installed remote control software package by Tektronix called e*Scope. You can use this to “talk” to a networked PC’s web browser so that you can view and control the oscilloscope wherever it is on your network.

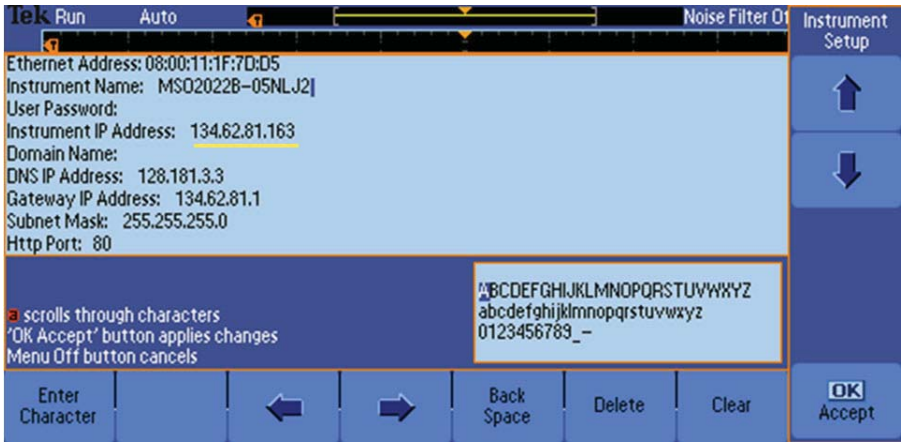
First, you’ll need to acquire an optional DPO2CONN Connectivity Module, which provides Ethernet and video out ports for your oscilloscope (search on www.tektronix.com).

To set up e*Scope communications between your oscilloscope and a networked computer:

1. With the DPO2CONN Connectivity Module installed, connect an Ethernet cable from the back of the oscilloscope to the same network as your computer.
2. Power up your oscilloscope and test the network connection:
   a. Push the **Utility** button, and then push **Utility Page** on the bottom menu.
   b. Turn multipurpose knob **a** to select **I/O**, and then push **Ethernet Network Settings** on the bottom menu.
   c. Push **Test Connection** on the side menu. The button should say **OK.**



3. Next, find your oscilloscope's IP address:
   a. Push **Change Instrument Settings** on the side menu to display the network parameters configured on your oscilloscope.
   b. Note down the Instrument IP address.



4. On the menu bar of your PC's web browser, type in the oscilloscope's Instrument IP address and press **Enter**.

5. You should now see the e*Scope screen on your PC and an image of your oscilloscope's display. You may use the menu items at the top to control your oscilloscope from your PC's browser.

Documentation

The following documents are available for download on the Manuals Finder Web site at *www.tektronix.com*:

MSO/DPO2000B Series User Manual. Information about installing and operating the oscilloscope.

MSO/DPO2000B Series Technical Reference. Oscilloscope specifications and a performance verification procedure.

TekVISA Programmer Manual. Description of TekVISA, the Tektronix implementation of the VISA Application Programming Interface (API). TekVISA is industry-compliant software for writing interoperable oscilloscope drivers in a variety of Application Development Environments (ADEs).

Command Syntax

You can control the operations and functions of the oscilloscope through the Ethernet port or the USB 2.0 device port using commands and queries. The related topics listed below describe the syntax of these commands and queries. The topics also describe the conventions that the oscilloscope uses to process them. See the *Command Groups* topic in the table of contents for a listing of the commands by command group, or use the index to locate a specific command.

Backus-Naur Form Notation

This documentation describes the commands and queries using Backus-Naur Form (BNF) notation. Refer to the following table for the symbols that are used.

Table 2-1: Symbols for Backus-Naur Form

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 called commands and queries). Commands modify oscilloscope settings or tell the oscilloscope to perform a specific action. Queries cause the oscilloscope to return data and status information.

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 at 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.

Messages

A command message is a command or query name followed by any information the oscilloscope needs to execute the command or query. Command messages may contain five element types, defined in the following table.

Table 2-2: Command Message Elements

Symbol	Meaning
<Header>	This is 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>	This is a header subfunction. Some command headers have only one mnemonic. If a command header has multiple mnemonics, a colon (:) character always separates them from each other.
<Argument>	This is a quantity, quality, restriction, or limit associated with the header. Some commands have no arguments while others have multiple arguments. A <space> separates arguments from the header. A <comma> separates arguments from each other.
<Comma>	A single comma is used between arguments of multiple-argument commands. Optionally, there may be white space characters before and after the comma.
<Space>	A white space character is used between a command header and the related argument. Optionally, a white space may consist of multiple white space characters.

Commands

Commands cause the oscilloscope to perform a specific function or change one of the 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 oscilloscope to return status or setting information. Queries have the structure:

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

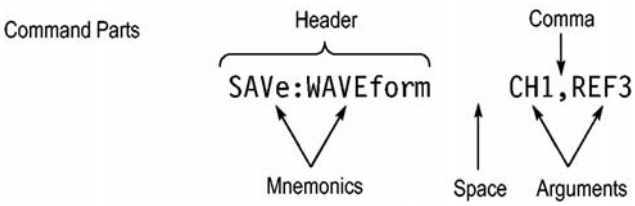
You can specify a query command at any level within the command tree unless otherwise noted. These branch queries return information about all the mnemonics below the specified branch or level.

Headers

Use the HEADer command to control whether the oscilloscope returns headers as part of the query response. If header is on, the query response returns command headers, then 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. The table below shows the difference in responses.

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

Query	Header Off	Header On
TIME?	14:30:00	:TIME “14:30:00”
ACQuire:NUMAVg?	100	:ACQUIRE:NUMAVG 100



Clearing the oscilloscope

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

Command Entry

The following rules apply when entering commands:

- You can enter commands in upper or lower case.
- You can precede any command with white space characters. White space characters include any combination of the ASCII control characters 00 through 09 and 0B through 20 hexadecimal (0 through 9 and 11 through 32 decimal).
- The oscilloscope ignores commands consisting of any combination of white space characters and line feeds.

Abbreviating

You can abbreviate many oscilloscope commands. Each command in this documentation shows the minimum acceptable abbreviations in capitals. For example, you can enter the command ACQuire:NUMAvg simply as ACQ:NUMA or acq:numa.

Abbreviation rules may change over time as new oscilloscope models are introduced. Thus, for the most robust code, use the full spelling.

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 with the VERBose command.

Concatenating

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

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

1. Separate completely different headers by a semicolon and by the beginning colon on all commands except the first one. For example, the commands `TRIGger:MODe NORMal` and `ACQuire:NUMAVg 8`, can be concatenated into the following single command:

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

2. If concatenated commands have headers that differ by only the last mnemonic, you can abbreviate the second command and eliminate the beginning colon. For example, you can concatenate the commands `ACQuire:MODe AVErage` and `ACQuire:NUMAVg 8` into a single command:

   `ACQuire:MODe AVErage; NUMAVg 8`

   The longer version works equally well:

   `ACQuire:MODe AVErage;:ACQuire:NUMAVg 8`

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

   `ACQuire:STATE 1;*OPC`

   Any commands that follow will be processed as if the star command was not there so the commands, `ACQuire:MODe ENVelope;*OPC;NUMAVg 8` will set the acquisition mode to envelope and set the number of acquisitions for averaging to 8.

4. When you concatenate queries, the responses to all the queries are concatenated into a single response message.

5. Set commands and queries may be concatenated in the same message. For example,

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

   is a valid message that sets the acquisition mode to sample. The message then queries the number of acquisitions for averaging and the acquisition state. Concatenated commands and queries are executed in the order received.

   Here are some invalid concatenations:

   `HORizontal:SCAle 400E-9;ACQuire:NUMAVg 8` (no colon before ACQuire)

   `DISPlay:GRAticule FULL;:*TRG` (colon before a star (*) command)

   `MATH:HORizontal:SCAle 1.0e-1;HORizontal:POSition 5.0el` (levels of the mnemonics are different; either remove the second use of `HORizontal:` or place `:MATH` in front of `HORizontal:POSition`)

Terminating

This documentation uses `<EOM>` (End of Message) to represent a message terminator.

Table 2-4: End of Message Terminator

Symbol	Meaning
<EOM>	Message terminator

The end-of-message terminator must be the END message (EOI asserted concurrently with the last data byte). The last data byte may be an ASCII line feed (LF) character.

This oscilloscope does not support ASCII LF only message termination. The oscilloscope always terminates outgoing messages with LF and EOI.

Constructed Mnemonics

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. For example, there is a CH1:POSition command, and there is also a CH2:POSition command. In the command descriptions, this list of choices is abbreviated as CH<x>.

Cursor Position Mnemonics

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

Table 2-5: Channel Mnemonics

Symbol	Meaning
CH<x>	A channel specifier; <x> is 1 through 4.

Table 2-6: Cursor Mnemonics

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

Math Specifier Mnemonics

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

Table 2-7: Math Specifier Mnemonics

Symbol	Meaning
Math<x>	A math waveform specifier; <x> is 1.

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.

Table 2-8: Measurement Specifier Mnemonics

Symbol	Meaning
MEAS<x>	A measurement specifier; <x> is 1 through 4.

Channel Mnemonics

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

Reference Waveform Mnemonics

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

Table 2-9: Reference Waveform Mnemonics

Symbol	Meaning
REF<x>	A reference waveform specifier; <x> 1 or 2. The MSO/DPO2000B and MSO/DPO2000 series provides only two REF waveforms regardless of whether the instrument is a 2 or 4 channel model.

Argument Types

Numeric

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

Table 2-10: Numeric Arguments

Symbol	Meaning
<NR1>	Signed integer value
<NR2>	Floating point value without an exponent
<NR3>	Floating point value with an exponent
<bin>	Digital data in binary format

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

Quoted String

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 ("). The following is an example of a quoted string: `"This is a quoted string"`. This documentation represents these arguments as follows:

Table 2-11: Quoted String Argument

Symbol	Meaning
<QString>	Quoted string of ASCII text

A quoted string can include any character defined in the 7-bit ASCII character set. Follow these rules when you use quoted strings:

1. Use the same type of quote character to open and close the string. For example: `"this is a valid string"`.
2. You can mix quotation marks within a string as long as you follow the previous rule. For example: `"this is an 'acceptable' string"`.
3. You can include a quote character within a string by repeating the quote. For example: `"here is a "" mark"`.
4. Strings can have upper or lower case characters.
5. If you use a GPIB network, you cannot terminate a quoted string with the END message before the closing delimiter.
6. A carriage return or line feed embedded in a quoted string does not terminate the string. The return is treated as another character in the string.
7. 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)

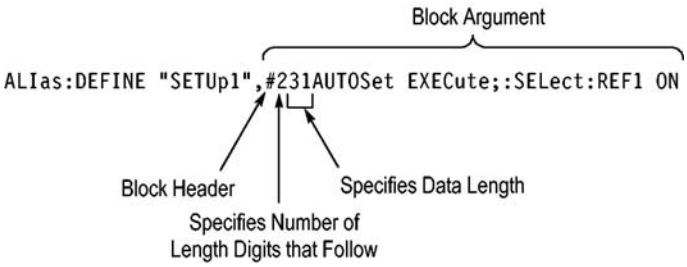
Block

Several oscilloscope commands use a block argument form, as defined in the table below.

Table 2-12: Block Argument

Symbol	Meaning
<NZDig>	A nonzero digit character in the range of 1–9
<Dig>	A digit character, in the range of 0–9
<DChar>	A character with the hexadecimal equivalent of 00 through FF (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 <NZDig> and <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 Number of
Length Digits that Follow
Specifies Data Length

Command Groups

This manual lists the MSO/DPO2000B and MSO/DPO2000 series IEEE488.2 commands in two ways. First, it presents them by functional groups. Then, it lists them alphabetically. The functional group list starts below. The alphabetical list provides detail on each command. (See page 2-59, *Commands Listed in Alphabetical Order*.)

Acquisition Command Group

Use the commands in the Acquisition Command Group to set up the modes and functions that control how the oscilloscope acquires signals input to the channels, and processes them into waveforms.

Using the commands in this group, you can do the following:

- Start and stop acquisitions.
- Control whether each waveform is simply acquired, averaged over successive acquisitions of that waveform.
- Set the controls or conditions that start and stop acquisitions.
- Control acquisition of channel waveforms.
- Set acquisition parameters.

Table 2-13: Acquisition Commands

Command	Description
ACQuire?	Returns acquisition parameters
ACQuire:MAXSamplerate?	Returns the maximum real-time sample rate
ACQuire:MODe	Sets or returns the acquisition mode
ACQuire:NUMACq?	Returns number of acquisitions that have occurred
ACQuire:NUMAVg	Sets or returns the number of acquisitions for an averaged waveform
ACQuire:STATE	Starts or stops the acquisition system
ACQuire:STOPAfter	Sets or returns whether the acquisition is continuous or single sequence

Alias Command Group

Use the Alias commands to define new commands as a sequence of standard commands. You may find this useful when repeatedly using the same commands to perform certain tasks like setting up measurements.

Aliases are similar to macros but do not include the capability to substitute parameters into alias bodies. The alias mechanism obeys the following rules:

- The alias name must consist of a valid IEEE488.2 message unit, which may not appear in a message preceded by a colon, comma, or a command or query program header.
- The alias name may not appear in a message followed by a colon, comma, or question mark.
- An alias name must be distinct from any keyword or keyword short form.
- An alias name cannot be redefined without first being deleted using one of the alias deletion functions.
- Alias names do not appear in response messages.

Table 2-14: Alias Commands

Command	Description
ALIas	Sets or returns the alias state
ALIas:CATalog?	Returns a list of the currently defined alias labels
ALIas:DEFine	Assigns a sequence of program messages to an alias label
ALIas:DELEte	Removes a specified alias
ALIas:DELEte:ALL	Deletes all existing aliases
ALIas:DELEte[:NAMe]	Removes a specified alias
ALIas[:STATE]	Sets or returns the alias state

Bus Command Group

Use the Bus commands when working with serial bus measurements.

- Install the DPO2EMBD application module when working with I²C or SPI bus signals.
- Install the DPO2AUTO module when working with CAN or LIN bus signals.
- Install the DPO2COMP module when working with RS232 bus signals.

Table 2-15: Bus Commands

Commands	Description
BUS	Returns the parameters for each bus
BUS:B<x>:CAN:BITRate	Sets or returns the bit rate for the CAN bus
BUS:B<x>:CAN:PRObe	Sets or returns the probing method used to probe the CAN bus
BUS:B<x>:CAN:SAMPLEpoint	Sets or returns the sample point (in %) to sample during each bit period
BUS:B<x>:CAN:SOUrce	Sets or returns the CAN data source
BUS:B<x>:DISplay:FORMAt	Sets the display format for the numerical information in the specified bus waveform
BUS:B<x>:I2C:ADDRess:RWINClude	Sets and returns whether the read/write bit is included in the address
BUS:B<x>:I2C{:CLOCK|:SCLK}:SOUrce	Sets or returns the I2C SCLK source
BUS:B<x>:I2C{:DATA|:SDATA}:SOUrce	Sets or returns the I2C SDATA source
BUS:B<x>:LABel	Sets or returns the waveform label for the specified bus
BUS:B<x>:LIN:BITRate	Sets or returns the bit rate for LIN
BUS:B<x>:LIN:IDFORmat	Sets or returns the LIN ID format
BUS:B<x>:LIN:POLARity	Sets or returns the LIN polarity
BUS:B<x>:LIN:SAMPLEpoint	Sets or returns the sample point (in %) at which to sample during each bit period
BUS:B<x>:LIN:SOUrce	Sets or returns the LIN data source
BUS:B<x>:LIN:STANDard	Sets or returns the LIN standard
BUS:B<x>:PARallel:BIT<x>:SOUrce	Sets or returns the parallel bit <x> source
BUS:B<x>:PARallel:CLOCK:EDGE	Sets or returns the parallel clock edge for bus <x>
BUS:B<x>:PARallel:CLOCK:ISCLOCKed	Sets or returns whether the parallel bus is clocked
BUS:B<x>:PARallel:CLOCK:SOUrce	Sets or returns the parallel bus<x> clock source
BUS:B<x>:PARallel:WIDth	Sets or returns the number of bits used for the width of the parallel bus <x>

Table 2-15: Bus Commands (cont.)

Commands	Description
BUS:B<x>:POSition	Sets or returns the position of the specified bus waveform
BUS:B<x>:RS232C:BITRate	Sets or returns the RS232 bit rate for the specified bus
BUS:B<x>:RS232C:DATABits	Sets or returns the number of bits for the data frame
BUS:B<x>:RS232C:DELIMiter	Sets or returns the RS232 delimiting value for a packet on the specified bus
BUS:B<x>:RS232C:DISplaymode	Sets or returns the display mode for the specified bus display and event table
BUS:B<x>:RS232C:PARity	Sets or returns parity for RS232 data
BUS:B<x>:RS232C:POLarity	Sets or returns the RS232C polarity for the specified bus
BUS:B<x>:RS232C:RX:SOUrce	Sets or returns the RS232 RX source
BUS:B<x>:RS232C:TX:SOUrce	Sets or returns the RS232 TX Source
BUS:B<x>:SPI{:CLOCK|:SCLK}:POLARity	Sets or returns the SPI SCLK polarity
BUS:B<x>:SPI{:CLOCK|:SCLK}:SOUrce	Sets or returns the SPI SCLK source
BUS:B<x>:SPI:DATA{:IN|:MISO}:POLARity	Sets or returns the SPI MISO polarity
BUS:B<x>:SPI:DATA{:IN|:MISO}:SOUrce	Sets or returns the SPI MISO source
BUS:B<x>:SPI:DATA{:OUT|:MOSI}: POLARity	Sets or returns the SPI MOSI polarity
BUS:B<x>:SPI:DATA{:OUT|:MOSI}:SOUrce	Sets or returns the SPI MOSI source
BUS:B<x>:SPI{:SELect|:SS}:POLARity	Sets or returns the SPI SS polarity
BUS:B<x>:SPI{:SELect|:SS}:SOUrce	Sets or returns the SPI SS source
BUS:B<x>:SPI:BITOrder	Sets or returns the bit order for the specified SPI bus
BUS:B<x>:SPI:DATA:SIZe	Sets or returns the number of bits per word for the specified SPI bus
BUS:B<x>:SPI:FRAMING	Sets or returns the type of SPI framing
BUS:B<x>:SPI:IDLETime	Sets or returns the SPI bus idle time in seconds for the specified SPI bus
BUS:B<x>:STATE	Turns the specified bus on and off
BUS:B<x>:TYPE	Sets or returns the specified bus type
BUS:LOWerthreshold:CH<x>	Sets or returns the lower threshold for each channel
BUS:THReshold:CH<x>	Sets or returns the threshold for a channel
BUS:UPPerthreshold:CH<x>	Sets or returns the upper threshold for each channel
BUS:THReshold:D<x>	Sets or returns the threshold for digital channel

Table 2-15: Bus Commands (cont.)

Commands	Description
SEARCH:SEARCH<x>:TRIGger:A:BUS?	Returns the serial search type
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:CONDition	Sets or returns the search condition for a LIN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue	Sets or returns the binary data string
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier	Sets or returns the LIN data qualifier
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:SIZe	Sets or returns the length of the data string in bytes
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:VALue	Sets or returns the binary data string used for a LIN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:ERRTYPE	Sets or returns the error type used for a LIN Search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue	Sets or returns the binary address string used for LIN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:CONDition	Sets or returns the trigger condition for a RS232 trigger
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe	Sets or returns the length of the data string for a RS232 RX trigger
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue	Sets or returns the binary data string for a RX RS232 trigger
TRIGger:A:BUS:B<x>:LIN:CONDition	Sets or returns the trigger condition for LIN
TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue	Sets or returns the binary data string to be used for LIN trigger
TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier	Sets or returns the LIN data qualifier
TRIGger:A:BUS:B<x>:LIN:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for LIN trigger
TRIGger:A:BUS:B<x>:LIN:DATa:VALue	Sets or returns the binary data string
TRIGger:A:BUS:B<x>:LIN:ERRTYPE	Sets or returns the error type
TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue	Sets or returns the binary address string used for LIN trigger
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe	Sets or returns the length of the data string for a RX RS232 trigger
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue	Sets or returns the binary data string for a RX RS232 trigger
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe	Sets or returns the length of the data string to be used for a TX RS232 Trigger
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue	Sets or returns the binary data string to be used for a TX RS232 trigger

Calibration and Diagnostic Command Group

The Calibration and Diagnostic commands provide information about the current state of oscilloscope calibration. They also initiate internal signal path calibration (SPC) or execute diagnostic tests. Commands that are specific to factory calibration are not described in this manual. They are described in the Service manual, available for download at www.tektronix.com.

Table 2-16: Calibration and Diagnostic Commands

Command	Description
*CAL?	Instructs the oscilloscope to perform self-calibration and returns the oscilloscope self calibration status
CALibrate:FACtory	Provides the controls for starting and stopping the factory calibration process
CALibrate:FACtory:STATus?	Returns the factory calibration status value saved in nonvolatile memory
CALibrate:INTERNal	Starts a signal path compensation
CALibrate:INTERNal:STARt	Starts the internal signal path calibration
CALibrate:INTERNal:STATus?	Returns the current status of the internal signal path calibration
CALibrate:RESults?	Returns the status of all calibration subsystems without performing an SPC operation
CALibrate:RESults:FACtory?	Returns the status of internal and factory calibration
CALibrate:RESults:SPC?	Returns the results of the last SPC operation
CALibrate:TEMPerature?	Returns 0 as MSO/DPO2000/B series does not support recording of oscilloscope temperature
DIAg:LOOP:OPTion	Sets the self-test loop option
DIAg:LOOP:OPTion:NTIMes	Sets the self-test loop option to run N times
DIAg:LOOP:STOP	Stops the self-test at the end of the current loop
DIAg:RESUlt:FLAg?	Returns the pass/fail status from the last self-test sequence execution
DIAg:RESUlt:LOG?	Returns the internal results log from the last self-test sequence execution
DIAg:SELect:<function>	Selects one of the available self-test areas
DIAg:STATE	Sets the oscilloscope operating state
DIAg:SELect	Runs self tests on the specified system subsystem

Cursor Command Group

Use the commands in the Cursor Command Group to control the cursor display and readout. You can use these commands to control the setups for cursor 1 and cursor 2, such as cursor position.

You can also use the commands to select one of the following cursor functions:

- Off. Turns off the display of all cursors.
- Waveform Cursors. Consists of two cursors. Waveform cursors enable you to conveniently measure waveform amplitude and time.
- Screen Cursors. Consists of two pairs of independent horizontal and vertical cursors. You can use these cursors to indicate an arbitrary position within the waveform display area.

Table 2-17: Cursor Commands

Command	Description
CURSor?	Returns cursor settings
CURSor:FUNCtion	Sets or returns the cursor type
CURSor:HBArs?	Returns hbar cursor settings
CURSor:HBArs:DELTa?	Returns hbars cursors vertical difference
CURSor:HBArs:POSITION<x>	Sets or returns the hbar cursor<x> vertical position
CURSor:HBArs:UNIts	Returns hbar cursor units
CURSor:HBArs:USE	Sets the horizontal bar cursor measurement scale
CURSor:MODe	Sets or returns whether cursors move in unison or separately
CURSor:VBArs?	Sets or returns the position of vertical bar cursors
CURSor:VBArs:ALTERNATE<x>?	Returns the alternate readout for the waveform (Vbar) cursors
CURSor:VBArs:DELTa?	Returns the difference between vbar cursors
CURSor:VBArs:HPOS<x>?	Returns the horizontal value of the specified vertical bar ticks
CURSor:VBArs:POSITION<x>	Sets or returns the vbar cursor<x> horizontal position
CURSor:VBArs:UNIts	Sets or returns the units for vbar cursors
CURSor:VBArs:USE	Sets the vertical bar cursor measurement scale
CURSor:VBArs:VDELTa?	Returns the vertical difference between the two vertical bar cursor ticks
CURSor:XY:POLar:RADIUS:DELta?	Returns the difference between the cursors X radius and the cursor Y radius

Table 2-17: Cursor Commands (cont.)

Command	Description
CURSor:XY:POLar:RADIUS:POSITION<x>?	Returns the polar radius of the specified cursor
CURSor:XY:POLar:RADIUS:UNIts?	Returns the polar radius units
CURSor:XY:POLar:THETA:DELta?	Returns the XY cursor polar coordinate delta
CURSor:XY:POLar:THETA:POSITION<x>?	Returns the cursor X or cursor Y polar coordinate
CURSor:XY:POLar:THETA:UNIts?	Returns the cursor polar coordinate units
CURSor:XY:PRODUCT:DELta?	Returns the difference between the cursors X position and cursor Y position
CURSor:XY:PRODUCT:POSITION<x>?	Returns the position of the X or Y cursor used to calculate the X × Y cursor measurement
CURSor:XY:PRODUCT:UNIts?	Returns the XY cursor product units
CURSor:XY:RATIO:DELta?	Returns the ratio of the difference between the cursor X position and cursor Y position
CURSor:XY:RATIO:POSITION<x>?	Returns the X or Y position for the specified cursor
CURSor:XY:RATIO:UNIts?	Returns the X and Y cursor units for the ratio measurement
CURSor:XY:RECTangular:X:DELta?	Returns the cursor X delta value in rectangular coordinates
CURSor:XY:RECTangular:X:POSITION<x>	Sets or returns the cursor X rectangular coordinates
CURSor:XY:RECTangular:X:UNIts?	Returns the Cursor X rectangular units
CURSor:XY:RECTangular:Y:DELta?	Returns The cursor Y delta value in rectangular coordinates
CURSor:XY:RECTangular:Y:POSITION<x>>	Sets or returns the cursor Y rectangular coordinates
CURSor:XY:RECTangular:Y:UNIts?	Returns the cursor Y rectangular units

Display Command Group

Use the commands in the Display Command Group to change the graticule style, the displayed intensities, and to set the characteristics of the waveform display.

Use these commands to set the style that best displays your waveforms and graticule display properties. Note that the mode you choose globally affects all displayed waveforms.

Table 2-18: Display Commands

Command	Description
DISplay?	Returns current display settings
DISplay:CLOCk	Sets or returns the display of the date/time stamp
DISplay:DIGital:HEIght	Sets or returns the height of the digital display and the number of waveforms that you can display
DISplay:FORMat	Sets or returns the display format
DISplay:GRAticule	Sets or returns the type of graticule that is displayed
DISplay:INTENSITy?	Returns all display intensity settings
DISplay:INTENSITy:BACKLight	Sets or returns the backlight intensity for the display
DISplay:INTENSITy:GRAticule	Sets or returns the graticule intensity for the display
DISplay:INTENSITy:WAVEform	Sets or returns the intensity of the waveforms
DISplay:PERSistence	Sets or returns display persistence setting
MESSage:BOX	Sets or returns the size and position of the message window
MESSage:CLEAR	Removes the message text from the message window
MESSage:SHOW	Clears the contents of the message window
MESSage:STATE	Controls the display of the message window

Ethernet Command Group

Use the commands in the Ethernet Command Group to set up the Ethernet remote interface.

Table 2-19: Ethernet Commands

Command	Description
ETHERnet:DHCPbootp	Sets or returns the network initialization search for a DHCP/BOOTP server
ETHERnet:DNS:IPADDress	Sets or returns the network Domain Name Server (Dns) IP address
ETHERnet:DOMAINname	Sets or returns the network domain name
ETHERnet:ENET:ADDress?	Returns the Ethernet address value assigned to the oscilloscope
ETHERnet:GATEWay:IPADDress	Sets or returns the remote interface gateway IP address

Table 2-19: Ethernet Commands (cont.)

Command	Description
ETHERnet:HTTPPort	Sets or returns the remote interface HTTP port value
ETHERnet:IPADDress	Sets or returns the IP address assigned to the oscilloscope
ETHERnet:NAME	Sets or returns the network name assigned to the oscilloscope
ETHERnet:PASSWord	Sets or returns the Ethernet access password
ETHERnet:PING	Causes the oscilloscope to ping the gateway IP address
ETHERnet:PING:STATUS?	Returns the results from pinging the gateway IP address
ETHERnet:SUBNETMask	Sets or returns the remote interface subnet mask value

File System Command Group

Use the commands in the File System Command Group to access USB media. You can use the commands to do the following:

- List the contents of a directory
- Create, rename and delete directories
- Create, read, rename, or delete a file
- Format media

When using these commands, keep the following points in mind:

- File arguments are always enclosed within double quotes: "E:/MYDIR/TEK00001.SET"
- File names follow the non-case sensitive, MSDOS format: [DRIVE:][\PATH\]filename
- File names for commands and queries are not case sensitive. Save commands translate all file names to uppercase for storage

  For Example: SAVE:WAVEFORM CH1,"ch1.isf" results in Ch1 being saved to a file named CH1.ISF.

  RECALL:WAVEFORM "Ch1.isf",REF1 would recall the waveform from the file CH1.ISF to the REF1 internal waveform storage location.
- Path separators may be either forward slashes (/) or back slashes (\)
- The file and directory names have no more than eight characters as the base name, and no more than three characters as the extension as in 8.3 format

NOTE. *Using back slash as a path separator may produce some unexpected results, depending on how your application treats escaped characters. Many applications recognize the sequence of back slash followed by an alphabetic character as an escaped character, and, as such, interpret that alphabetic character as a control character. For example, the sequence "\n" may be interpreted as a newline character; "\t" may be interpreted as a tab character. To ensure that this interpretation does not occur, you can use double back slashes. For example, "E:\\testfile.txt".*

Table 2-20: File System Commands

Command	Description
FILESystem	Returns the file system state
FILESystem:CWD	Sets or returns the current working directory for FILESystem commands.
FILESystem:DELEte	Deletes a named file or directory
FILESystem:DIR?	Returns a list of directory contents
FILESystem:FORMat	Formats a named drive
FILESystem:FREESpace?	Returns the number of bytes of free space on the current drive
FILESystem:MKDir	Creates a new directory
FILESystem:READFile	Writes the contents of the specified file to the specified interface
FILESystem:REName	Assigns a new name to an existing file
FILESystem:RMDir	Deletes a named directory
FILESystem:WRITEFile	Writes the specified block data to the oscilloscope current working directory

FilterVu Command Group

Use the commands in the FilterVu Command Group to set up the FilterVu feature. FilterVu provides a variable low pass filter to block unwanted noise while still displaying high frequency events. The oscilloscope does this by superimposing a filtered foreground waveform over a glitch capture background waveform.

Table 2-21: FilterVuCommands

Command	Description
DISplay:GLITch	Controls the display of the glitch capture waveform
DISplay:INTENSITy:GLITch	Sets the glitch intensity on the display

Table 2-21: FilterVuCommands (cont.)

Command	Description
FILTERVu:FREQuency	Sets or queries the FilterVu frequency
FILTERVu:FREQuency:AVAILable?	Returns a comma separated list of filter frequencies available based on the current instrument settings

For more information, refer to Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments. (See page B-1.)

Hard Copy Command Group

Use the commands in the Hard Copy Command Group to make hard copies.

Table 2-22: Hard Copy Commands

Command	Description
HARDCopy	Sends a copy of the screen display to the selected printer
HARDCopy:INKSaver	Changes hard copy output to print color traces and graticule on a white background
HARDCopy:PREVIEW	Previews the current screen contents with the InkSaver palette applied

Horizontal Command Group

Use the commands in the Horizontal Command Group to control the oscilloscope time bases. You can set the time-per-division of the main time base. You can also use the Horizontal commands to set the scale, horizontal position, and reference of the time base.

Table 2-23: Horizontal Commands

Command	Description
HORizontal?	Returns settings for the horizontal commands
HORizontal:ACQLENGTH?	Returns the record length
HORizontal:DELay:MODe	Sets or returns the horizontal delay mode
HORizontal:DELay:TIMe	Sets or returns the horizontal delay time
HORizontal:DIGital:RECOrdlength:MAIN?	Returns the record length of the main digital acquisition
HORizontal:DIGital:SAMPLERate:MAIN?	Returns the sample rate of the main digital acquisition
HORizontal:POSition	Sets or returns the horizontal position
HORizontal:PREViewstate?	Returns whether or not the acquisition system is in the preview state
HORizontal:RECOrdlength	Sets the horizontal record length to the number of data points in each frame Or returns the current horizontal record length
HORizontal:RESOlution	Sets or returns the horizontal record length to the number of data points in each frame
HORizontal:SAMPLERate	Sets or returns the current horizontal sample rate
HORizontal:SCAle	Sets or returns the time base horizontal scale
HORizontal:TRIGger:POSition	Sets the horizontal position when delay mode is OFF

Mark Command Group

Use the commands in the Mark Command Group to identify areas of the acquired waveform that warrant further investigation.

Table 2-24: Mark Commands

Command	Description
MARK	Move to the next or previous mark on the waveform or returns all learnable settings from the mark commands
MARK:CREATE	Creates a mark on a particular waveform or all waveforms in a column
MARK:DELEte	Deletes a mark on a particular waveform, all waveforms in a column, or all marks
MARK:FREE?	Returns how many marks are free to be used
MARK:SELected:END?	Returns the end of the selected mark, in terms of 0 to 100% of the waveform
MARK:SELected:FOCUS?	Returns the focus of the selected mark, in terms of 0 to 100% of the waveform
MARK:SELected:MARKSINCOLumn?	Returns how many marks are in the current zoom pixel column
MARK:SELected:OWNer?	Returns the owner of the selected mark
MARK:SELected:SOURCE?	Returns the source waveform of the selected mark
MARK:SELected:STARt?	Returns the start of the selected mark, in terms of 0 to 100% of the waveform
MARK:SELected:STATe?	Returns the on or off state of the selected mark
MARK:SELected:ZOOm:POSition?	Returns the position of the selected mark, in terms of 0 to 100% of the upper window
MARK:TOTal?	Returns how many marks are used

Math Command Group

Use the commands in the Math Command Group to create and define a math waveform. Use the available math functions to define your math waveform.

The math waveform you create depends on sources listed in the math expression. If you change these sources, the math waveform you previously defined will be affected.

Math expressions can be simple without any mathematical computations. For example: CH1, which specifies that a waveform shows the signal source of Channel 1.

The acquisition of a live waveform can stop for several reasons: You can turn off the channel, stop the waveform, or stop the trigger. When you turn off the channel, math continues and data is acquired but is not displayed. When you stop either the waveform or the trigger, the math calculation stops, and the last math calculation performed is displayed.

When a live waveform update or reference waveform is altered, math waveforms containing those waveforms as sources are also updated to reflect the changes. Remember that sources must exist, but do not need to be displayed, to be used in and to update math waveforms.

Table 2-25: Math Commands

Command	Description
MATH[1]?	Returns the definition of the math waveform
MATH[1]:DEFine	Sets or returns the current math function as a text string
MATH[1]:HORizontal:POSition	Sets or returns the math horizontal display position for FFT or (non-live) math reference waveforms
MATH[1]:HORizontal:SCAle	Sets or returns the math horizontal display scale for FFT or for Dual Math waveforms
MATH[1]:HORizontal:UNIts	Returns the math waveform horizontal unit value
MATH[1]:LABel	Sets or queries the waveform label for the math waveform
MATH[1]:SPECTral:GATing:INDICators	Enables or disables the display of indicators that show the portion of the source waveform record used to compute the math FFT waveform
MATH[1]:SPECTral:GATing:INDICators:END?	Returns the ending point in the source waveform record used for computing the FFT
MATH[1]:SPECTral:GATing:INDICators:STARt?	Returns the starting point in the source waveform record used for computing the FFT
MATH[1]:SPECTral:MAG	Sets or returns the units of spectral magnification in the math string

Table 2-25: Math Commands (cont.)

Command	Description
MATH[1]:SPECTral:NYQUISTFreq?	Returns the Nyquist frequency of the FFT math waveform
MATH[1]:SPECTral:WINdow	Sets or returns the window function for math waveform spectral input data
MATH[1]:TYPe	Sets or returns the math waveform mode type
MATH[1]:VERTical:POSition	Sets or returns the vertical position of the currently selected math type
MATH[1]:VERTical:SCAle	Sets or returns the vertical scale of the currently selected math type
MATH[1]:VERTical:UNIts?	Returns the math waveform vertical units

Measurement Command Group

Use the commands in the Measurement Command Group to control the automated measurement system.

Up to four automated measurements can be displayed on the screen. In the commands, these measurement readouts are named MEAS<x>, where <x> is the measurement number.

In addition to the four displayed measurements, the measurement commands let you specify an additional measurement, IMMed. The immediate measurement has no front-panel equivalent. Immediate measurements are never displayed. Because they are computed only when needed, immediate measurements slow the waveform update rate less than displayed measurements.

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

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

Table 2-26: Measurement Commands

Command	Description
MEASUrement?	Returns all measurement parameters
MEASUrement:CLEARSNapshot	Removes the measurement snapshot display
MEASUrement:GATing	Sets or returns the measurement gating
MEASUrement:IMMed?	Returns all immediate measurement setup parameters

Table 2-26: Measurement Commands (cont.)

Command	Description
MEASUrement:IMMed:DELay?	Returns information about the immediate delay measurement
MEASUrement:IMMed:DELay:DIRection	Sets or returns the search direction to use for immediate delay measurements
MEASUrement:IMMed:DELay:EDGE<x>	Sets or returns the slope of the edge used for immediate delay “from” and “to” waveform measurements
MEASUrement:IMMed:SOUrce1	Sets or returns the “from” source for all single channel immediate measurements
MEASUrement:IMMed:SOUrce2	Sets or returns the source to measure “to” for phase or delay immediate measurements
MEASUrement:IMMed:TYPe	Sets or returns the type of the immediate measurement
MEASUrement:IMMed:UNIts?	Returns the units of the immediate measurement
MEASUrement:IMMed:VALue?	Returns the value of the immediate measurement
MEASUrement:INDICators?	Returns all measurement indicator parameters
MEASUrement:INDICators:HORZ<x>?	Returns the position of the specified horizontal measurement indicator
MEASUrement:INDICators:NUMHORZ?	Returns the number of horizontal measurement indicators currently being displayed
MEASUrement:INDICators:NUMVERT?	Returns the number of vertical measurement indicators currently being displayed
MEASUrement:INDICators:STATE	Sets or returns the state of visible measurement indicators
MEASUrement:INDICators:VERT<x>?	Returns the value of the specified vertical measurement indicator
MEASUrement:MEAS<x>?	Returns all measurement parameters
MEASUrement:MEAS<x>:COUNt?	Returns the number of values accumulated since the last statistical reset
MEASUrement:MEAS<x>:DELay?	Returns the delay measurement parameters for the specified measurement
MEASUrement:MEAS<x>:DELay:DIRection	Sets or returns the search direction to use for delay measurements
MEASUrement:MEAS<x>:DELay:EDGE<x>	Sets or returns the slope of the edge to use for delay “from” and “to” waveform measurements
MEASUrement:MEAS<x>:MAXimum?	Returns the maximum value found since the last statistical reset

Table 2-26: Measurement Commands (cont.)

Command	Description
MEASUrement:MEAS<x>:MEAN?	Returns the mean value accumulated since the last statistical reset
MEASUrement:MEAS<x>:MINImum?	Returns the minimum value found since the last statistical reset
MEASUrement:MEAS<x>:SOURCE[1]	Sets or returns the channel from which measurements are taken
MEASUrement:MEAS<x>:SOURCE2	Sets or returns the channel to which measurements are sent
MEASUrement:MEAS<x>:STATE	Sets or returns whether the specified measurement slot is computed and displayed
MEASUrement:MEAS<x>:STDdev?	Returns the standard deviation of values accumulated since the last statistical reset
MEASUrement:MEAS<x>:TYPe	Sets or returns the measurement<x> type
MEASUrement:MEAS<x>:UNIts?	Returns measurement<x> units
MEASUrement:MEAS<x>:VALue?	Returns the value of measurement<x>
MEASUrement:METHod	Sets or returns the method used for calculating reference levels
MEASUrement:REFLevel?	Returns the current reference level parameters
MEASUrement:REFLevel:ABSolute:HIGH	Sets or returns the top reference level for rise time
MEASUrement:REFLevel:ABSolute:LOW	Sets or returns the low reference level for rise time
MEASUrement:REFLevel:ABSolute:MID	Sets or returns the mid reference level for measurements
MEASUrement:REFLevel:ABSolute:MID2	Sets or returns the mid reference level for delay "to" measurements
MEASUrement:REFLevel:METHod	Sets or returns the method for assigning high and low reference levels
MEASUrement:REFLevel:PERCent:HIGH	Sets or returns the top reference percent level for rise time
MEASUrement:REFLevel:PERCent:LOW	Sets or returns the low reference percent level for rise time
MEASUrement:REFLevel:PERCent:MID	Sets or returns the mid reference percent level for waveform measurements
MEASUrement:REFLevel:PERCent:MID2	Sets or returns the mid reference percent level for second waveform measurements
MEASUrement:SNAPShot	Displays the measurement snapshot list
MEASUrement:IMMed:SOUrce<x>	Sets or returns the source for the current single channel measurement

Table 2-26: Measurement Commands (cont.)

Command	Description
MEASUrement:MEAS<x>:SOUrce<x>	Sets or returns the source for the specified measurement.
MEASUrement:REFLevel:ABSolute:MID<x>	Sets or returns the mid reference level for the specified channel in absolute volts
MEASUrement:REFLevel:PERCent:MID<x>	Sets or returns the mid reference level for the specified channel in percent

Miscellaneous Command Group

Use the commands in the Miscellaneous Command Group to perform actions that do not fit into other categories.

Several commands and queries are common to all 488.2-1987 devices. The 488.2-1987 standard defines these commands. The common commands begin with an asterisk (*) character.

Table 2-27: Miscellaneous Commands

Command	Description
AUTOSet	Sets the vertical, horizontal and trigger controls to provide a stable display of the selected waveform
CLEARMenu	Clears the current menu from the display
DATE	Sets or returns the date displayed by the oscilloscope
*DDT	Sets or returns the commands that will be executed by the group execute trigger
FPAnel:PRESS	Simulates the action of pressing a specified front-panel button
FPAnel:TURN	Duplicates the action of turning a specified front-panel control knob
GPIBUsb:ID?	Returns the identification string of the connected adaptor module and firmware version
HEADer	Sets or returns the Response Header Enable State
ID?	Returns identifying information about the oscilloscope and its firmware
*IDN?	Returns the same information as the ID? command except the data is formatted according to Tektronix Codes & Formats
LANGuage	Sets or returns the user interface display language

Table 2-27: Miscellaneous Commands (cont.)

Command	Description
LOCk	Sets or returns the front panel lock state
*LRN?	Returns a listing of oscilloscope settings
MESSage	Sets or queries message parameters
NEWpass	Changes the password for user protected data
PASSWord	Enables the *PUD and NEWpass set commands
REM	Specifies a comment, which is ignored by the oscilloscope
SET?	Returns a listing of oscilloscope settings
TEKSecure	Initializes both waveform and setup memories
TIME	Sets or returns the time displayed by the oscilloscope
TOTaluptime?	Returns the total number of hours that the oscilloscope has been turned on since the nonvolatile memory was last programmed
*TRG	Performs the group execute trigger (GET)
*TST?	Tests the interface and returns the status
UNLock	Unlocks front panel
USBTMC	Returns the USBTMC information
USBTMC:PRODUCTID:DECimal?	Returns the product ID in decimal format
USBTMC:PRODUCTID:HEXadecimal?	Returns the product ID in hexadecimal format
USBTMC:SERIALnumber?	Returns the instrument serial number
USBTMC:VENDORID:DECimal?	Returns the vendor ID in decimal format
USBTMC:VENDORID:HEXadecimal?	Returns the vendor ID in hexadecimal format
VERBose	Sets or returns the verbose state

PictBridge Command Group

Use the commands in the PictBridge to store the printer settings.

Table 2-28: PictBridge Commands

Command	Description
PICTBridge:DATEPrint	Enables or disables printing the date on the print output
PICTBridge:DEFault	Sets the arguments for all PictBridge commands to their default values

Table 2-28: PictBridge Commands (cont.)

Command	Description
PICTBridge:IDPrint	Enables or disables printing the oscilloscope model and serial number on the print output
PICTBridge:IMAGESize	Sets or returns the image print size
PICTBridge:PAPERSize	Sets the output print paper size
PICTBridge:PAPERType	Sets or returns the paper type
PICTBridge:PRINTQual	Sets or returns the output print quality
USBDevice:CONFigure	Enables or disables the rear USB port for use with Pictbridge printers

Save and Recall Command Group

Use the commands in the Save and Recall Command Group to store and retrieve internal waveforms and settings. When you save a setup, you save all the settings of the oscilloscope. When you recall a setup, the oscilloscope restores itself to the state it was in when you originally saved the setting.

Table 2-29: Save and Recall Commands

Command	Description
FACtory	Resets the oscilloscope to factory default settings
*RCL	Recalls saved oscilloscope settings
RECAll:SETUp	Recalls saved oscilloscope settings
RECAll:WAVEform	Recalls a stored waveform to a reference location
REF<x>:SCAle	Sets or returns the vertical scale for the channel specified by <x>, where x is the reference channel number
*SAV	Stores the state of the oscilloscope to a specified memory location
SAVe:ASSIgn:TYPe	Sets or returns the assignment of the data to be saved
SAVe:EVENTtable:BUS<x>	Saves event table data from bus<x> to a specified file and location
SAVe:IMAGe	Saves a capture of the screen image into the specified file
SAVe:IMAGe:FILEFormat	Sets or returns the file format to use for saving screen images when the file type cannot be determined from the given file name or when screen images are captured by using the front panel

Table 2-29: Save and Recall Commands (cont.)

Command	Description
SAVe:IMAGe:LAYout	Sets or returns the layout to use for saved screen images
SAVe:SETUp	Saves the state of the oscilloscope to a specified memory location or file
SAVe:WAVEform	Saves a waveform to one of four reference memory locations or a file
SAVe:WAVEform:FILEFormat	Sets or returns the format for saved waveforms
SAVe:WAVEform:GATIng	Specifies whether save waveform operations should save the entire waveform or a specified portion of the waveform
SAVe:WAVEform:SPREADSheet:RESOlution	Specifies whether to save the full (LRL) or reduced (thumb) resolution waveform to a CSV file.
SETUP<x>:DATE?	Returns the date when the specified oscilloscope setup was saved
SETUP<x>:LABEL	Sets or returns the specified oscilloscope setup label
SETUP<x>:TIME?	Returns the time when the specified oscilloscope setup was saved

Search Command Group

Use the commands in the Search Commands Group to seek out information in waveform records.

Search Commands

Command	Description
SEARCH?	Returns all search-related settings
SEARCH:SEARCH<x>:COPy	Copies the search criteria to the trigger, or the trigger criteria to the search.
SEARCH:SEARCH<x>:STATE	Sets the search state to on or off
SEARCH:SEARCH<x>:TOTAL?	Returns the total number of matches for search <x>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:CONDition	Sets or returns the search condition for CAN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:DIRection	Sets or returns the CAN search condition to be valid on a READ, WRITE or either
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier	Sets or returns the CAN data qualifier
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for CAN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue	Sets or returns the binary data string to be used for CAN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:FRAMEtype	Sets or returns the CAN Frame Type to be used
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe	Sets or returns the CAN addressing mode to standard or extended format
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue	Sets or returns the binary address string to be used for CAN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:CONDition	Sets or returns the search condition for a LIN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue	Sets or returns the binary data string
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier	Sets or returns the LIN data qualifier
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:SIZe	Sets or returns the length of the data string in bytes
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:VALue	Sets or returns the binary data string used for a LIN search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:ERRTYPE	Sets or returns the error type used for a LIN Search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue	Sets or returns the binary address string used for LIN search

Search Commands (cont.)

Command	Description
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe	Sets or returns the I2C address mode to 7 or 10-bit
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe	Sets or returns the I2C address type to I2C special addresses
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue	Sets or returns the binary address string to be used for I2C search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:CONDition	Sets or returns the search condition for I2C search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:DIRection	Sets or returns the I2C search condition to be valid on a READ, WRITE or either
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for I2C search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:VALue	Sets or returns the binary data string to be used for I2C search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:PARallel:VALue	Sets or returns the binary data string to be used for a parallel trigger search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:CONDition	Sets or returns the trigger condition for a RS232 trigger
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe	Sets or returns the length of the data string for a RS232 trigger, if the trigger condition is RX
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue	Sets or returns the binary data string for a RS232 trigger, if the condition involves RX
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe	Sets or returns the length of the data string to be used for a RS232 Trigger, if the Trigger condition is TX
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue	Sets or returns the binary data string to be used for a RS232 trigger, if the condition involves RX
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:CONDition	Sets or returns the search condition for SPI search
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MISO|:IN}:VALue	Sets or returns the binary data string to be used for SPI search if the search condition is MISO or MISOMOSI.
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MOSI|:OUT}:VALue	Sets or returns the binary data string for an SPI search if the search condition is MISO or MISOMOSI
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for SPI search
SEARCH:SEARCH<x>:TRIGger:A:BUS:SOUrce	Sets or returns the bus for a serial search
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SLOpe	Sets or returns the slope for an edge search

Search Commands (cont.)

Command	Description
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SOUrce	Sets or returns the source waveform for an edge search
SEARCH:SEARCH<x>:TRIGger:A:LEVel	Sets or returns the level for an edge search
SEARCH:SEARCH<x>:TRIGger:A:LEVel:CH<x>	Sets or returns the level for an edge search of the specified channel
SEARCH:SEARCH<x>:TRIGger:A:LEVel:MATH	Sets or returns the math waveform level for edge search
SEARCH:SEARCH<x>:TRIGger:A:LEVel:REF<x>	Sets or returns the reference waveform level for edge search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:FUNCtion	Sets or returns the logic operator for the logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CH<x>	Sets or returns the Boolean logic criteria for the logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:EDGE	Sets or returns whether the clock edge is rise or fall for a logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:SOUrce	Sets or returns the clock source definition for logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:MATH	Sets or returns the Boolean logic criteria for the logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:REF<x>	Sets or returns the Boolean logic criteria for the logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:CH<x>	Sets or returns the Boolean logic criteria for the logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:D<x>	Sets or returns the waveform logic criteria for a trigger search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:MATH	Sets or returns the Boolean logic criteria for the logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:D<x>	Sets or returns the logic criteria for a logic pattern search to determine where to place a mark
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:REF<x>	Sets or returns the Boolean logic criteria for the logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn	Sets or returns the condition for generating a logic pattern search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit	Sets or returns the maximum time that the selected pattern may be true
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit	Sets or returns the minimum time that the selected pattern may be true
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:CH<x>	Sets or returns the channel threshold level for an logic search
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:MATH	Sets or returns the math waveform threshold level for logic search

Search Commands (cont.)

Command	Description
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:REF<x>	Sets or returns the reference waveform threshold level for logic search
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:CH<x>	Sets or returns the lower waveform threshold level for all channel waveform searches
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:MATH	Sets or returns the lower waveform threshold level for all math waveform searches
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:REF<x>	Sets or returns the lower waveform threshold level for all reference waveform searches
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:POLarity	Sets or returns the polarity for a pulse search
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:SOUrce	Sets or returns the source waveform for a pulse search
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WHEn	Sets or returns the condition for generating a pulse width search
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth	Sets or returns the pulse width setting for a pulse width search
SEARCH:SEARCH<x>:TRIGger:A:RUNT:POLarity	Sets or returns the polarity setting for a runt search
SEARCH:SEARCH<x>:TRIGger:A:RUNT:SOUrce	Sets or returns the source setting for a runt search
SEARCH:SEARCH<x>:TRIGger:A:RUNT:WHEn	Sets or returns the condition setting for a runt search
SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth	Sets or returns the width setting for a runt search
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:EDGE	Sets or returns the clock slope setting for a setup/hold search
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:SOUrce	Sets or returns the clock source setting for an setup/hold search
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:THReshold	Sets or returns the clock threshold setting for an setup/hold search
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:SOUrce	Sets or returns the data source setting for an setup/hold search
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:THReshold	Sets or returns the data threshold setting for an setup/hold search
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:CH<x>	Sets or returns the trigger search setup and hold lower threshold to determine where to place a mark. Ch<x> is the channel number
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold{:MATH|:MATH1}	Sets or returns the trigger search setup and hold lower threshold to determine where to place a mark

Search Commands (cont.)

Command	Description
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:REF<x>	Sets or returns the trigger search setup and hold lower threshold to determine where to place a mark. Ref<x> is the reference waveform number
SEARCH:SEARCH<x>:TRIGger:A:SETHold:HOLDTime	Sets or returns the hold time setting for an setup/hold search
SEARCH:SEARCH<x>:TRIGger:A:SETHold:SETTime	Sets or returns the setup time setting for an setup/hold search
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:DELTatime	Sets or returns the transition time setting for an transition search
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:POLarity	Sets or returns the polarity setting for an transition search
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:SOUrce	Sets or returns the source setting for an transition search
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:WHEn	Sets or returns the condition setting for an transition search
SEARCH:SEARCH<x>:TRIGger:A:TYPe	Sets or returns the trigger type setting for a search
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:CH<x>	Sets or returns the waveform upper threshold level for all channel waveform searches
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:MATH	Sets or returns the waveform upper threshold level for all math waveform searches
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:REF<x>	Sets or returns the waveform upper threshold level for all reference waveform searches

Status and Error Command Group

Use the commands in the Status and Error Command Group to determine the status of the oscilloscope and control events.

Several commands and queries used with the oscilloscope are common to all IEEE488.2 compliant devices. The IEEE Std 488.2-1987 defines these commands and queries. The common commands begin with an asterisk (*) character.

Table 2-30: Status and Error Commands

Command	Description
ALLEv?	Returns all events and their messages
BUSY?	Returns oscilloscope status
*CLS	Clears status
DESE	Sets or returns the bits in the Device Event Status Enable Register

Table 2-30: Status and Error Commands (cont.)

Command	Description
*ESE	Sets or returns the bits in the Event Status Enable Register
*ESR?	Returns the contents of the Standard Event Status Register
EVENT?	Returns event code from the event queue
EVMsg?	Returns event code, message from the event queue
EVQty?	Return number of events in the event queue
*OPC	Generates the operation complete message in the standard event status register when all pending operations are finished Or returns "1" when all current operations are finished
*PSC	Sets or returns the power on status flag
*PUD	Sets or returns a string of protected user data
*RST	Resets the oscilloscope to factory default settings
*SRE	Sets or returns the bits in the Service Request Enable Register
*STB?	Returns the contents of the Status Byte Register
*WAI	Prevents the oscilloscope from executing further commands until all pending operations finish

Trigger Command Group

Use the commands in the Trigger Command Group to control all aspects of triggering for the oscilloscope. There is one trigger: A. You can perform edge, pulse width, logic, video, runt, transition (rise/fall time), setup & hold and bus triggering using the A trigger.

Example command sequences that show different triggers and searches are available. (See page E-1, *Search and Trigger Command Sequence Examples*.)

Edge Trigger

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

To set up an edge trigger, set the A trigger type to EDGe using the command TRIGger:A:TYPe. Then use the commands TRIGger:A:EDGE:COUPling, TRIGger:A:EDGE:SLOpe, and TRIGger:A:EDGE:SOUrce You can

set the threshold voltage levels using TRIGger:A:LEVel:AUXin, TRIGger:A:LEVel:CH<x>, or TRIGger:A:LEVel:D<x>.

Logic Trigger

Logic triggering lets you logically combine the signals on one or more channels; the oscilloscope then triggers when it detects a certain combination of signal levels (set using one of the `TRIGger:A:LOGIc:INPut` commands as well as TRIGger:A:LOGIc:FUNCtion).

To set up a logic trigger, first set the trigger type to `LOGIc` using the command TRIGger:A:TYPe, and then use the command TRIGger:A:LOGIc:CLAss to select `LOGIC`.

You can perform logic triggering using a clock (by setting TRIGger:A:LOGIc:INPut:CLOCk:SOUrce to one of the channels), so that the oscilloscope triggers when the specified logical combination of data channels is true during a transition on the clock channel. The specified data sources cannot include the waveform specified as the clock source. (Use TRIGger:A:LOGIc:INPut:CH<x>, TRIGger:A:LOGIc:INPut:D<x>, etc.)

You can also perform logic triggering without using a clock (by setting TRIGger:A:LOGIc:INPut:CLOCk:SOUrce to `NONE`), so that the oscilloscope triggers when the specified logical pattern of data channels is met. (Use the commands TRIGger:A:LOGIc:PATtern:WHEn and TRIGger:A:LOGIc:PATtern:DELTatime.)

Setup and Hold Trigger

A setup and hold trigger occurs when a logic data input changes state inside of the setup or hold time relative to a clock edge.

To set up a setup/hold trigger, first set the trigger type to `LOGIc` using the command TRIGger:A:TYPe, and then use the command TRIGger:A:LOGIc:CLAss to select `SETHold`. Then use TRIGger:A:SETHold:DATa:SOUrce, TRIGger:A:SETHold:CLOCk:SOUrce, TRIGger:A:SETHold:CLOCk:EDGE and related commands.

Pulse width Trigger

A pulse width trigger occurs whenever the oscilloscope detects a pulse that is less than, greater than, equal to, or not equal to a specified time. Additionally, it can occur when the pulse width is within, or outside of a specified range. You can trigger on either positive or negative pulses.

To set up a pulse width trigger, first set the trigger type to `PULSe` using the command TRIGger:A:TYPe. Then select `WIDth` using the command TRIGger:A:PULse:CLAss. Then use TRIGger:A:PULSEWidth:SOUrce, TRIGger:A:PULSEWidth:WHEn, TRIGger:A:PULSEWidth:WIDth and related commands. You can set the threshold voltage levels using TRIGger:A:LEVel:AUXin, TRIGger:A:LEVel:CH<x>, or TRIGger:A:LEVel:D<x>.

Runt Trigger

Runt triggering lets you trigger on a pulse amplitude that crosses one threshold but fails to cross a second threshold before recrossing the first.

To set up a runt trigger, first set the trigger type to `PULSe` using the command TRIGger:A:TYPe. Then select `RUNt` using the command TRIGger:A:PULse:CLAss. Then use TRIGger:A:RUNT:SOUrce, TRIGger:A:RUNT:WIDth, TRIGger:A:RUNT:WHEn, TRIGger:A:UPPerthreshold:CH<x> and related commands. You can set the threshold voltage levels using TRIGger:A:LEVel:AUXin, TRIGger:A:LEVel:CH<x>, or TRIGger:A:LEVel:D<x>.

Transition Trigger

Transition triggering (also called rise/fall time, and once known as slew rate) causes a trigger on pulse edges that traverse between two thresholds at a rate faster than or slower than the specified time.

To set up a transition trigger, first set the trigger type to `PULSe` using the command TRIGger:A:TYPe. Then select `TRANsition` using the command TRIGger:A:PULse:CLAss. Then use TRIGger:A{:TRANsition|:RISEFall}:SOUrce, TRIGger:A{:TRANsition|:RISEFall}:DELTatime and related commands. You can set the threshold voltage levels using TRIGger:A:LEVel:AUXin, TRIGger:A:LEVel:CH<x>, or TRIGger:A:LEVel:D<x>.

Bus Trigger

Bus triggering can be performed if you have one or more appropriate application modules installed (see below for application module list).

To set up a bus trigger, set the A trigger type to `BUS` using the command TRIGger:A:TYPe. Specify which bus (1-4) to trigger on using TRIGger:A:BUS:SOUrce. Set the bus type (i.e., parallel, I²C, etc. using the command TRIGger:A:BUS. Then set the appropriate trigger criteria according to bus type (using TRIGger:A:BUS:B<x>:LIN:CONDition, etc.)

Video Trigger

To set up a video trigger, set the A trigger type to `VIDeo` using the command TRIGger:A:TYPe. Then use TRIGger:A:VIDeo:SOUrce, TRIGger:A:VIDeo:STANdard,, and related commands.

Serial bus application modules

A table is available that lists all commands enabled by each application module (See page D-1, *Application Module-enabled Commands*.)

- For I²C or SPI buses, install the DPO2EMBD application module.
- For RS-232, RS-422, RS-485, or UART buses, install the DPO2COMP module.
- For CAN or LIN buses, install the DPO2AUTO module.

Table 2-31: Trigger Commands

Command	Description
TRIGger	Forces a trigger event to occur
TRIGger:FREQuency?	Returns the available trigger frequency in hertz.
TRIGger:A	Sets A trigger level to 50% or returns current A trigger parameters
TRIGger:A:BUS	Sets or returns the serial trigger type
TRIGger:A:BUS:B<x>:CAN:CONDition	Sets or returns the CAN condition
TRIGger:A:BUS:B<x>:CAN:DATa:DIRection	Sets or returns the CAN trigger condition to be valid on a READ, WRITE, or either
TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier	Sets or returns the CAN data qualifier
TRIGger:A:BUS:B<x>:CAN:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for CAN trigger
TRIGger:A:BUS:B<x>:CAN:DATa:VALue	Sets or returns the binary data string to be used for CAN trigger
TRIGger:A:BUS:B<x>:CAN:FRAMEtype	Sets or returns the CAN trigger frame type
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe	Sets or returns the CAN addressing mode
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue	Sets or returns the binary address string used for the CAN trigger
TRIGger:A:BUS:B<x>:LIN:CONDition	Sets or returns the trigger condition for LIN
TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue	Sets or returns the binary data string to be used for LIN trigger
TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier	Sets or returns the LIN data qualifier
TRIGger:A:BUS:B<x>:LIN:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for LIN trigger
TRIGger:A:BUS:B<x>:LIN:DATa:VALue	Sets or returns the binary data string
TRIGger:A:BUS:B<x>:LIN:ERRTYPE	Sets or returns the error type

Table 2-31: Trigger Commands (cont.)

Command	Description
TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue	Sets or returns the binary address string used for LIN trigger
TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe	Sets or returns the I2C address mode to 7 or 10-bit
TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe	Sets or returns the I2C address type to USER
TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue	Sets or returns the binary address string used for the I2C trigger
TRIGger:A:BUS:B<x>:I2C:CONDition	Sets or returns the trigger condition for I2C trigger
TRIGger:A:BUS:B<x>:I2C:DATa:DIRection	Sets or returns the I2C trigger condition valid on a READ, WRITE, or either
TRIGger:A:BUS:B<x>:I2C:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for I2C trigger
TRIGger:A:BUS:B<x>:I2C:DATa:VALue	Sets or returns the binary data string used for I2C triggering
TRIGger:A:BUS:B<x>:PARallel:VALue	Sets or returns the binary data string to be used for a parallel trigger
TRIGger:A:BUS:B<x>:RS232C:CONDition	Sets or returns the condition for a RS232C trigger
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe	Sets or returns the length of the data string in Bytes for a RX RS232 Trigger
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue	Sets or returns the binary data string for a RX RS232 trigger
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe	Sets or returns the length of the data string for a TX RS232 trigger
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue	Sets or returns the binary data string for a RS232 trigger if the trigger condition involves TX
TRIGger:A:BUS:B<x>:SPI:CONDition	Sets or returns the trigger condition for SPI triggering
TRIGger:A:BUS:B<x>:SPI:DATa{:IN|:MISO}:VALue	Sets or returns the binary data string to be used for SPI trigger

Table 2-31: Trigger Commands (cont.)

Command	Description
TRIGger:A:BUS:B<x>:SPI:DATa{:OUT|:MOSI}:VALue	Sets or returns the binary data string used for the SPI trigger
TRIGger:A:BUS:B<x>:SPI:DATa:SIZe	Sets or returns the length of the data string in bytes to be used for SPI trigger
TRIGger:A:BUS:SOUrce	Sets or returns the source for a bus trigger
TRIGger:A:EDGE?	Returns the source, coupling and source for the A edge trigger
TRIGger:A:EDGE:COUPling	Sets or returns the type of coupling for the A edge trigger
TRIGger:A:EDGE:SLOpe	Sets or returns the slope for the A edge trigger
TRIGger:A:EDGE:SOUrce	Sets or returns the source for the A edge trigger
TRIGger:A:HOLDoff?	Returns the A trigger holdoff parameters
TRIGger:A:HOLDoff:TIMe	Sets or returns the A trigger holdoff time
TRIGger:A:LEVel	Sets or returns the trigger level for the A trigger
TRIGger:A:LEVel:AUXin	Sets or returns the trigger level for the AUXIN port
TRIGger:A:LEVel:CH<x>	Specifies or returns the trigger level for the specified trigger channel
TRIGger:A:LEVel:D<x>	Sets or returns the trigger level for the specified digital channel <x>, where x is the channel number
TRIGger:A:LOGIc?	Returns all A trigger logic settings
TRIGger:A:LOGIc:CLAss	Sets or returns the type of A trigger logic
TRIGger:A:LOGIc:FUNCtion	Sets or returns the logical combination of the input channels for the A logic trigger
TRIGger:A:LOGIc:INPut?	Returns the logic input values for all channels

Table 2-31: Trigger Commands (cont.)

Command	Description
TRIGger:A:LOGIc:INPut:CH<x>	Specifies or returns the logic setting for the specified channel
TRIGger:A:LOGIc:INPut:D<x>	Sets or returns the logic pattern for a trigger on digital channel <x>, where x is the channel number
TRIGger:A:LOGIc:INPut:CLOCk:EDGE	Sets the polarity of the clock channel.
TRIGger:A:LOGIc:INPut:CLOCk:SOUrce	Sets or returns the channel to use as the clock source
TRIGger:A:LOGIc:PATtern?	Returns the conditions for generating an A logic pattern trigger
TRIGger:A:LOGIc:PATtern:DELTatime	Sets or returns the pattern trigger delta time value
TRIGger:A:LOGIc:PATtern:INPut:D<x>	Sets or returns the A logic trigger input for the specified digital channel <x>, where x is the channel number.
TRIGger:A:LOGIc:PATtern:WHEn	Sets or returns the pattern logic condition on which to trigger the oscilloscope
TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit	Sets or returns the maximum time that the selected pattern may be true and still generate an A logic pattern trigger
TRIGger:A:LOGIc:PATtern:WHEn:MORELimit	Sets or returns the minimum time that the selected pattern may be true and still generate an A logic pattern trigger
TRIGger:A:LOGIc:THReshold:(CH<x>|D<x>)	Sets or queries the logic trigger threshold voltage for the specified channel
TRIGger:A:LOWerthreshold:CH<x>	Sets or returns the lower threshold for the channel selected
TRIGger:A:LOWerthreshold{:EXT|:AUX}	Sets or returns the lower threshold for the Auxiliary input.
TRIGger:A:MODe	Sets or returns the A trigger mode
TRIGger:A:PULse?	Returns the A pulse trigger parameters

Table 2-31: Trigger Commands (cont.)

Command	Description
TRIGger:A:PULse:CLAss	Sets or returns the type of pulse on which to trigger
TRIGger:A:PULSEWIDth?	Returns the trigger A pulse width parameters
TRIGger:A:PULSEWidth:POLarity	Sets or returns the polarity for the A pulse width trigger
TRIGger:A:PULSEWidth:SOUrce	Sets or returns the source for the pulse width trigger
TRIGger:A:PULSEWidth:WHEn	Sets or returns the criteria for width specification of pulse width trigger events
TRIGger:A:PULSEWidth:WIDth	Sets or returns the width setting for the pulse width trigger
TRIGger:A:RUNT?	Returns the current A runt pulse trigger logic parameters
TRIGger:A:RUNT:POLarity	Sets or returns the polarity for the A pulse runt trigger
TRIGger:A:RUNT:SOUrce	Sets or returns the source for the A pulse trigger
TRIGger:A:RUNT:WHEn	Sets or returns the type of pulse width the trigger checks for when it uncovers a runt
TRIGger:A:RUNT:WIDth	Sets or returns the minimum width for A pulse runt trigger
TRIGger:A:SETHold?	Returns settings for setup and hold violation triggering
TRIGger:A:SETHold:CLOCk?	Returns clock edge polarity, voltage threshold and source input for setup/hold triggering
TRIGger:A:SETHold:CLOCk:EDGE	Sets or returns the clock edge polarity for setup and hold triggering
TRIGger:A:SETHold:CLOCk:SOUrce	Sets or returns the clock source for the A logic trigger setup and hold input
TRIGger:A:SETHold:CLOCk:THReshold	Sets or returns the clock voltage threshold for setup and hold trigger

Table 2-31: Trigger Commands (cont.)

Command	Description
TRIGger:A:SETHold:DATa?	Returns the voltage threshold and data source for the setup/hold trigger
TRIGger:A:SETHold:DATa:SOUrce	Sets or returns the data source for the setup and hold trigger
TRIGger:A:SETHold:DATa:THReshold	Sets or returns the data voltage threshold for setup and hold trigger
TRIGger:A:SETHold:HOLDTime	Sets or returns the hold time for the setup and hold violation triggering
TRIGger:A:SETHold:SETTime	Sets or returns the setup time for setup and hold violation triggering
TRIGger:A:SETHold:THReshold:CH<x>	Sets or queries the threshold for the channel
TRIGger:A:UPPerthreshold:CH<x>	Sets the upper threshold for the channel selected
TRIGger:A{:TRANsition|:RISEFall}?	Returns the delta time, polarity, and both upper and lower threshold limits for the transition time trigger
TRIGger:A{:TRANsition|:RISEFall}:DELTatime	Sets or returns the delta time used in calculating the transition value
TRIGger:A{:TRANsition|:RISEFall}:POLarity	Sets or returns the polarity for the A pulse transition trigger
TRIGger:A{:TRANsition|:RISEFall}:SOUrce	Sets or returns the source for transition trigger
TRIGger:A{:TRANsition|:RISEFall}:WHEn	Sets or returns the relationship of delta time to transitioning signal
TRIGger:A:TYPe	Sets or returns the type of A trigger
TRIGger:A:VIDeo?	Returns the video parameters for the A trigger
TRIGger:A:VIDeo:HOLDoff:FIELD	Sets or returns the video trigger holdoff
TRIGger:A:VIDeo:LINE	Sets or returns the video line number on which the oscilloscope triggers

Table 2-31: Trigger Commands (cont.)

Command	Description
TRIGger:A:VIDeo:POLarity	Sets or returns the polarity of the A video trigger
TRIGger:A:VIDeo:SOUrce	Sets or returns the polarity of the video trigger
TRIGger:A:VIDeo:STANdard	Sets or returns the video standard
TRIGger:A:VIDeo{:SYNC|:FIELD}	Sets or returns the video field trigger
TRIGger:EXTernal?	Returns external trigger parameters
TRIGger:EXTernal:PRObe	Sets or returns the attenuation factor value of the external probe connector
TRIGger:EXTernal:YUNIts?	Returns the external trigger vertical (Y) units value
TRIGger:STATE?	Returns the current state of the triggering system

Vertical Command Group

Use the commands in the Vertical Command Group to control the vertical setup of all live (channel) waveforms for acquisition and to display of channel, reference, and math waveforms.

Table 2-32: Vertical Commands

Command	Description
AUXin?	Returns auxiliary input parameters
AUXin:PRObe	Returns all information concerning the probe attached to auxiliary input
AUXin:PRObe:AUTOZero	Sets the TekVPI probe attached to the auxiliary input to autozero
AUXin:PRObe:COMMAND	Sets the state of the specified probe control
AUXin:PRObe:DEGAUss	Starts a degauss/autozero cycle on a TekVPI current probe attached to the auxiliary input
AUXin:PRObe:DEGAUss:STATE?	Returns the degauss state of the TekVPI current probe attached to the auxiliary input
AUXin:PRObe:FORCEDRange	Sets or returns the range of the TekVPI probe attached to the auxiliary input
AUXin:PRObe:GAIN	Sets or returns the gain factor of the probe that is attached to the auxiliary input

Table 2-32: Vertical Commands (cont.)

Command	Description
AUXin:PRObe:ID:SERnumber?	Returns the serial number of the probe that is attached to the auxiliary input
AUXin:PRObe:ID:TYPE?	Returns the type of probe that is attached to the auxiliary input
AUXin:PRObe:RESistance?	Returns the resistance of the probe that is attached to the Auxiliary input
AUXin:PRObe:SIGnal	Sets or returns the input bypass setting on VPI probes that support input bypass
AUXin:PRObe:UNIts?	Returns the units of measure of the probe that is attached to the auxiliary input
CH<x>?	Returns vertical parameters for the specified channel
CH<x>:BANdwidth	Sets or returns the bandwidth of the specified channel
CH<x>:COUPling	Sets or returns the coupling setting for the specified channel
CH<x>:DESKew	Sets or returns the deskew time for the specified channel
CH<x>:IMPedance	Sets or returns channel <x> input impedance
CH<x>:INVert	Sets or returns the invert function for the specified channel
CH<x>:LABel	Sets or returns the waveform label for channel <x>
CH<x>:OFFSet	Sets or returns the channel offset
CH<x>:POSition	Sets or returns the channel vertical position
CH<x>:PRObe?	Returns the gain, resistance, units, and ID of the probe that is attached to the specified channel
CH<x>:PRObe:AUTOZero	Sets the TekVPI probe attached to the specified channel input to autozero
CH<x>:PRObe:COMMAND	Sets the state of the specified probe control
CH<x>:PRObe:DEGAUss	Starts a degauss/autozero cycle on a TekVPI current probe attached to the specified channel input
CH<x>:PRObe:DEGAUss:STATE?	Returns the state of the probe degauss
CH<x>:PRObe:FORCEDRange	Sets or returns the range on a TekVPI probe attached to the specified channel
CH<x>:PRObe:GAIN	Sets or returns the gain factor of the probe that is attached to the specified channel
CH<x>:PRObe:ID?	Returns the type and serial number of the probe that is attached to the specified channel

Table 2-32: Vertical Commands (cont.)

Command	Description
CH<x>:PRObe:ID:SERnumber?	Returns the serial number of the probe that is attached to the specified channel
CH<x>:PRObe:ID:TYPE?	Returns the type of probe that is attached to the specified channel
CH<x>:PRObe:RESistance?	Returns the resistance of the probe that is attached to the specified channel
CH<x>:PRObe:SIGnal	Sets or returns the input bypass setting of channel <x>TekVPI probe
CH<x>:PRObe:UNIts?	Returns the units of measure of the probe that is attached to the specified channel
CH<x>:SCAle	Sets or returns the vertical scale of the specified channel
CH<x>:TERmination	Sets or returns channel input termination
CH<x>:VOLts	Sets or returns the vertical sensitivity for channel <x>
CH<x>:YUNits	Sets or returns the units for the specified channel to a custom string
REF<x>?	Returns reference waveform data for channel <x>
REF<x>:DATE?	Returns the date that a reference waveform was stored
REF<x>:HORizontal:DELay:TIMe	Sets or returns the horizontal position of the specified reference waveform in percent of the waveform that is displayed to the right of the center vertical graticule
REF<x>:HORizontal:SCAle	Sets or returns the horizontal scale for a reference waveform
REF<x>:LABel	Sets or returns the specified reference waveform label
REF<x>:POSition	Sets or returns the vertical position for channel <x>, where x is the reference channel number
REF<x>:TIMe?	Returns the time that a reference waveform was stored
REF<x>:VERTical:POSition	Sets or returns the vertical position of the specified reference waveform
REF<x>:VERTical:SCAle	Sets or returns the reference waveform vertical scale in vertical units/div
SELect	Returns information on which waveforms are on or off and which waveform is selected.

Table 2-32: Vertical Commands (cont.)

Command	Description
SELect:BUS<x>	Turns on or off the specified bus waveform or returns whether the specified bus channel is on or off
SELect:CH<x>	Turns on or off the specified waveform or returns whether the specified channel is on or off
SELect:D<x>	Turns on the display of the digital channel <x> and resets the acquisition. <x > is the channel number
SELect:CONTROl	Sets or returns the waveform that is selected as the implied recipient of channel-related commands
SELect:MATH[1]	Turns on or off the math waveform or returns whether the math waveform is on or off
SELect:REF<x>	Turns on or off the specified reference waveform or returns whether the specified reference waveform is on or off

Waveform Transfer Command Group

Use the commands in the Waveform Transfer Command Group to transfer waveform data points to and from the oscilloscope. Waveform data points are a collection of values that define a waveform. One data value usually represents one data point in the waveform record. When working with envelope waveforms, each data value is either the minimum or maximum of a min/max pair.

NOTE. *The source waveform (as specified by the DATa:SOUrce) must be turned on for the following queries to return a response:*

CURVe?

DATa:COMPosition:AVAILable?

WAVFrm?

WFMOutpre:RECOrdlength?

WFMOutpre:FILTERFreq?

WFMOutpre:NR_Pt?

WFMOutpre:PT_Fmt?

WFMOutpre:PT_ORder?

WFMOutpre:XUNit?

WFMOutpre:XINcr?

WFMOutpre:XZEro?

WFMOutpre:PT_Off?

WFMOutpre:YUNit?

WFMOutpre:YMUlt?

WFMOutpre:YOFf?

WFMOutpre:YZEro?

Before you transfer waveform data, you must specify the data format, composition, resolution, record length, and waveform source or destination. For more information on the waveform transfer features which are unique to the MSO/DPO2000B and MSO/DPO2000 series, refer to Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instructions.(See page B-1.)

Data Formats

All data points for DPO models are signed integer format only. Valid data widths for CH1–CH4, MATH, and REF1, REF2 are 1 and 2-byte widths. The valid data widths for the digital collection (MSO models) is either 4 or 8-byte widths.

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

ASCII Data. ASCII data is represented by 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
```

NOTE. *You can use ASCII to obtain a readable and easier to format output than binary. However, the oscilloscope may require bytes to send the same values with ASCII than with binary, reducing transmission speed.*

The use of ASCII for waveform data transfer is inefficient. ASCII-formatted Waveform (WAVFRM?) and Curve (CURVE?) queries, exceeding 1 M points, are not supported.

Binary Data. Binary data is 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. and positive integer values range from 0 to 65,535.

Table 2-33: Binary data ranges

Byte width	Signed integer range	Positive integer range
1	-128 to 127	0 to 255
2	32,768 to 32,767	0 to 65,535

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 and 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. You can use the DATa:STARt and DATa:STOP commands to specify the first and last data points of the waveform record.

When transferring data into the oscilloscope, you must first specify the record length of the destination waveform record. You do this with the WFMInpre:NR_Pt

command. Next, specify 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. The oscilloscope will ignore the value set by DATa:STOP when reading in data. It will stop reading data when there is no data to read or when it has reached the specified record length.

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

Waveform Data Locations and Memory Allocation

The DATa:SOUrce command specifies the waveform source when transferring a waveform from the oscilloscope. You can only transfer one waveform at a time. Waveforms sent to the oscilloscope are always stored in one of the reference memory locations. Use the DATa:DESTination command to specify a reference memory location.

Waveform Preamble

Each waveform you transfer has an associated waveform preamble, which contains information such as horizontal scale, vertical scale, and the other settings in effect when the waveform was created. Refer to the individual WFMInpre and WFMOutpre commands for information.

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.

Transferring Waveform Data from the Oscilloscope

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

1. Select the waveform source(s) using DATa:SOUrce.
2. Specify the waveform data format using DATa:ENCdg.
3. Specify the number of bytes per data point using WFMOutpre:BYT_Nr.
4. Specify the portion of the waveform that you want to transfer using DATa:STARt and DATa:STOP.
5. Transfer waveform preamble information using the WFMOutpre? query.
6. Transfer waveform data from the oscilloscope using the CURVe? query.

Transferring Waveform Data to the Oscilloscope

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

1. Specify the reference waveform using DATa:DESTination.
2. Specify the record length of the reference waveform using WFMPre:NR_Pt.
3. Specify the waveform data format using WFMInpre:ENCdg.
4. Specify the number of bytes per data point using WFMInpre:BYT_Nr.
5. Specify first data point in the waveform record using DATa:STARt.
6. Transfer waveform preamble information using WFMInpre.
7. Transfer waveform data to the oscilloscope using CURVe.

Table 2-34: Waveform Transfer Commands

Command	Description
CURVe	The command format transfers waveform data to the oscilloscope (reference waveform specified by DATa:DESTination) The query format transfers waveform data from oscilloscope specified by the DATa:SOUrce command
DATa	Sets the format and location of the waveform data that is transferred with the CURVe Command Or returns the format and location of the waveform data that is transferred with the CURVe? command
DATa:COMPosition	Sets or returns the type of data that the CURVe? query returns
DATa:COMPosition:AVAILable?	Lists the waveform data types that are available for return from the instrument under the current instrument settings
DATa:DESTination	Sets or returns the reference waveform for storing waveform data sent to the oscilloscope
DATa:ENCdg	Sets or returns the format of outgoing waveform data
DATa:RESOlution	Sets or returns whether the CURVE query returns full resolution records (acquired data) or reduced resolution records (filtered/displayed data).
DATa:SOUrce	Sets or returns the location of waveform data transferred from the oscilloscope
DATa:STARt	Sets or returns the starting point in waveform transfer

Table 2-34: Waveform Transfer Commands (cont.)

Command	Description
DATa:STOP	Sets or returns the ending data point in waveform transfer
WAVFrm?	Returns a branch query containing waveform data in either binary or ASCII format, waveform formatting data, and the location of the waveform data source
WFMInpre?	Returns the waveform formatting specification to be applied to the next incoming CURVE command data
WFMInpre:BIT_Nr	Sets or returns the number of bits per binary waveform point for the incoming waveform
WFMInpre:BN_Fmt	Sets or returns the format of binary data for the incoming waveform
WFMInpre:BYT_Nr	Sets or returns the data width for the incoming waveform
WFMInpre:BYT_Or	Sets or returns the byte order of waveform points for the incoming waveform
WFMInpre:COMPosition	Sets and queries the type of waveform data to be transferred to the instrument with the CURVe? command.
WFMInpre:FILTERFreq	Sets or returns the byte order of waveform points for the outgoing waveform
WFMInpre:ENCdg	Sets or returns the type of encoding for incoming waveform data
WFMInpre:NR_Pt	Sets or returns the number of points in the incoming waveform record
WFMInpre:PT_Fmt	Sets or returns the point format of incoming waveform data
WFMInpre:PT_Off	This query always returns a 0
WFMInpre:XINcr	Sets or returns the horizontal sampling interval between incoming waveform points
WFMInpre:XUNit	Sets or returns the horizontal units of the incoming waveform
WFMInpre:XZEro	Sets or returns the time of the first point in the incoming waveform
WFMInpre:YMUlt	Sets or returns the vertical scale factor, per digitizing level, of the incoming waveform points
WFMInpre:YOFf	Sets or returns the vertical position of the incoming waveform in digitizing levels
WFMInpre:YUNit	Sets or returns the vertical units of the incoming waveform

Table 2-34: Waveform Transfer Commands (cont.)

Command	Description
WFMInpre:YZEro	Sets or returns the vertical offset of the incoming waveform
WFMOutpre?	Returns the waveform formatting data for the waveform specified by the DATa:SOUrce command
WFMOutpre:BYT_Or	Sets or returns which byte of binary waveform data is transmitted first, during a waveform data transfer, when data points require more than one byte.
WFMOutpre:BIT_Nr	Sets or returns the number of bits per waveform point that outgoing waveforms contain
WFMOutpre:BN_Fmt	Sets or returns the format of binary data for the outgoing waveform
WFMOutpre:BYT_Nr	Sets or returns the data width for the outgoing waveform
WFMOutpre:COMPosition	Controls or indicates the type of data used by the CURVe? query.
WFMOutpre:ENCdg	Sets or returns the type of encoding for outgoing waveforms
WFMOutpre:FILTERFreq?	Specifies or returns the FilterVu low pass filter frequency, which was applied to the waveform being sent to the instrument.
WFMOutpre:NR_Pt?	Returns the number of points for the waveform transmitted in response to a CURVe? query
WFMOutpre:PT_Fmt?	Returns the point format for the outgoing waveform
WFMOutpre:PT_Off?	This query always returns a 0 if the waveform specified by DATA:SOUrce is on or displayed
WFMOutpre:PT_ORder?	This query always returns LINEAR.
WFMOutpre:RECOrdlength?	Returns the record length for the source waveform as specified by the DATa:SOUrce command.
WFMOutpre:WFId?	Returns a string describing the acquisition parameters for the outgoing waveform
WFMOutpre:XINcr?	Returns the horizontal sampling interval for the outgoing waveform
WFMOutpre:XUNit?	Returns the horizontal units for the outgoing waveform
WFMOutpre:XZEro?	Returns the time of the first point in the outgoing waveform

Table 2-34: Waveform Transfer Commands (cont.)

Command	Description
WFMOutpre:YMUlt?	Returns the vertical scale factor per digitizing level for the outgoing waveform
WFMOutpre:YOFf?	Returns the vertical position in digitizing levels for the outgoing waveform
WFMOutpre:YUNit?	Returns the vertical units for the outgoing waveform
WFMOutpre:YZEro?	Returns the vertical offset for the outgoing waveform

For more information, refer to Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instructions.(See page B-1.)

Zoom Command Group

Use the commands in the Zoom Command Group to expand and position the waveform display horizontally and vertically, without changing the time base.

Table 2-35: Zoom Commands

Command	Description
ZOOm?	Returns the current horizontal positioning and scaling of the display
ZOOm{:MODe|:STATE}	Sets or returns the zoom mode
ZOOm:ZOOM1?	Returns the current horizontal positioning and scaling of the display
ZOOm:ZOOM1:FACtor?	Returns the zoom factor of a particular zoom box
ZOOm:ZOOM1:HORizontal:POSition	Sets or returns the horizontal zoom position for the specified waveform in the specified zoom
ZOOm:ZOOM1:HORizontal:SCAle	Sets or returns the horizontal zoom scale of the specified waveform in the specified zoom
ZOOm:ZOOM1:POSition	Sets or returns the horizontal zoom position for the specified waveform in the specified zoom
ZOOm:ZOOM1:SCAle	Sets or returns the horizontal zoom scale of the specified waveform in the specified zoom
ZOOm:ZOOM1:STATE	Specifies or returns a trace as zoomed, on or off

Commands Listed in Alphabetical Order

ACQuire? (Query Only)

Returns the following current acquisition parameters:

- Stop after
- Acquisition state
- Mode
- Number of averages
- Sampling mode

Group Acquisition

Syntax `ACQuire?`

Related Commands ACQuire:MODe, ACQuire:NUMACq?, ACQuire:NUMAVg, ACQuire:STOPAfter

Examples `ACQuire?` might return `ACQUIRE:STOPAFTER RUNSTOP;STATE 1;MODE SAMPLE;NUMAVG 16;SAMPLINGMODE RT`

ACQuire:MAXSamplerate? (Query Only)

Returns the maximum real-time sample rate, which is the same across all MSO/DPO2000/B series models.

Group Acquisition

Syntax `ACQuire:MAXSamplerate?`

Examples `ACQUIRE:MAXSAMPLERATE?` might return 1.0000E+9 in a DPO2024 instrument indicating that the maximum real-time sample rate is 1 GS/s.

ACQuire:MODe

Sets or returns the acquisition mode of the oscilloscope for all live waveforms.

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 oscilloscope sampling system always samples at the maximum rate, so the acquisition interval may include than one sample.

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

Group Acquisition

Syntax

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

Related Commands ACQuire:NUMAVg, CURVe

Arguments

`SAMple` specifies that the displayed data point value is the first sampled value that is 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, in which the resulting waveform shows an average of SAMple data points from several separate waveform acquisitions. The oscilloscope processes the number of waveforms you specify into the acquired waveform, creating a running exponential average of the input signal. The number of waveform acquisitions that go into making up the average waveform is set or queried using the ACQuire:NUMAVg command.

Examples

`ACQUIRE:MODE AVErage` sets an average acquisition mode indicating that the resulting waveform is the average of the specified number of waveform acquisitions.

`ACQUIRE:MODE?` might return`:ACQUIRE:MODE AVERAGE` indicating that the displayed waveform is the average of the specified number of waveform acquisitions.

ACQuire:NUMACq? (Query Only)

The acquisition number will continue to increase while acquisitions are running until there is a reset. Starting and stopping acquisitions do not cause this number to reset. For example, if acquisitions are running, the acquisition count will increase (assuming the oscilloscope is triggering). If you stop the acquisitions, the acquisition number will freeze at a given number (For example: 5000). If you

start acquisitions again, it will continue from 5000. The number will reset to 0 only if you change the horizontal scale while acquisitions are running.

Group Acquisition

Syntax

```
ACQuire:NUMACq?
```

Related Commands ACQuire:STATE

Returns `ACQuire:NUMACq?` might return `:ACQUIRE:NUMACQ 350` indicating that 350 acquisitions have occurred.

ACQuire:NUMAVg

Sets or returns the number of waveform acquisitions that make up an averaged waveform. Use the ACQuire:MODe command to enable the Average mode. Sending this command is equivalent to turning a multipurpose knob to enter the number of waveform acquisitions to average.

Group Acquisition

Syntax

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

Related Commands ACQuire:MODe

Arguments `<NR1>` is the number of waveform acquisitions to average. The range of values is from 2 to 512 in powers of two.

Examples `ACQUIRE:NUMAVG 16` specifies that 16 waveform averages will be performed before exponential averaging starts.

`ACQUIRE:NUMAVG?` might return `:ACQUIRE:NUMAVG 64` indicating that there are 64 acquisitions specified for averaging.

ACQuire:STATE

Starts or stops acquisitions. When state is set to ON or RUN, a new acquisition will be started. If the last acquisition was a single acquisition sequence, a new

single sequence acquisition will be started. If the last acquisition was continuous, a new continuous acquisition will be started.

If RUN is issued in the middle of completing a single sequence acquisition (for example, averaging or enveloping), the acquisition sequence is restarted, and any accumulated data is discarded. Also, the oscilloscope resets the number of acquisitions. If the RUN argument is issued while in continuous mode, acquisition continues.

Group

Acquisition

Syntax

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

Related Commands

ACQuire:STOPAfter

Arguments

`OFF` stops acquisitions.

`STOP` stops acquisitions.

`ON` starts acquisitions.

`RUN` starts acquisitions.

`<NR1>` = 0 stops acquisitions; any other value starts acquisitions.

Examples

`ACQUIRE:STATE RUN` starts the acquisition of waveform data.

`ACQUIRE:STATE?` might return `:ACQUIRE:STATE 0` indicating that the acquisition is stopped.

ACQuire:STOPAfter

Sets or returns whether the oscilloscope continually acquires acquisitions or acquires a single sequence.

Group Acquisition

Syntax

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

Related Commands ACQuire:STATE

Arguments `RUNSTop` specifies that the oscilloscope will continually acquire data, if ACQuire:STATE is turned on.

`SEQuence` specifies that the next acquisition will be a single-sequence acquisition.

Examples `ACQUIRE:STOPAFTER RUNSTop` sets the oscilloscope to continually acquire data.

`ACQUIRE:STOPAFTER?` might return `:ACQUIRE:STOPAFTER SEQUENCE` indicating that the next acquisition the oscilloscope makes will be of the single-sequence type.

ALIas

Sets or returns the state of alias functionality. Use Alias commands to define new commands as a sequence of standard commands. You may find this useful when repeatedly using the same commands to perform certain tasks like setting up measurements. Aliases are similar to macros but do not include the capability to substitute parameters into alias bodies.

To use Alias commands, first define the alias, then turn on the alias state.

Group Alias

Syntax

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

Related Commands ALIas:DEFine

ALIas[:STATE]

Arguments

`OFF` turns alias expansion off. If a defined alias is sent when ALIas is off, a command error (110) will be generated.

`ON` turns alias expansion on. When a defined alias is received, the specified command sequence is substituted for the alias and executed.

`<NR1>` = 0 disables alias mode; any other value enables alias mode.

Examples

`ALIAS ON` turns the alias feature on. When a defined alias is received, the specified command sequence is substituted for the alias and executed.

`ALIAS?` might return `:ALIAS 1` indicating that the alias feature is 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?
```

Examples

`ALIAS:CATALOG?` might return the string `:ALIAS:CATALOG "SETUP1","TESTMENU1","DEFAULT"` showing that there are three 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 that ALIas:STATE has been turned on. The query form of this command returns the definitions of a selected alias.

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

Group

Alias

Syntax

```
ALIas:DEFine <QString><,>{<QString>|<Block>}
ALIas:DEFine?  <QString>
```

Related Commands ALIas[:STATE]

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 less than or equal to 12 characters.

The second<QString> or <Block> is a complete sequence of program messages.

The messages can contain only valid commands that must be separated by semicolons and must follow all rules for concatenating commands. The sequence must be less than or equal to 256 characters.

Examples `ALIAS:DEFINE "ST1",":RECALL:SETUP 5;:AUTOSET EXECUTE;:SELECT:CH1 ON"` defines an alias named "ST1" that sets up the oscilloscope.

`ALIAS:DEFINE? "ST1"` returns `:ALIAS:DEFINE "ST1",#246 :RECALL:SETUP 5;:AUTOSET EXECUTE;:SELECT:CH1 ON`

ALIas:DELEte (No Query Form)

Removes a specified alias and is identical to ALIas:DELEte:NAMe. An error message is generated if the named alias does not exist.

Group Alias

Syntax `ALIas:DELEte <QString>`

Related Commands *ESR?, ALIas:DELEte:ALL

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

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`

Related Commands ALIas:DELEte, ALIas:DELEte[:NAMe]

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

ALIas:DELEte[:NAMe] (No Query Form)

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

Group Alias

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

Arguments `<QString>` is the name of the alias to remove. Using `ALIas:DELEte[:NAMe]` without specifying an alias causes an execution error. `<QString>` must be an existing 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] {<NR1>|OFF|ON}
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) is generated.

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

Examples

`ALIAS[:STATE] OFF` turns the command alias feature off.

`ALIAS[:STATE]?` returns 0 when the alias feature is off.

ALLEv? (Query Only)

Prompts the oscilloscope to return all events and their messages (delimited by commas), and removes the returned events from the Event Queue. Use the *ESR? query to enable the events to be returned. This command is similar to repeatedly sending *EVMsg? queries to the oscilloscope.

Group

Status and Error

Syntax

`ALLEv?`

Related Commands

*ESR?, EVMsg?

Examples

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

AUTOSet (No Query Form)

Sets the vertical, horizontal, and trigger controls of the oscilloscope to automatically acquire and display the selected waveform.

Group

Miscellaneous

Syntax

`AUTOSet {EXECute|UNDo}`

Arguments

`EXECute` autosets the displayed waveform.

`UNDo` restores the oscilloscope settings to those present prior to the autoset execution.

Examples

`AUTOSET EXECUTE` vertical, horizontal, and trigger controls of the oscilloscope to automatically acquire and display the selected waveform.

AUTOSet:ENAble

Enables or disables the autoset.

Group Miscellaneous

Syntax

```
AUTOSet:ENAble {OFF|ON}
AUTOSet:ENAble?
```

Related Commands AUTOSet

Arguments

`ON` enables the autoset feature.

`OFF` disables the autoset feature.

Examples

`AUTOSET:ENABLE ?` might return `AUTOSET:ENABLE 1` indicating that autoset is enabled.

`AUTOSET:ENABLE OFF` disables autoset.

AUXin? (Query Only)

Returns all auxiliary input parameters.

Group Vertical

Syntax

```
AUXin?
```

AUXin:PRObe

Returns all information concerning the probe attached to auxiliary input.

Group Vertical

Syntax

```
AUXin:PRObe
AUXin:PRObe?
```

Examples `AUXIN:PROBE?` might return `AUXIN:PROBE:ID:TYPE "No Probe Detected";SERNUMBER "";:AUXIN:PROBE:UNITS "";RESISTANCE 1.0000E+6` giving information about the probe attached to the AUX In input.

AUXin:PRObe:AUTOZero (No Query Form)

Sets the TekVPI probe attached to the Aux In input to autozero. The oscilloscope will ignore this command if the Auxiliary input does not have a TekVPI probe connected to it.

Group Vertical

Syntax `AUXin:PRObe:AUTOZero {EXECute}`

Arguments `EXECute` sets the probe to autozero.

Examples `AUXin:PROBE:AUTOZERO EXECUTE`

AUXin:PRObe:COMMAND (No Query Form)

Sets the state of the probe control specified with the first argument to the state specified with the second argument. The commands and states are unique to the attached probe type. Only certain VPI probes support this command. See the probe documentation for how to set these string arguments.

Group Vertical

Syntax `AUXin:PRObe:COMMAND <QString>, <QString>`

Arguments `<QString>` are quoted strings specifying the probe command and value to set in the probe attached to the auxiliary input.

Examples `AUXIN:PROBE:COMMAND “OUTPUT”, “ON”` turns the output of a Tektronix VPI-DPG probe on.

`AUXIN:PROBE:COMMAND “MODE”, “4–4V1MHz”` sets a Tektronix VPI-DPG probe to the 4-4V1MHz mode.

`AUXIN:PROBE:COMMAND?“MODE”` might return `AUXIN:PROBE:COMMAND “MODE”,“4-4V1MHZ”`.

AUXin:PRObe:DEGAUss (No Query Form)

Starts a degauss/autozero cycle on a TekVPI current probe attached to the Aux In input. If you send this command to a probe that does not support this function, it is ignored

Group Vertical

Syntax `AUXin:PRObe:DEGAUss {EXECute}`

Arguments `EXECute` starts a probe degauss cycle.

Examples `AUXin:PROBE:DEGAUSS EXECUTE` degausses the probe attached to the Aux In input.

AUXin:PRObe:DEGAUss:STATE? (Query Only)

Returns the state of the probe degauss (`NEEDED`, `RECOMMENDED`, `PASSED`, `FAILED`, `RUNNING`). The command will return `PASSED` for probes that do not support degauss operations.

Group Vertical

Syntax `AUXin:PRObe:DEGAUss:STATE?`

Examples `AUXin:PROBE:DEGAUSS:STATE?` might return:
`AUXin:PROBE:DEGAUSS:STATE PASSED` indicating that the probe has been degaussed.

AUXin:PRObe:FORCEDRange

Changes or returns the range on a TekVPI probe attached to the Aux In input.

Group Vertical

Syntax
```
AUXin:PRObe:FORCEDRange <NR3>
AUXin:PRObe:FORCEDRange?
```

Arguments `<NR3>` is the probe range, which is probe dependent.

AUXin:PRObe:GAIN

Sets or returns the gain factor of a probe that is attached to the Aux In input.

Group Vertical

Syntax

```
AUXin:PRObe:GAIN <NR3>
AUXin:PRObe:GAIN?
```

Arguments `<NR3>` is the probe gain, which is probe dependent.

Examples `AUXin:PROBE:GAIN?` might return `:AUXin:PROBE:GAIN 100.0000E-3` indicating that the attached 10x probe delivers 0.1 V to the Aux In BNC for every 1.0 V applied to the probe input.

AUXin:PRObe:ID:SERnumber? (Query Only)

Returns the serial number of the probe that is attached to the auxiliary input.

Group Vertical

Syntax

```
AUXin:PRObe:ID:SERnumber?
```

AUXin:PRObe:ID:TYPE? (Query Only)

Returns the type of probe that is attached to the auxiliary input.

Group Vertical

Syntax

```
AUXin:PRObe:ID:TYPE?
```

AUXin:PRObe:RESistance? (Query Only)

Returns the resistance of the probe attached to the front panel Aux In connector.

Group Vertical

Syntax `AUXin:PRObe:RESistance?`

Examples `AUXin:PRObe:RESistance?` might return `:AUXin:PROBE:RESISTANCE 1.0000E+6` indicating that the input resistance of the probe attached to the front panel Aux In connector is 1 MΩ.

NOTE. *This query will return 0.0 if no probe is attached or the attached probe does not report the input resistance.*

AUXin:PRObe:SIGnal

This command changes the input bypass setting on VPI probes that support input bypass, for example the TCP0001. If sent to a probe that does not support input bypass, it is ignored.

Group Vertical

Syntax
```
AUXin:PRObe:SIGnal {BYPass|PASS}
AUXin:PRObe:SIGnal?
```

Arguments `BYPass` sets the probe to Bypass mode.

`PASS` sets the probe to Pass mode.

AUXin:PRObe:UNIts? (Query Only)

Returns a string describing the units of measure of the probe attached to the Aux In input.

Group Vertical

Syntax `AUXin:PRObe:UNIts?`

Examples `AUXin:PROBE:UNITS?` might return: `:AUXin:PROBE:UNITS "V"` indicating that the units of measure for the attached probe are volts.

BUS

Sets or returns the parameters for each bus. These parameters affect either the Serial Trigger Setup or the Bus Display.

Conditions This command requires a DPO2AUTO, DPO2EMBD, or DPO2COMP application module.

Group Bus

Syntax

```
BUS
```

BUS:B<x>:CAN:BITRate

Sets or returns the bit rate for CAN bus <x>, where x is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:CAN:BITRate <NR1>
BUS:B<x>:CAN:BITRate?
```

Arguments `<NR1>` sets the bit rate to the closest bit rate supported by the instrument.

Returns The query always returns the numerical bit rate value.

BUS:B<x>:CAN:PRObe

Sets or returns the probing method to probe CAN bus <x>, where x is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:CAN:PRObe {CANH|CANL|RX|TX|DIFFerential}
BUS:B<x>:CAN:PRObe?
```

Arguments

`CANH` specifies the single-ended CANH signal, as specified by the CAN standard.

`CANL` specifies the single-ended CANL signal, as specified by the CAN standard.

`RX` specifies the receive signal on the bus side of the CAN transceiver.

`TX` specifies the transmit signal.

`DIFFerential` specifies the differential CAN signal.

BUS:B<x>:CAN:SAMPLEpoint

Sets or returns the sampling point during each bit period for bus <x>, where x is the bus number

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
BUS:B<x>:CAN:SAMPLEpoint <NR1>
BUS:B<x>:CAN:SAMPLEpoint?
```

Arguments

`<NR1>` is the sample point in percent. Values are limited to 25, 30, ... 70, 75.

BUS:B<x>:CAN:SOUrce

Sets or returns the CAN bus data source for bus <x>, where x is the bus number.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
BUS:B<x>:CAN:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:CAN:SOUrce?
```

Arguments

`CH1–CH4` is the analog channel to use as the data source.

`D0–D15` is the digital channel to use as the data source.

BUS:B<x>:DISplay:FORMAt

Sets or returns the display format for the numerical information in the bus waveform <x>, where x is the bus number.

Conditions

This command requires a DPO2AUTO, DPO2EMBD, or DPO2COMP application module.

Group

Bus

Syntax

```
BUS:B<x>:DISplay:FORMAt {BINary|HEXadecimal|ASCII|MIXed}
BUS:B<x>:DISplay:FORMAt?
```

Related Commands

BUS:B<x>:TYPE

Arguments

`BINary` specifies a binary data display.

`HEXadecimal` specifies a hexadecimal data display.

`ASCII` specifies an ASCII format for RS232 only.

`MIXed` specifies a mixed format for LIN only.

BUS:B<x>:I2C:ADDRess:RWINClude

Sets and returns whether the read/write bit is included in the address.

Conditions

This command requires a DPO2EMBD application module.

Group

Bus

Syntax

```
BUS:B<x>:I2C:ADDRess:RWINClude {<NR1>|OFF|ON}
BUS:B<x>:I2C:ADDRess:RWINClude?
```

Arguments

`<NR1>` = 0 does not include the read/write bit in the address; any other value includes the read/write bit in the address.

OFF does not include the read/write bit in the address.

ON includes the read/write bit in the address.

Examples

`BUS:B1:I2C:ADDRESS:RWINCLUDE ON` includes the read/write bit in the address.

`BUS:B1:I2C:ADDRESS:RWINCLUDE?` might return `BUS:B1:I2C:ADDRESS:RWINCLUDE 0` indicating that the read/write bit is not included in the address.

BUS:B<x>:I2C{:CLOCK|:SCLK}:SOUrce

Sets or returns the I2C SCLK source for bus <x>, where x is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Bus

Syntax

```
BUS:B<x>:I2C{:CLOCK|:SCLK}:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:I2C{:CLOCK|:SCLK}:SOUrce?
```

Arguments

`CH1–CH4` specifies the analog channel to use as the I2C SCLK source.

`D0–D15` specifies the digital channel to use as the I2C SCLK source.

BUS:B<x>:I2C{:DATA|:SDATA}:SOUrce

Sets or returns the I2C SDATA source for bus <x>, where x is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Bus

Syntax

```
BUS:B<x>:I2C{:DATA|:SDATA}:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:I2C{:DATA|:SDATA}:SOUrce?
```

Arguments `CH1-CH4` specifies the analog channel to use as the I2C SDATA source.

`D0-D15` specifies the digital channel to use as the I2C SDATA source.

BUS:B<x>:LABel

Sets or returns the waveform label for bus < x>, where x is the bus number 1 through 4.

Group Bus

Syntax

```
BUS:B<x>:LABel <Qstring>
BUS:B<x>:LABel?
```

Arguments `<Qstring>` is an alpha-numeric string of text, enclosed in quotes, that contains the text label information for bus <x>. The text string is limited to 30 characters.

BUS:B<x>:LIN:BITRate

Sets or returns the bit rate for LIN.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:LIN:BITRate <NR1>
BUS:B<x>:LIN:BITRate?
```

Arguments `<NR1>` is the LIN bit rate.

Examples `BUS:B1:LIN:BITRATE 9600` sets the bit rate 9600.

`BUS:B1:LIN:BITRATE?` might return `BUS:B1:LIN:BITRATE 2400` indicating that the bit rate is set to 2400.

BUS:B<x>:LIN:IDFORmat

Sets or returns the LIN ID format.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:LIN:IDFORmat {NOPARity|PARity}
BUS:B<x>:LIN:IDFORmat?
```

Arguments `NOPARity` sets the LIN ID format to no parity.

`PARity` sets the LIN ID format to parity.

Examples `BUS:B1:LIN:IDFORMAT PARITY` sets the LIN ID format to parity.

`BUS:B1:LIN:IDFORMAT?` might return `BUS:B1:LIN:IDFORMAT NOPARITY` indicating that the LIN ID format is no parity.

BUS:B<x>:LIN:POLARity

Sets or returns the LIN polarity.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:LIN:POLARity {NORMal|INVerted}
BUS:B<x>:LIN:POLARity?
```

Arguments `NORMal` specifies normal LIN polarity.

`INVerted` specifies inverted LIN polarity.

Examples `BUS:B1:LIN:POLARITY INVERTED` sets the LIN polarity to INVERTED.

`BUS:B1:LIN:POLARITY?` might return `BUS:B1:LIN:POLARITY NORMAL` indicating that the LIN polarity is normal.

BUS:B<x>:LIN:SAMPLEpoint

Sets or returns the sample point (in %) at which to sample during each bit period.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:LIN:SAMPLEpoint <NR1>
BUS:B<x>:LIN:SAMPLEpoint?
```

Arguments `<NR1>` is the sample point (in %) at which to sample during each bit period.

Examples `BUS:B1:LIN:SAMPLEPOINT 10` sets the sample point is at 10% of the bit period

`BUS:B1:LIN:SAMPLEPOINT?` might return `BUS:B1:LIN:SAMPLEPOINT 50` indicating that the sample point is at 50% of the bit period

BUS:B<x>:LIN:SOUrce

Sets or returns the LIN data source.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:LIN:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:LIN:SOUrce?
```

Arguments `CH<x>` specifies the LIN source channel where x is 1 to 4.

`D<x>` specifies the LIN source channel where x is 0 to 15.

Examples `BUS:B1:LIN:SOURCE CH4` sets the LIN source to channel 4.

`BUS:B1:LIN:SOURCE?` might return `BUS:B1:LIN:SOURCE CH1` indicating that the LIN source is channel 1.

BUS:B<x>:LIN:STANDard

Sets or returns the LIN standard.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
BUS:B<x>:LIN:STANDard {V1X|V2X|MIXed}
BUS:B<x>:LIN:STANDard?
```

Arguments

`V1X` sets the LIN standard to V1X.

`V2X` sets the LIN standard to V2X

`MIXed` sets the LIN standard to MIXED.

Examples

`BUS:B1:LIN:STANDARD V1X` sets the LIN standard is V1X.

`BUS:B1:LIN:STANDARD?` might return `BUS:B1:LIN:STANDARD V2X` indicating that the LIN standard is V2X.

BUS:B<x>:PARallel:BIT<x>:SOUrce

Sets or returns the parallel bit source for B<x>, where x is the bus number and Bit<x> is the bit number.

Group Bus

Syntax

```
BUS:B<x>:PARallel:BIT<x>:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:PARallel:BIT<x>:SOUrce?
```

Arguments

`CH1-CH4` specifies the bit source B<x>. BIT<x> specifies the bit number.

`D0-D15` specifies the bit source B<x>. BIT<x> specifies the bit number.

BUS:B<x>:PARallel:CLOCK:EDGE

Sets or returns the parallel clock edge for bus <x>, where x is the bus number.

Group Bus

Syntax

```
BUS:B<x>:PARallel:CLOCK:EDGE {EITher|RISing|FALling}
BUS:B<x>:PARallel:CLOCK:EDGE?
```

Arguments

`EIther` specifies either rising or falling edge as the clock edge.

`RISing` specifies the rising edge as the clock edge.

`FALling` specifies the falling edge as the clock edge.

BUS:B<x>:PARallel:CLOCK:ISCLOCKed

Sets or returns the parallel bus clock function for bus <x>, where x is the bus number.

Group

Bus

Syntax

```
BUS:B<x>:PARallel:CLOCK:ISCLOCKed {YES|NO}
BUS:B<x>:PARallel:CLOCK:ISCLOCKed?
```

Arguments

`YES` specifies that the parallel bus is clocked.

`NO` specifies that the parallel bus is not clocked.

BUS:B<x>:PARallel:CLOCK:SOUrce

Sets or returns the parallel bus source <x>, where x is the bus number.

Group

Bus

Syntax

```
BUS:B<x>:PARallel:CLOCK:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:PARallel:CLOCK:SOUrce?
```

Arguments

`CH1-CH4` specifies the channel to use as the parallel bit source.

`D0-D15` specifies the digital channel to use as the parallel bit source.

BUS:B<x>:PARallel:WIDth

Sets or returns the number of bits used for the width of the parallel bus <x>, where x is the bus number.

Group

Bus

Syntax

```
BUS:B<x>:PARallel:WIDth <NR1>
BUS:B<x>:PARallel:WIDth?
```

Arguments

<NR1> is the number of bits.

BUS:B<x>:POSition

Sets or returns the position of the bus <x> waveform on the display, where x is the bus number 1 through 4.

Conditions

This command requires a DPO2AUTO, DPO2EMBD, or DPO2COMP application module.

Group

Bus

Syntax

```
BUS:B<x>:POSition <NR3>
BUS:B<x>:POSition?
```

Arguments

<NR3> specifies the position.

BUS:B<x>:RS232C:BITRate

Sets or returns the RS232 bit rate for bus <x>, where x is the bus number.

Conditions

This command requires a DPO2COMP application module.

Group

Bus

Syntax

```
BUS:B<x>:RS232C:BITRate <NR1>
BUS:B<x>:RS232C:BITRate?
```

Arguments

<NR1> is the bit rate in bits-per-second: 50 to 10000000. You can enter any positive integer, and the instrument will coerce the value to the closest supported bit rate.

BUS:B<x>:RS232C:DATABits

Sets or returns the number of RS232 data bits for bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Bus

Syntax

```
BUS:B<x>:RS232C:DATABits {7|8|9}
BUS:B<x>:RS232C:DATABits?
```

Arguments

7 specifies seven bits in the RS232 data frame.

8 specifies eight bits in the RS232 data frame.

9 specifies nine bits in the RS232 data frame.

BUS:B<x>:RS232C:DELIMiter

Sets or returns the RS232 delimiting value for a packet on bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Bus

Syntax

```
BUS:B<x>:RS232C:DELIMiter {NUL|LF|CR|SPace|XFF}
BUS:B<x>:RS232C:DELIMiter?
```

Arguments

NUL specifies 0x00.

LF specifies 0x0A.

CR specifies 0x0D.

XFF specifies 0xFF.

BUS:B<x>:RS232C:DISplaymode

Sets or returns the display mode for the bus <x> display and event table, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Bus

Syntax

```
BUS:B<x>:RS232C:DISplaymode {FRAme|PACKET}
BUS:B<x>:RS232C:DISplaymode?
```

Arguments `FRAme` displays each frame as a single entity.

`PACKET` displays a group of frames terminated with a single frame defined by the `BUS:B<x>:RS232C:DELImiter` command or the front panel.

BUS:B<x>:RS232C:PARity

Sets or returns the RS232C parity for bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Bus

Syntax

```
BUS:B<x>:RS232C:PARity {NONe|EVEN|ODD}
BUS:B<x>:RS232C:PARity?
```

Arguments `NONe` specifies no parity.

`EVEN` specifies even parity.

`ODD` specifies odd parity.

NOTE. *9 data bits (BUS:B<x>:DATABITS 9) does not support parity. Changing the parity with the PI while using 9 data bits, has no effect.*

BUS:B<x>:RS232C:POLarity

Sets or returns the RS232C polarity for bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Bus

Syntax

```
BUS:B<x>:RS232C:POLarity {NORMal|INVERTed}
BUS:B<x>:RS232C:POLarity?
```

Arguments

`NORMal` sets the RS232C bus polarity to positive.

`INVERTed` sets the RS232C bus polarity to negative.

BUS:B<x>:RS232C:RX:SOUrce

Sets or returns the RS232 RX source for bus <x>, where x is the bus number.

Conditions

This command requires a DPO2COMP application module.

Group

Bus

Syntax

```
BUS:B<x>:RS232C:RX:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:RS232C:RX:SOUrce?
```

Arguments

`CH1–CH4` specifies the channel to use for the RS232 RX source.

`D0–D15` specifies the digital channel to use for the RS232 RX source.

BUS:B<x>:RS232C:TX:SOUrce

Sets or returns the RS232 TX Source for bus <x>, where x is the bus number.

Conditions

This command requires a DPO2COMP application module.

Group

Bus

Syntax

```
BUS:B<x>:RS232C:TX:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:RS232C:TX:SOUrce?
```

Arguments

`CH1–CH4` specifies the channel to use as the RS232 TX source.

`D0–D15` specifies the digital channel to use as the RS232 TX source.

BUS:B<x>:SPI:BITOrder

Sets or returns the bit order for the specified SPI bus.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:BITOrder {LSB|MSB}
BUS:B<x>:SPI:BITOrder?
```

Arguments

`LSB` specifies least significant bit first

`MSB` specifies most significant bit first

Examples

`BUS:B1:SPI:BITOrder LSB` sets the bit order to LSB for the specified bus.

`BUS:B1:SPI:BITOrder?` might return `BUS:B1:SPI:BITOrder MSB` indicating that the returned bit order for the specified bus is MSB.

BUS:B<x>:SPI{:CLOCK|:SCLK}:POLARity

Sets or returns the SPI SCLK polarity for bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI{:CLOCK|:SCLK}:POLARity {FALL|RISe}
BUS:B<x>:SPI{:CLOCK|:SCLK}:POLARity?
```

Arguments

`FALL` specifies the falling edge.

`RISe` specifies the rising edge.

BUS:B<x>:SPI{:CLOCK|:SCLK}:SOUrce

Sets or returns the SPI SCLK source for bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI{:CLOCK|:SCLK}:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:SPI{:CLOCK|:SCLK}:SOUrce?
```

Arguments `CH1-CH4` is the channel to use as the SPI SCLK source.

`D0-D15` is the digital channel to use as the SPI SCLK source.

BUS:B<x>:SPI:DATA{:IN|:MISO}:POLARity

Sets or returns the SPI MISO polarity for bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:DATA{:IN|:MISO}:POLARity {LOW|HIGH}
BUS:B<x>:SPI:DATA{:IN|:MISO}:POLARity?
```

Arguments `LOW` specifies an active low polarity.

`HIGH` specifies an active high polarity.

BUS:B<x>:SPI:DATA{:IN|:MISO}:SOUrce

Sets or returns the SPI MISO source for bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:DATA{:IN|:MISO}:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:SPI:DATA{:IN|:MISO}:SOUrce?
```

Arguments `CH1-CH4` is the channel to use as the SPI MISO source.

`D0-D15` is the digital channel to use as the SPI MISO source.

BUS:B<x>:SPI:DATA{:OUT|:MOSI}:POLARity

Sets or returns the SPI MOSI polarity for bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:DATA{:OUT|:MOSI}:POLARity {LOW|HIGH}
BUS:B<x>:SPI:DATA{:OUT|:MOSI}:POLARity?
```

Arguments `LOW` specifies the active low polarity.

`HIGH` specifies the active high polarity.

BUS:B<x>:SPI:DATA{:OUT|:MOSI}:SOUrce

Sets or returns the SPI MOSI source for bus <x, where x is the bus number>.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:DATA{:OUT|:MOSI}:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:SPI:DATA{:OUT|:MOSI}:SOUrce?
```

Arguments `CH1-CH4` is the channel to use as the SPI MISO source.

`D0-D15` is the digital channel to use as the SPI MISO source.

BUS:B<x>:SPI:DATA:SIZe

Sets or returns the number of bits per word for the specified SPI bus.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:DATA:SIZe <NR1>
BUS:B<x>:SPI:DATA:SIZe?
```

Arguments `<NR1>` specifies the number of bits per word.

Examples `BUS:B1:SPI:DATA:SIZe 16` sets the data size to 16.

`BUS:B1:SPI:DATA:SIZe?` might return `BUS:B1:SPI:DATA:SIZe 8` indicating that the data size is 8 bits.

BUS:B<x>:SPI:FRAMING

Sets or returns the type of SPI framing.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:FRAMING {SS|IDLEtime}
BUS:B<x>:SPI:FRAMING?
```

Arguments `SS` specifies framing by SS (non 2-wire).

`IDLEtime` specifies framing by Idle Time (2-wire).

Examples `BUS:B1:SPI:FRAMING SS` sets the SPI framing type to SS.

`BUS:B1:SPI:FRAMING IDLETIME` might return `BUS:B1:SPI:FRAMING IDLETIME` indicating that the SPI framing type is set to IDLETIME.

BUS:B<x>:SPI:IDLETime

Sets or returns the SPI bus idle time in seconds for the specified SPI bus.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI:IDLETime <NR3>
BUS:B<x>:SPI:IDLETime?
```

Arguments `<NR3>` specifies the idle time in seconds.

Examples `BUS:B1:SPI:IDLETime?` might return `BUS:B1:SPI:IDLETime 5.0000E-6` indicating that the idle time is 5 µs.

BUS:B<x>:SPI{:SELect|:SS}:POLARity

Sets or returns the SPI SS polarity for bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI{:SELect|:SS}:POLARity {LOW|HIGH}
BUS:B<x>:SPI{:SELect|:SS}:POLARity?
```

Arguments `LOW` specifies an active low polarity.

`HIGH` specifies an active high polarity.

BUS:B<x>:SPI{:SELect|:SS}:SOUrce

Sets or returns the SPI SS source for bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Bus

Syntax

```
BUS:B<x>:SPI{:SELect|:SS}:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
BUS:B<x>:SPI{:SELect|:SS}:SOUrce?
```

Arguments

`CH1–CH4` is the channel to use as the SPI SS source.

`D0–D15` is the digital channel to use as the SPI SS source.

BUS:B<x>:STATE

Sets or returns the on/off state of bus <x>, where x is the bus number.

Group

Bus

Syntax

```
BUS:B<x>:STATE {<NR1>|OFF|ON}
BUS:B<x>:STATE?
```

Related Commands

SELect:BUS<x>

Arguments

`ON` or `<NR1>` ≠ 0 turns on the bus.

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

BUS:B<x>:TYPE

Sets or returns the bus type for <x>, where x is the bus number.

Group

Bus

Syntax

```
BUS:B<x>:TYPE {I2C|SPI|CAN|RS232C|PARallel|LIN}
BUS:B<x>:TYPE
```

Arguments

`I2C` specifies the Inter-IC bus.

`SPI` specifies the Serial Peripheral Interface bus (not available on two-channel models).

`CAN` specifies the Controller Area Network bus.

`RS232C` specifies the RS232C bus.

`PARallel` specifies the parallel bus.

`LIN` specifies the LIN bus.

BUS:LOWerthreshold:CH<x>

Sets the lower threshold for each channel. This applies to all search and trigger types that use the channel. This command supersedes the `:BUS:THReshold:CH<x>` above.

Group Bus

Syntax

```
BUS:LOWerthreshold:CH<x> {<NR3>|ECL|TTL}
BUS:LOWerthreshold:CH<x>?
```

Arguments

`<NR3>` specifies the threshold in volts.

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

Examples

`BUS:LOWERTHRESHOLD:CH1 TTL` sets the CH1 lower threshold to 800 mV.

`BUS:LOWERTHRESHOLD:CH1?` might return `:BUS:LOWERTHRESHOLD:CH1 -800.0000E-3` indicating that the CH1 lower threshold is -800 mV.

BUS:THReshold:CH<x>

Sets or returns the threshold for analog channel <x>, where x is the channel number. This setting applies to all trigger types that use the channel. The threshold value of CH<x> is set to the closest value used by hardware. The threshold settings for both BUS: THReshold:CH<x> and BUS:THRESHOLD:D<x> channels will be rounded to the nearest threshold value used by hardware and not necessarily the value set by the user. For example: If you set the threshold to 2.48 volts and the hardware only supports 2.5 volts, then the threshold value will be 2.5 volts and not 2.48 as set by you.

Conditions This command requires a DPO2AUTO, DPO2EMBD, or DPO2COMP application module.

Group Bus

Syntax

```
BUS:THReshold:CH<x> {ECL|TTL|<NR3>}
BUS:THReshold:CH<x>?
```

Related Commands TRIGger:A:LEVel:CH<x>

Arguments

ECL specifies a preset ECL high level of –1.3 V.

TTL specifies a TTL preset high level of 1.4 V.

<NR3> specifies the threshold level in volts.

BUS:THReshold:D<x>

Sets or returns the threshold for digital channel <x>, where x is the digital channel number. This will apply to all search and trigger types that use the channel. The threshold value of D<x> is set to the closest value used by hardware.

Digital channel thresholds are set in two groups. One for channels D0–D7 and the other for D8–D15. Setting the thresholds in one of these groups sets the threshold for all the channels in that group.

Conditions

This command requires a DPO2AUTO, DPO2EMBD, or DPO2COMP application module.

Group

Bus

Syntax

```
BUS:THReshold:D<x> {<NR3>|ECL|TTL}
BUS:THReshold:D<x>?
```

Related Commands

TRIGger:A:LEVel:D<x>

Arguments

ECL specifies a preset ECL high level of –1.3 V.

TTL specifies a preset TTL high level of 1.4 V.

<NR3> specifies the threshold level in volts.

BUS:UPPerthreshold:CH<x>

Sets the upper threshold for each channel. This applies to all search and trigger types that use the channel.

Group

Bus

Syntax

```
BUS:UPPerthreshold:CH<x> {<NR3>|ECL|TTL}
BUS:UPPerthreshold:CH<x>?
```

Arguments

<NR3> specifies the threshold in volts.

ECL specifies a preset ECL high level of –1.3 V.

TTL specifies a preset TTL high level of 1.4 V.

Examples

BUS:UPPERTHRESHOLD:CH1 800.0000E-3 sets the CH1 upper threshold to 800 mV.

BUS:UPPERTHRESHOLD:CH1? might return :BUS:UPPERTHRESHOLD:CH1 -800.0000E-3 indicating that the CH1 upper threshold is set to -800 mV.

BUSY? (Query Only)

Returns the status of the oscilloscope. This command allows you to synchronize the operation of the oscilloscope with your application program.

Group

Status and Error

Syntax

BUSY?

Related Commands

*OPC, *WAI

Returns

<NR1> = 0 means the oscilloscope is not busy processing a command whose execution time is extensive.

<NR1> = 1 means the oscilloscope is busy processing one of the commands listed in the table below.

Commands that affect BUSY? response

Operation	Command
Single sequence acquisition	ACQuire:STATE ON or ACQuire:STATE RUN or ACQuire:STATE1 (when ACQuire:STOPAfter is set to SEQuence)
Hard copy operation	HARDCopy STArt
Calibration step	Refer to the optional oscilloscope Service Manual.

Examples

BUSY? might return :BUSY 1 indicating that the oscilloscope is currently busy.

See the section titled “Using the BUSY Query” for an example of how to use this query.

*CAL? (Query Only)

Performs an internal self-calibration and returns the oscilloscope calibration status.

NOTE. *Disconnect or otherwise remove all input signals prior to starting self-calibration. The self-calibration can take several minutes to complete.*

No other commands are executed until calibration is complete.

Group Calibration and Diagnostic

Syntax `*CAL?`

Returns `<NR1>` = 1 indicates the calibration did not complete successfully.

`<NR1>` = 0 indicates the calibration completed without errors.

Examples `*CAL?` starts the internal signal path calibration and might return 0 to indicate that the calibration was successful.

CALibrate:FACtory (No Query Form)

Provides the controls for starting and stopping the factory calibration process. The factory calibration process consists of a series of steps.

Group Calibration and Diagnostic

Syntax `CALibrate:FACtory {STARt|CONTinue|PREVious|ABOrt|DUmp}`

Arguments `STARt` initializes the factory calibration sequence and starts the first calibration step.

`CONTinue` begins the next factory calibration step.

`PREVious` attempts to run the most recent factory calibration step again.

`ABOrt` stops the calibration process.

`DUmp` stops the calibration and prints the calibration constants.

Examples `CALIBRATE:FACTORY START` starts the calibration process.

CALibrate:FACtory:STATus? (Query Only)

Returns the factory calibration status value saved in nonvolatile memory.

Group Calibration and Diagnostic

Syntax `CALibrate:FACtory:STATus?`

Examples `CALIBRATE:FACTORY:STATUS?` might return `CALIBRATE:FACTORY:STATUS PASS` indicating that factory calibration passed.

CALibrate:INTERNal (No Query Form)

This command starts a signal path compensation.

Group Calibration and Diagnostic

Syntax `CALibrate:INTERNal`

Arguments None

Examples `CALIBRATE:INTERNAL` starts a serial path compensation cycle.

CALibrate:INTERNal:STARt (No Query Form)

Starts the internal signal path calibration (SPC) of the oscilloscope. You can use the CALibrate:INTERNal:STATus? query to return the current status of the internal signal path calibration of the oscilloscope.

Group Calibration and Diagnostic

Syntax `CALibrate:INTERNal:STARt`

Related Commands CALibrate:RESults:SPC?

Examples `CALIBRATE:INTERNAL:START` initiates the internal signal path calibration of the oscilloscope.

CALibrate:INTERNal:STATus? (Query Only)

Returns the current status of the oscilloscope internal signal path calibration for the last SPC operation.

Group Calibration and Diagnostic

Syntax `CALibrate:INTERNal:STATus?`

Related Commands *CAL?

Returns This query will return one of the following:

- `INIT` indicates the oscilloscope has not had internal signal path calibration run.
- `PASS` indicates the signal path calibration completed successfully.
- `FAIL` indicates the signal path calibration did not complete successfully.
- `RUNNING` indicates the signal path calibration is currently running.

Examples `CALIBRATE:INTERNAL:STATUS?` might return `:CALIBRATE:INTERNAL:STATUS INIT` indicating that the current status of the internal signal path calibration is that it has not been run.

CALibrate:RESults? (Query Only)

Returns the status of internal and factory calibrations, without performing any calibration operations. The results returned do not include the calibration status of attached probes. The `CALibrate:RESults?` query is intended to support GO/NoGO testing of the oscilloscope calibration readiness: all returned results should indicate PASS status if the oscilloscope is "fit for duty". It is quite common, however, to use uncalibrated probes (particularly when the oscilloscope inputs are connected into a test system with coaxial cables).

Group Calibration and Diagnostic

Syntax `CALibrate:RESults?`

Related Commands *CAL?

CALibrate:RESults:FACtory? (Query Only)

Returns the status of internal and factory calibration, without performing any calibration operations.

Group Calibration and Diagnostic

Syntax `CALibrate:RESults:FACtory?`

CALibrate:RESults:SPC? (Query Only)

Returns the status of the SPC operation. This query does not initiate a SPC.

Group Calibration and Diagnostic

Syntax `CALibrate:RESults:SPC?`

Related Commands *CAL?

Returns `INIT` indicates that SPC has never successfully completed.

`PASS` indicates that the last SPC operation passed.

`FAIL` indicates that the last SPC operation failed.

`RUNNING` indicates that the SPC operation is running.

Examples `CALIBRATE:RESULTS:SPC?` returns the results of the last SPC operation: either PASS or FAIL.

CALibrate:TEMPerature? (Query Only)

Returns 0 as the MSO/DPO2000/B series instruments do not support recording of oscilloscope temperature.

Group Calibration and Diagnostic

Syntax `CALibrate:TEMPerature?`

CH<x>? (Query Only)

Returns the vertical parameters for channel <x>, where x is the channel number.

Group Vertical

Syntax

```
CH<x>?
```

CH<x>:BANdwidth

Sets or returns the selectable low-pass bandwidth limit filter for channel <x>, where x is the channel number.

Group Vertical

Syntax

```
CH<x>:BANdwidth {TWEnty|FULl|<NR3>}
CH<x>:BANdwidth?
```

Arguments

`TWEnty` sets the upper bandwidth limit of channel <x> to 20 MHz.

`FULl` disables any optional bandwidth limiting. The specified channel operates at its maximum attainable bandwidth.

`<NR3>` is a double-precision ASCII string. The oscilloscope rounds this value to an available bandwidth using geometric rounding, and then uses this value to set the upper bandwidth limit.

NOTE. *Other values may be possible depending on the attached probes.*

Examples `CH1:BANDWIDTH TWENTY` sets the bandwidth of channel 1 to 20 MHz.

CH<x>:COUPling

Sets or returns the input attenuator coupling setting for channel <x>, where x is the channel number.

Group Vertical

Syntax

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

Arguments

`AC` sets channel <x> to AC coupling.

`DC` sets channel <x> to DC coupling.

`GND` sets channel<x> to ground. Only a flat, ground-level waveform will be displayed.

Examples

`CH2:COUPLING GND` sets channel 2 to ground.

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

CH<x>:DESKew

Sets or returns the deskew time for channel <x>, where x is the channel number. You can adjust the deskew time to add an independent, channel-based delay time to the delay (set by the horizontal position control and common to all channels) from the common trigger point to first sample taken for each channel. This lets you compensate individual channels for different delays introduced by their individual input hook ups.

Group

Vertical

Syntax

```
CH<x>:DESKew <NR3>
CH<x>:DESKew?
```

Arguments

`<NR3>` is the deskew time for channel <x>, ranging from -100 ns to +100 ns with a resolution of 1 ns.

Examples

`CH4:DESKew 5.0E-9` sets the deskew time for channel 4 to 5 ns.

`CH2:DESKew?` might return `:CH2:DESKEW 2.0000E-09` indicating that the deskew time for channel 2 is set to 2 ns.

CH<x>:IMPedance

Sets or returns the input impedance of channel <x>, where x is the channel number. This command is provided for compatibilty.

Group

Vertical

Syntax

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

CH<x>:INVert

Sets or returns the invert function for channel <x>, where is the channel number. When on, the invert function inverts the waveform for the specified channel.

NOTE. *This command inverts the waveform for display purposes only. The oscilloscope does not use an inverted waveform for triggers or trigger logic inputs.*

Group Vertical

Syntax

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

Arguments

`OFF` sets the invert function for channel <x> to off.

`ON` sets the invert function for channel <x> to on.

Examples

`CH4:INVert ON` inverts the waveform on channel 4.

`CH2:INVert?` might return `:CH2:INVERT 0` indicating that channel 2 is not inverted.

CH<x>:LABel

Sets or returns the waveform label for channel <x>, where x is the channel number.

Group Vertical

Syntax

```
CH<x>:LABel <Qstring>
CH<x>:LABel?
```

Arguments

`<Qstring>` is an alpha-numeric string of text, enclosed in quotes, that contains the text label information for the channel <x> waveform. The text string is limited to 30 characters.

CH<x>:OFFSet

Sets or returns the vertical offset for channel <x>, where x is the channel number.

This command offsets the vertical acquisition window (moves the level at the vertical center of the acquisition window) for the specified channel. Visualize offset as scrolling the acquisition window towards the top of a large signal for increased offset values, and scrolling towards the bottom for decreased offset values. The resolution of the vertical window sets the offset increment for this control.

Offset adjusts only the vertical center of the acquisition window for channel waveforms to help determine what data is acquired. The oscilloscope always displays the input signal minus the offset value.

The channel offset range depends on the vertical scale factor. The valid ranges for the MSO/DPO2000/B series oscilloscopes are (when the probe and external attenuation factor is X1):

- For V/Div settings from 2 mV/div to 200 mV/div, the offset range is +/– 1 V
- For V/Div settings from 202 mV/div to 5 V/div, the offset range is +/– 25 V

Group Vertical

Syntax

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

Related Commands CH<x>:POSition

Arguments `<NR3>` is the offset value for the specified channel <x>.

Examples `CH3:OFFSet 2.0E-3` sets the offset for channel 3 to 2 mV.

`CH4:OFFSet?` might return `:CH4:OFFSET 1.0000E-03` indicating that the offset for channel 4 is set to 1 mV.

CH<x>:POSition

Sets or returns the vertical position of channel <x>, where x is the channel number. The position value is applied to the signal before it is digitized.

Increasing the position value of a waveform causes the waveform to move up. Decreasing the position value causes the waveform to move down. The position value determines the vertical graticule coordinate at which input signal values, minus the present offset setting for that channel, are displayed. For example,

if the position for Channel 3 is set to 2.0 and the offset is set to 3.0, then input signals equal to 3.0 units are displayed 2.0 divisions above the center of the screen (at 1 V/div).

Group Vertical

Syntax

```
CH<x>:POSition <NR3>
CH<x>:POSition?
```

Related Commands CH<x>:OFFSet, REF<x>:VERTical:POSition, MATH[1]:VERTical:POSition

Arguments `<NR3>` is the position value for channel <x>, in divisions, from the center graticule. The range is 4 to -4 divisions.

Examples `CH2:POSition 1.3` positions the Channel 2 input signal 1.3 divisions above the center graticule.

`CH1:POSition?` might return `:CH1:POSITION -1.3000` indicating that the current position of Channel 1 is 1.3 divisions below the center graticule.

CH<x>:PRObe? (Query Only)

Returns all information concerning the probe attached to channel <x>, where x is the channel number.

Group Vertical

Syntax

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

CH<x>:PRObe:AUTOZero (No Query Form)

Sets the TekVPI probe attached to channel <x> to zero, where x is the channel number

Group Vertical

Syntax

```
CH<x>:PRObe:AUTOZero EXECute
```

Arguments `Execute` auto zeros the probe.

CH<x>:PRObe:COMMAND (No Query Form)

Sets the state of the probe control specified with the first argument to the state specified with the second argument. The commands and states are unique to the attached probe type. Only certain VPI probes support this command. See the probe documentation for how to set these string arguments.

Group Vertical

Syntax `CH<x>:PRObe:COMMAND <QString>, <QString>`

Arguments `<QString>` are quoted strings specifying the probe command and value to set in the probe attached to the specified channel.

Examples `CH1:PROBE:COMMAND "MODE", "4-4V1MHz"` sets a Tektronix VPI-DPG probe to the 4-4V1MHz mode.

`CH1:PROBE:COMMAND "OUTPUT", "ON"` turns the output of a Tektronix VPI-DPG probe on.

`CH1:PROBE:COMMAND?"MODE"` might return `CH1:PROBE:COMMAND "MODE","4-4V1MHZ"`.

CH<x>:PRObe:DEGAUss (No Query Form)

Starts a degauss auto-zero cycle on a TekVPI current probe attached to the input channel specified by <x>, where x is the channel number.

Group Vertical

Syntax `CH<x>:PRObe:DEGAUss EXECute`

Arguments `EXECute` initiates the degauss operation.

CH<x>:PRObe:DEGAUss:STATE? (Query Only)

Returns the state of the probe degauss for the channel specified by <x>, where is x is the channel number.

NOTE. *This command will return* `PASSED` *for probes that do not support degauss operations.*

Group Vertical

Syntax

```
CH<x>:PRObe:DEGAUss:STATE?
```

Returns `NEEDED` indicates the probe should be degaussed before taking measurements.

`RECOMMENDED` indicates the measurement accuracy might be improved by degaussing the probe.

`PASSED` indicates the probe is degaussed.

`FAILED` indicates the degauss operation failed.

`RUNNING` indicates the probe degauss operation is currently in progress.

CH<x>:PRObe:FORCEDRange

Sets or returns the range of a TekVPI probe attached to the channel specified by <x>, where x is the channel number.

NOTE. *This command will return* `PASSED` *for probes that do not support degauss operations.*

Group Vertical

Syntax

```
CH<x>:PRObe:FORCEDRange <NR3>
CH<x>:PRObe:FORCEDRange?
```

Arguments `<NR3>` specifies the range, which is probe specific.

Returns This command returns `0.0` for probes that do no support forced range.

CH<x>:PRObe:GAIN

Sets or returns the gain factor for the probe attached to the channel specified by <x>, where x is the channel number. The "gain" of a probe is the output divided by the input transfer ratio. For example, a common 10x probe has a gain of 0.1.

Group Vertical

Syntax
```
CH<x>:PRObe:GAIN <NR3>
CH<x>:PRObe:GAIN?
```

Related Commands CH<x>:SCAle

Arguments `<NR3>` is the probe gain. Allowed values depend on the specific probe.

Examples `CH2:PROBE:GAIN?` might return `:CH2:PROBE:GAIN 0.1000E+00` indicating that the attached 10x probe delivers 1 V to the channel 2 BNC for every 10 V applied to the probe input.

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

Returns the type and serial number of the probe attached to channel <x>, where x is the channel number.

Group Vertical

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

Examples CH2:PROBE:ID? might return `:CH2:PROBE:ID:TYPE "10X";SERNUMBER "N/A"` indicating that a passive 10x probe of unknown serial number is attached to channel 2.

CH<x>:PRObe:ID:SERnumber? (Query Only)

Returns the serial number of the probe attached to channel <x>, where x is the channel number.

NOTE. *For Level 0 and 1 probes, the serial number will be "".*

Group Vertical

Syntax `CH<x>:PRObe:ID:SERnumber?`

Examples `CH1:PROBE:ID:SERNUMBER?` might return `:CH1:PROBE:ID:SERNUMBER "B010289"` indicating that the serial number of the probe attached to channel 1 is B010289.

CH<x>:PRObe:ID:TYPE? (Query Only)

Returns the type of probe attached to the channel specified by <x>, where x is the channel number. Level 2 (or higher) probes supply their exact product nomenclature; for Level 0 or 1 probes, a generic *"No Probe Detected* message is returned.

Group Vertical

Syntax `CH<x>:PRObe:ID:TYPE?`

Examples CH1:PROBE:ID:TYPE? might return `:CH1:PROBE:ID:TYPE "P6203"` indicating that P6203-type probe is attached to channel 1.

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

Returns the resistance factor of the probe attached to channel <x>, where x is the channel number.

Group Vertical

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

Examples `CH2:PRObe:RESistance?` might return `:CH2:PROBE:RESISTANCE 10.0000E+6` indicating that the input resistance of the probe attached to Channel 2 is 1 MΩ.

NOTE. *This query will return 0.0 if no probe is attached or the attached probe does not report the input resistance.*

CH<x>:PRObe:SIGnal

Sets or returns the input bypass setting of a TekVPI probe attached to channel <x>, where x is the channel number. The probe must support input bypass, for example TCP0001. This command is ignored if sent to an unsupported probe.

Group Vertical

Syntax

```
CH<x>:PRObe:SIGnal {BYPass|PASS}
CH<x>:PRObe:SIGnal?
```

Arguments `BYPass` sets the probe to Bypass mode.

`PASS` sets the probe to Pass mode.

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

Returns a string describing the units of measure for the probe attached to channel <x>, where x is the channel number.

Group Vertical

Syntax

```
CH<x>:PRObe:UNIts?
```

Examples `CH4:PROBE:UNITS?` might return `:CH4:PROBE:UNITS "V"` indicating that the units of measure for the probe attached to channel 4 are volts.

CH<x>:SCAle

Sets or returns the vertical scale for the channel specified by <x>, where x is the channel number.

Each waveform has a vertical scale parameter. For a signal with constant amplitude, increasing the Scale causes the waveform to be displayed smaller. Decreasing the scale causes the waveform to be displayed larger.

Scale affects all waveforms, but affects channel waveforms differently from other waveforms:

- For channel waveforms, this setting controls the vertical size of the acquisition window as well as the display scale. The range and resolution of scale values depends on the probe attached and any other external factors you have specified.
- For reference and math waveforms, this setting controls the display only, graphically scaling these waveforms and having no affect on the acquisition hardware.

Group Vertical

Syntax

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

Related Commands

CH<x>:OFFSet, CH<x>:POSition, REF<x>:VERTical:SCAle, MATH[1]:VERTical:SCAle

Arguments

<NR3> is the vertical channel scale in units-per-division. The value entered here is truncated to three significant digits.

Examples

`CH4:SCALE 100E-03` sets the channel 4 scale to 100 mV per division.

`CH2:SCALE?` might return `:CH2:SCALE 1.0000` indicating that the current scale setting of channel 2 is 1 V per division.

CH<x>:TERmination

Sets the connected-disconnected status of a 50 Ω resistor, which may be connected between the specified channel's coupled input and oscilloscope ground. The channel is specified by <x>. There is also a corresponding query that requests the termination parameter and translates this enumeration into one of the two float values. This command is maintained for compatibility.

Group

Vertical

Syntax

```
CH<x>:TERmination
CH<x>:TERmination?
```

CH<x>:VOLts

Sets or returns the vertical sensitivity for channel <x>, where x is the channel number.

Group

Vertical

Syntax

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

Arguments

<NR3> is the vertical sensitivity, in volts.

CH<x>:YUNits

Sets or returns the units for the channel specified by <x>, where x is the channel number. String arguments are case insensitive and any unsupported units will generate an error.

Supported units are:
%, /Hz, A, A/A, A/V, A/W, A/dB, A/s, AA, AW, AdB, As, B, Hz, IRE, S/s, V, V/A, V/V, V/W, V/dB, V/s, VV, VW, VdB, Volts, Vs, W, W/A, W/V, W/W, W/dB, W/s, WA, WV, WW, WdB, Ws, dB, dB/A, dB/V, dB/W, dB/dB, dBA, dBV, dBW, dBdB, day, degrees, div, hr, min, ohms, percent, s

Group Vertical

Syntax

```
CH<x>:YUNits <QString>
CH<x>:YUNits?
```

Arguments `QString` is a string of text surrounded by quotes, specifying the supported units.

CLEARMenu (No Query Form)

Clears the current menu from the display. This command is equivalent to pressing the front panel Menu off.

Group Miscellaneous

Syntax

```
CLEARMenu
```

*CLS (No Query Form)

Clears the following:

- Event Queue
- Standard Event Status Register
- Status Byte Register (except the MAV bit)

If the *CLS command immediately follows an <EOI>, the Output Queue and MAV bit (Status Byte Register bit 4) are also cleared. The MAV bit indicates that 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.

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

Group Status and Error

Syntax `*CLS`

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

Examples `*CLS` clears the oscilloscope status data structures.

CURSor?

Returns all of the current cursor settings.

Group Cursor

Syntax `CURSor?`

Examples `CURSOR?` might return the following as the current cursor settings:

```
:CURSOR:FUNCTION SCREEN;HBARS:POSITION1 0.0000;POSITION2
0.0000;UNITS BASE;:CURSOR:MODE INDEPENDENT;VBARS:POSITION1
-19.0006E-6;POSITION2 -18.9994E-6;UNITS SECONDS
```

CURSor:FUNCtion

Sets or returns the cursor type. Cursors are attached to the selected waveform in Waveform mode and are attached to the display area in Screen mode.

Group Cursor

Syntax

```
CURSor:FUNCtion {OFF|SCREEN|WAVEform|VBArs|HBArs}
CURSor:FUNCtion?
```

Arguments

`OFF` removes the cursors from the display but does not change the cursor type.

`SCREEN` specifies both horizontal and vertical bar cursors, which measure the selected waveform in horizontal and vertical units. Use these cursors to measure anywhere in the waveform display area.

`WAVEform` specifies paired cursors in YT display format for measuring waveform amplitude and time. In XY and XYZ format, these cursors indicate the amplitude positions of an XY pair (Ch1 vs Ch2 voltage, where Ch1 is the X axis and Ch2 is the Y axis) relative to the trigger.

`VBArs` specifies the vertical bar cursor to measure the selected waveform in vertical units.

`HBArs` specifies the horizontal bar cursor to measure the selected waveform in horizontal units.

Examples

`CURSOR:FUNCTION WAVEFORM` selects the paired cursors for measuring waveform amplitude and time.

`CURSOR:FUNCTION?` might return `:CURSor:FUNCtion SCREEN` indicating that the screen cursors are currently selected.

CURSor:HBArs? (Query Only)

Returns the current settings for the horizontal bar cursors.

Group

Cursor

Syntax

`CURSor:HBArs?`

Examples

`CURSOR:HBARS?` might return the horizontal bar setting as `:CURSOR:HBARS:POSITION1` 320.0000E-03;POSITION2-320.0000E-03;UNITS BASE

CURSor:HBArs:DELTa? (Query Only)

Returns the vertical difference between the two horizontal bar cursors.

Group

Cursor

Syntax

`CURSor:HBArs:DELTa?`

Related Commands CURSor:HBArs:UNIts

Returns A floating point value with an exponent.

Examples `CURSOR:HBARS:DELTA?` might return `:CURSOR:HBARS:DELTA 5.0800E+00` indicating that the difference between the two cursors is 5.08.

CURSor:HBArs:POSITION<x>

Sets or returns the horizontal bar cursor position relative to ground, which is expressed in vertical units (usually volts). The cursor is specified by x, which can be 1 or 2.

Group Cursor

Syntax

```
CURSor:HBArs:POSITION<x> <NR3>
CURSor:HBArs:POSITION<x>?
```

Related Commands CURSor:FUNCtion

Arguments `<NR3>` specifies the cursor position relative to ground.

Examples `CURSOR:HBARS:POSITION1 25.0E-3` positions Cursor 1 of the horizontal cursors at 25 mV.

`CURSOR:HBARS:POSITION2?` might return `:CURSOR:HBARS:POSITION2 -64.0000E-03` indicating that Cursor 2 of the horizontal bar cursors is at -64 mV.

CURSor:HBArs:UNIts

Sets or returns the units for the horizontal bar cursors.

Group Cursor

Syntax

```
CURSor:HBArs:UNIts {BASe|PERcent}
CURSor:HBArs:UNIts?
```

Arguments

`BASe` selects the vertical units for the selected waveform.

`PERCent` selects ratio cursors.

Examples

`CURSOR:HBARS:UNITS` might return `:CURSOR:HBARS:UNITS BASE` indicating that the units for the horizontal bar cursors are base.

CURSor:HBArs:USE (No Query Form)

Sets the horizontal bar cursor measurement scale. This command is only applicable when ratio cursors are on.

Group

Cursor

Syntax

`CURSor:HBArs:USE {CURrent|HALFgrat|FIVEdivs}`

Related Commands

CURSor:HBArs:UNIts

Arguments

`CURrent` sets the H Bar measurement scale so that 0% is the current position of the lowest H Bar cursor and 100% is the current position of the highest H Bar cursor.

`HALFgrat` resets the H bar measurement scale to half the number of divisions (five for some models and four for others) so that 25% is the current position of the lowest H Bar cursor and 75% is the current position of the highest H Bar.

`FIVEdivs` sets H Bar measurement scale so that five screen major divisions is 100%, where 0% is –2.5 divisions and 100% is +2.5 divisions from the center horizontal graticule.

Examples

`CURSOR:HBARS:USE FIVEDIVS` sets the H Bar measurement scale so that 5 screen major divisions equals 100%.

CURSor:MODe

Sets or returns whether the two cursors move linked together in unison or separately. This applies to the Waveform cursors display mode.

Conditions

This command is only applicable when waveform cursors are displayed.

Group

Cursor

Syntax

```
CURSor:MODe {TRACk|INDependent}
CURSor:MODe?
```

Arguments

`TRACk` ties the navigational functionality of the two cursors together. For cursor 1 adjustments, this ties the movement of the two cursors together; however, cursor 2 continues to move independently of cursor 1.

`INDependent` allows independent adjustment of the two cursors.

Examples

`CURSOR:MODE TRACK` specifies that the cursor positions move in unison.

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

CURSor:VBArs? (Query Only)

Returns the current settings for the vertical bar cursors.

Group

Cursor

Syntax

```
CURSor:VBArs?
```

Examples

`CURSOR:VBARS?` might return the following vertical bar settings `:CURSOR:VBARS:UNITS SECONDS;POSITION1` 1.0000E-06;POSITION2 9.0000E-06

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

Returns the alternate readout for the waveform (Vbar) cursors specified by <x>. This alternate readout is in effect for a bus waveform.

Group

Cursor

Syntax

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

Arguments

`X = 1` specifies vertical bar cursor 1.

`X = 2` specifies vertical bar cursor 2.

CURSor:VBArs:DELTa? (Query Only)

Returns the horizontal difference between the two vertical bar cursors. The units are specified by the CURSor:VBArs:UNIts command.

This is equivalent to watching the cursor readout in the display while using the appropriate cursor mode.

Group Cursor

Syntax `CURSor:VBArs:DELTa?`

Related Commands CURSor:VBArs:UNIts

Returns `<NR3>`

Examples `CURSOR:VBARS:DELTA?` might return `:CURSOR:VBARS:DELTa 1.0640E+00` indicating that the time between the vertical bar cursors is 1.064 s.

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

Returns the horizontal value of the specified vertical bar ticks for cursor <x>.

Group Cursor

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

Related Commands CURSor:VBArs:UNIts

Arguments `<x>` specifies the cursor. Valid values are 1 and 2.

Returns `<NR3>` indicates the value of one of the tics. The units are specified by the CURSor:VBArs:UNIts command.

Examples `CURSor:VBArs:HPOS2?` might return CURSOR:VBARS:HPOS2 100E-3, indicating that the value of one vertical bar tick.

CURSor:VBArs:POSITION<x>

Sets or returns the horizontal position for the specified vertical bar cursor. The cursor is specified by <x>, which can be 1 or 2. Values are with respect to trigger position or the zero reference point for the designated waveform (if horizontal units are not set to time). Use the CURSor:VBArs:UNIts command to specify units.

Group Cursor

Syntax

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

Related Commands CURSor:VBArs:UNIts

Arguments `<NR3>` specifies the cursor position.

Returns A floating point value with an exponent.

Examples `CURSor:VBArs:POSITION2 9.00E-6` positions the cursor2 vertical bar cursor at 9 ms.

`CURSor:VBArs:POSITION1?` this command might return `:CURSOR:VBARS:POSITION1 1.0000E-06` indicating that the cursor1 vertical bar is positioned at 1 µs.

CURSor:VBArs:UNIts

Sets or returns the units for the vertical bar cursors.

Group Cursor

Syntax

```
CURSor:VBArs:UNIts {SEConds|HERtz|DEGrees|PERcent}
CURSor:VBArs:UNIts?
```

Arguments `SEConds` sets the units of the vertical bar cursors for the time domain (seconds).

`HERtz` sets the units of the vertical bar cursors for the frequency domain (Hertz).

`DEGrees` sets the units to degrees for use with an XY display.

`PERcent` sets the units to percent for use with ratio cursors.

Returns SECONDS, HERTZ, DEGREES, or PERCENT, depending on the current vertical bar cursor units.

Examples CURSOR:VBARS:UNITS HERtz sets the units of the VBArs cursors to 1/seconds.

CURSOR:VBARS:UNITS? might return :CURSOR:VBARS:UNITS SECONDS indicating that the units for the vertical bar cursor are currently set to seconds.

CURSor:VBArs:USE (No Query Form)

Sets the vertical bar cursor measurement scale.

Conditions This command is only applicable when ratio cursors are on.

Group Cursor

Syntax CURSor:VBArs:USE {CURrent|HALFgrat|FIVEdivs}

Related Commands CURSor:VBArs:UNIts

Arguments CURrent sets the V Bar measurement scale so that 0% is the current position of the left-most V Bar cursor and 100% is the current position of the right-most V Bar cursor.

HALFgrat resets the V Bar measurement scale to half the number of divisions (five for some models and four for others) so that 25% is the current position of the lowest V Bar cursor and 75% is the current position of the highest V Bar.

FIVEdivs sets V Bar measurement scale so that five screen major divisions is 100%, where 0% is –2.5 divisions and 100% is +2.5 divisions from the center vertical graticule.

Examples CURSOR:VBARS:USE CURRENT sets the V Bar measurement scale to use the current cursor positions as 0% and 100% of scale if units are set to %.

CURSor:VBArs:VDELTa? (Query Only)

Returns the vertical difference between the two vertical bar cursor ticks.

Group Cursor

Syntax `CURSor:VBArs:VDELTa?`

Related Commands CURSor:HBArs:UNIts

Returns `<NR3>` indicates the horizontal difference between the two vertical bar cursors.

Examples `CURSOR:VBARS:VDELTA?` might return CURSOR:VBARS:VDELTA 1.064E+0, indicating that the vertical difference between the vertical bar cursors ticks is 1.064 units.

CURSor:XY:POLar:RADIUS:DELta? (Query Only)

Returns the difference between the cursors X radius and the cursor Y radius (ΔY¸ ΔX). The ratio is calculated as (cursor 2 Y - cursor 1 Y) ÷ (cursor 2 X - cursor 1 X).

Group Cursor

Syntax `CURSor:XY:POLar:RADIUS:DELta?`

CURSor:XY:POLar:RADIUS:POSITION<x>? (Query Only)

Returns the polar radius for the specified cursor, where x can be either 1 or 2.

Group Cursor

Syntax `CURSor:XY:POLar:RADIUS:POSITION<x>?`

CURSor:XY:POLar:RADIUS:UNIts? (Query Only)

Returns the polar radius units.

Group Cursor

Syntax `CURSor:XY:POLar:RADIUS:UNIts?`

CURSor:XY:POLar:THETA:DELta? (Query Only)

Returns the XY cursor polar angle delta.

Group Cursor

Syntax `CURSor:XY:POLar:THETA:DELta?`

CURSor:XY:POLar:THETA:POSITION<x>? (Query Only)

Returns the cursor X or cursor Y polar coordinate, where x is either 1 or 2.

Group Cursor

Syntax `CURSor:XY:POLar:THETA:POSITION<x>?`

CURSor:XY:POLar:THETA:UNIts? (Query Only)

Returns the cursor coordinate units.

Group Cursor

Syntax `CURSor:XY:POLar:THETA:UNIts?`

CURSor:XY:PRODUCT:DELta? (Query Only)

Returns the difference between the cursors X position and cursor Y position. The $\Delta X \times \Delta Y$ value is calculated as $(X2 - X1) \times (Y2 - Y1)$.

Group Cursor

Syntax `CURSor:XY:PRODUCT:DELta?`

CURSor:XY:PRODUCT:POSITION<x>? (Query Only)

Returns the position of the X or Y cursor used to calculate the X × Y cursor measurement, Position 1 = (X1 × Y1); Position 2 = (X2 × Y2). The cursor is specified by x, which can be 1 or 2.

Group Cursor

Syntax `CURSor:XY:PRODUCT:POSITION<x>?`

CURSor:XY:PRODUCT:UNIts? (Query Only)

Returns the XY cursor product units.

Group Cursor

Syntax `CURSor:XY:PRODUCT:UNIts?`

CURSor:XY:RATIO:DELta? (Query Only)

Returns the ratio of the difference between the cursors X position and cursor Y position (ΔY , ΔX). The ratio is calculated as (Y2 – Y1) / (X2 – X1).

Group Cursor

Syntax `CURSor:XY:RATIO:DELta?`

CURSor:XY:RATIO:POSITION<x>? (Query Only)

Returns the X (horizontal) or Y (vertical) position for the specified cursor, which can be 1 (X) or 2 (Y). The ratio is calculated as Position 1 = (Y1/X1); Position 2 = (Y2/X2).

Group Cursor

Syntax `CURSor:XY:RATIO:POSITION<x>?`

CURSor:XY:RATIO:UNIts? (Query Only)

Returns the cursor X and cursor Y units for the ratio measurement.

Group Cursor

Syntax

```
CURSor:XY:RATIO:UNIts?
```

CURSor:XY:RECTangular:X:DELta? (Query Only)

Returns the cursor X delta value in rectangular coordinates.

Group Cursor

Syntax

```
CURSor:XY:RECTangular:X:DELta?
```

CURSor:XY:RECTangular:X:POSITION<x>

Sets or returns the X rectangular coordinate for cursor 1 or cursor 2. Cursors are specified by x and can be either 1 or 2.

Group Cursor

Syntax

```
CURSor:XY:RECTangular:X:POSITION<x> <NR3>
CURSor:XY:RECTangular:X:POSITION<x>?
```

Arguments <NR3> is the coordinate in volts.

CURSor:XY:RECTangular:X:UNIts? (Query Only)

Returns the cursor X rectangular units.

Group Cursor

Syntax

```
CURSor:XY:RECTangular:X:UNIts?
```

CURSor:XY:RECTangular:Y:DELta? (Query Only)

Returns The cursor Y delta value in rectangular coordinates.

Group Cursor

Syntax

```
CURSor:XY:RECTangular:Y:DELta?
```

CURSor:XY:RECTangular:Y:POSITION<x>

Sets or returns the Y rectangular coordinate for cursor 1 or cursor 2. The cursor is specified by x.

Group Cursor

Syntax

```
CURSor:XY:RECTangular:Y:POSITION<x> <NR3>
CURSor:XY:RECTangular:Y:POSITION<x>?
```

Arguments <NR3> is the coordinate in volts.

CURSor:XY:RECTangular:Y:UNIts? (Query Only)

Returns the cursor Y rectangular units.

Group Cursor

Syntax

```
CURSor:XY:RECTangular:Y:UNIts?
```

CURVe

Transfers waveform data to and from the oscilloscope in binary or ASCII format. Each waveform transferred includes a waveform preamble which contains the data format, scale, and associated information.

For analog waveforms, the CURVe? query transfers data from the oscilloscope. The data source is specified by the DATa:SOUrce command. The first and last data points are specified by the DATa:STARt and DATa:STOP commands.

The oscilloscope returns data from the last acquisition if the source is a channel waveform that is being previewed. The data does not reflect the acquisition

preview parameters. The user should always follow acquisition parameter changes with a single sequence OPC command prior to CURVe? to ensure the return data reflects the new acquisition parameters.

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

For CURVe queries, first set DATa:RESOlution and DATa:COMPosition appropriately. For MSO/DPO2000/B math FFT CURVe queries, set them to: `DATA:RESOLUTION FULL` and `DATA:COMPOSITION SINGULAR_YT`.

For more information, see: Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments.

(See page B-1.)

NOTE. *Transferring large volumes of data to or from the oscilloscope takes time. ASCII waveform transfer is very inefficient.*

NOTE. *For MSO models, reduced resolution waveform data is not available for the digital channels.*

S

Group Waveform Transfer

Syntax

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

Related Commands DATa:COMPositionDATa:DESTination, DATa:RESOlutionDATa:SOUrce, DATa:STARt, DATa:STOP, WFMInpre?, WFMInpre:BYT_Nr, WFMOutpre?, HEADer

Arguments `<Block>` is the waveform data in binary format. The waveform is formatted as: #<x><yyy><data><newline>, where:

<x> is the number of y bytes. For example, if <yyy>=500, then <x>=3)

<yyy> is the number of bytes to transfer if samples are one or two bytes wide. Use the WFMInpre:BYT_Nr command to set the width for waveforms transferred into the oscilloscope. Use WFMOutpre:BYT_Nr to set the width for waveforms transferred out from the oscilloscope.

<data> is the curve data.

<newline> is a single byte new line character at the end of the data.

`<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?` with ASCII encoding, start and stop of 1 and 10 respectively, and a width set to 1 might return `:CURVE 61,62,61,60,60,-59,-59,-58,-58,-59`

NOTE. *Curve data is transferred from the oscilloscope asynchronously, depending on the length of the curve record. Such transfers may require several seconds to complete. During this period, the oscilloscope will not respond to the user controls. You can interrupt these asynchronous data transfers by sending a device clear message to the oscilloscope or by interrupting the query with another command or query. In order to verify that curve data has been completely transferred, it is recommended that you follow such queries with an *ESR? query to verify there are no error bits set. You can also check the event queue to determine the cause of the error. If the error was caused by an interrupted query, then the asynchronous data transfer was not complete when the *ESR? query was sent. In such cases, it may be necessary to increase the program's time-out value to ensure that all data is transferred and read.*

DATa

Sets or returns the format and location of waveform data transferred with the CURVe? query or CURVe command.

Group

Waveform Transfer

Syntax

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

Related Commands

CURVe, DATa:STARt, DATa:STOP, DATa:ENCdg, WFMInpre:NR_Pt, WFMOutpre:NR_Pt?

Arguments

`INIT` initializes the waveform data parameters to their factory defaults except for DATa:STOP, which is set to the current acquisition record length.

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

Examples

`DATA?` might return `:DATA:DESTINATION REF1:ENCDG` RIBINARY;SOURCE CH1;START 1;STOP 500;WIDTH 1

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

DATa:COMPosition

Sets or returns the type of data that the CURVe? query returns.

NOTE. *For more information on querying waveform data, see Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments.)*

Group

Waveform Transfer

Syntax

```
DATa:COMPosition {COMPOSITE_YT|COMPOSITE_ENV|SINGULAR_YT}
DATa:COMPosition?
```

Arguments

`COMPOSITE_YT` is of native width 8-bits and is written as 8-bit data in ISF files and REF waveforms. The data can be queried as 1- or 2-byte data in CURVe? queries by setting the data width to 1 or 2.

`COMPOSITE_ENV` is min/max pair data. Its native width is 8-bits and is written as 8-bit data in ISF files and REF waveforms. The data can be queried as 1- or 2-byte data in CURVe? queries by setting the data width to 1 or 2.

`SINGULAR_YT` is 16-bit data which originates as averaged acquired data or math data. It is written to ISF files and REF waveforms as 16-bit data. The data can be queried as 1- or 2- byte data in CURVe? queries by setting the data width to 1 or 2.

Examples

`DATA:COMPOSITION?` might return `:DATA:COMPOSITION SINGULAR_YT` indicating that the data type returned by the CURVe? query is singular waveform.

`DATA:COMPOSITIONCOMPOSITE_YT` sets the return data type of CURVe? query as composite waveform.

DATa:COMPosition:AVAILable? (Query Only)

Lists the waveform data types that are available for return from the instrument under the current instrument settings.

Group

Waveform Transfer

Syntax

```
DATa:COMPosition:AVAILable?
```

Returns

`COMPOSITE_YT` for sample data.

`COMPOSITE_ENV` for envelope or peak-detect data.

`SINGULAR_YT` for sample data.

Examples

`DATA:COMPOSITION:AVAILABLE?` might return `DATA:COMPOSITION:AVAILABLE COMPOSITE_YT` indicating that the waveform data type.

DATa:DESTination

Sets or returns the reference memory location for storing waveform data transferred into the oscilloscope by the CURVe command.

Group

Waveform Transfer

Syntax

```
DATa:DESTination REF<x>
DATa:DESTination?
```

Related Commands

CURVe

Arguments

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

Examples

`DATA:DESTINATION?` might return `:DATA:DESTINATION REF2` indicating that reference 2is the currently selected reference memory location for incoming waveform data. `DATA:DESTINATION REF1` indicates that incoming waveform data be stored in reference 1.

DATa:ENCdg

Sets or returns the format of outgoing waveform data. This command is equivalent to setting WFMOutpre:ENCdg, WFMOutpre:BN_Fmt, andWFMInpre:FILTERFreq. Setting the DATa:ENGdg value causes the corresponding WFMOutpre values to be updated and conversely.

NOTE. *This command and query does not apply to incoming waveform data.*

Group Waveform Transfer

Syntax

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

Related Commands WFMOutpre:ENCdg, WFMOutpre:BN_Fmt, WFMInpre:FILTERFreq

Arguments

`ASCIi` specifies the ASCII representation for waveform data points. If ASCII is the value, then :BN_Fmt and :BYT_Or are ignored.

`FAStest` specifies that the data be sent in the fastest possible manner consistent with maintaining accuracy and is interpreted with respect to the waveform specified by DATa:SOUrce.

`RIBinary` specifies signed integer data point representation with the most significant byte transferred first.

When :BYT_Nr is 1, the range is from –128 through 127. When :BYT_Nr is 2, the range is from –32,768 through 32,767. Center screen is 0 (zero). The upper limit is the top of the screen and the lower limit is the bottom of the screen. This is the default argument.

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

When :BYT_Nr is 1, the range from 0 through 255. When :BYT_Nr is 2, the range is from 0 to 65,535. The center of the screen is 127. The upper limit is the top of the screen and the lower limit is the bottom of the screen.

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

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

Table 2-36: DATa and WFMOutpre Parameter Settings

DATa:ENCdg	WFMOutpre Settings			
Setting	:ENCdg	:BN_Fmt	:BYT_Or	:BYT_NR
ASCii	ASC	N/A	N/A	1,2
FAStest	BIN	RI	MSB	1,2
RIBinary	BIN	RI	MSB	1,2
RPBinary	BIN	RP	MSB	1,2
SRIbinary	BIN	RI	LSB	1,2
SRPbinary	BIN	RP	LSB	1,2

Examples

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

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

DATa:RESOlution

Sets or returns whether the CURVe? query returns full resolution records (acquired data) or reduced resolution records (filtered/displayed data).

NOTE. *For more information on querying waveform data, see Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments.)*

Group

Waveform Transfer

Syntax

```
DATa:RESOlution {FULL|REDUced}
DATa:RESOlution?
```

Arguments

`FULL` sets the instrument to return the full undecimated record acquired by the instrument. The full resolution records are not subject to the effects of FilterVu. Full resolution record lengths are 100,000, 125,000, 1,000,000 or 1,250,000 points.

`REDUced` sets the instrument to return short waveforms that are decimated from the full acquisition record. Reduced records are subject to the low pass filtering effects of FilterVu settings. Reduced record lengths range from 800 to 6250 points, depending on the acquisition settings.

Examples

`DATA:RESOLUTION?` might return `:DATA:RESOLUTION: FULL` indicating that the CURVe? query returns full resolution records (acquired data).

`DATA:RESOLUTIONREDUCED` sets the CURVe? returns to reduced resolution records.

DATa:SOUrce

Sets or returns the location of the waveform data transferred from the oscilloscope by the CURVe? query.

NOTE. *For more information on querying waveform data, see Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments.)*

Group Waveform Transfer

Syntax

```
DATa:SOUrce {CH1|CH2|CH3|CH4|MATH|REF1|REF2|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
DATa:SOUrce?
```

Related Commands CURVe

Arguments

`CH1–CH4` specifies which analog channel data will be transferred from the oscilloscope to the controller, channels 1 through 4.

`MATH` specifies that the Math waveform data will be transferred from the oscilloscope to the controller.

`REF1–REF2` specifies which Reference waveform data will be transferred from the oscilloscope to the controller, waveforms, 1 through 4.

`D0–D15` specifies that the returned waveform data is packaged as 4 bytes per sample, representing the states of all digital channels (D0–D15) and the logic states of the analog channels (Ch1–Ch4). When the `:DATA:SOURCE` is set to DIGITAL, the `:DATA:WIDTH` is automatically set to 4.

The format of the returned 4- byte (32 bit) data is as follows:

```
BIT       - Description
---         ---------------------------
31 (MSB)  - Unused
30        - Unused
29        - Unused
28        - Unused
27        - Unused
26        - Unused
25        - Unused
24        - Unused
23        - Unused
22        - Unused
21        - Unused
20        - Trigger (only when MagniVu is ON)
19        - Ch4 logic state
18        - Ch3 logic state
17        - Ch2 logic state
16        - Ch1 logic state
15        - D15 state
14        - D14 state
13        - D13 state
12        - D12 state
11        - D11 state
10        - D10 state
09        - D09 state
08        - D08 state
07        - D07 state
06        - D06 state
05        - D05 state
04        - D04 state
03        - D03 state
02        - D02 state
01        - D01 state
00 (LSB)  - D00 state
```

Examples

`DATA:SOURCE?` might return `:DATA:SOURCE REF2` indicating that the source for the waveform data which is transferred using a CURVe? query is reference 2.

`DATA:SOURCE CH1` specifies that the CH1 waveform will be transferred in the next CURVe? query.

DATa:STARt

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

Group Waveform Transfer

Syntax

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

Related Commands CURVe, DATa, DATa:STOP, WFMInpre:NR_Pt, WFMOutpre:NR_Pt?

Arguments `<NR1>` is the first data point that will be transferred, which ranges from 1 to the record length. Data will be transferred from `<NR1>` to DATa:STOP or the record length, whichever is less. If `<NR1>` is greater than the record length, the last data point in the record is transferred.

DATa:STARt and DATa:STOP are order independent. When DATa:STOP is greater than DATa:STARt, the values will be swapped internally for the CURVE? query.

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

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

DATa:STOP

Sets or returns the last data point that will be transferred when using the CURVe? query. This command allows for the transfer of partial waveforms from the oscilloscope.

Changes to the record length value are not automatically reflected in the `DATa:STOP` value. As record length is varied, the `DATa:STOP` value must be explicitly changed to ensure the entire record is transmitted. In other words, curve results will not automatically and correctly reflect increases in record length if the distance from DATa:STARt to `DATa:STOP` stays smaller than the increased record length.

Group Waveform Transfer

Syntax

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

Related Commands CURVe, DATa, DATa:STARt, WFMInpre:NR_Pt, WFMOutpre:NR_Pt?

Arguments

`<NR1>` is the last data point that will be transferred, which ranges from 1 to the record length. If `<NR1>` is greater than the record length, then data will be transferred up to the record length. If both DATa:STARt and DATa:STOP are greater than the record length, the last data point in the record is returned.

DATa:STARt and DATa:STOP are order independent. When DATa:STOP is less than DATa:STARt, the values will be swapped internally for the CURVE? query.

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

Examples

`DATA:STOP?` might return `:DATA:STOP 14900` indicating that 14900 is the last waveform data point that will be transferred.

`DATA:STOP 15000` specifies that the waveform transfer will stop at data point 15000.

DATa:WIDth

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

Group

Waveform Transfer

Syntax

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

Related Commands

CURVe

Arguments

`<NR1>` = 1 specifies that there is 1 byte (8 bits) per point.

`<NR1>` = 2 specifies that there are 2 bytes (16 bits) per point. This format is useful for AVErage waveforms.

Examples

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

DATE

Sets or returns the date the oscilloscope displays.

Group

Miscellaneous

Syntax

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

Related Commands

TIME

Arguments

`<QString>` is a date in the form "yyyy-mm-dd" where yyyy refers to a four-digit year number, mm refers to a two-digit month number from 01 to 12, and dd refers to a two-digit day number in the month.

Examples

`DATE "2006-01-24"` specifies that the date is set to January 24, 2006.

`DATE?` might return `:DATE 2006-01-24` indicating that the current date is set to January 24, 2006.

*DDT

Allows you to specify a command or a list of commands that execute when the oscilloscope receives a *TRG command or the GET IEEE488.2 interface message. Define Device Trigger (*DDT) is a special alias that the *TRG command uses.

Group

Miscellaneous

Syntax

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

Related Commands

ALIas, *TRG

Arguments

`<Block>` is a complete sequence of program messages. The messages can contain only valid commands that must be separated by semicolons and must follow all rules for concatenating commands. The sequence must be less than or equal to 80 characters. The format of this argument is always returned as a query.

`<QString>` is a complete sequence of program messages. The messages can contain only valid commands that must be separated by semicolons and must follow all rules for concatenating commands. The sequence must be less than or equal to 80 characters.

Examples

`*DDT #0ACQUIRE:STATE RUN` specifies that the acquisition system will be started each time a *TRG command is sent.

DESE

Sets or returns 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 detailed discussion of the use of these registers, see Registers.

Group Status and Error

Syntax

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

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

Arguments <NR1> sets the binary bits of the DESER according to this value, which ranges from 1 through 255. For example, DESE 209 sets the DESER to the binary value 11010001 (that is, the most significant bit in the register is set to 1, the next most significant bit to 1, the next bit to 0, etc.).

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

NOTE. *Setting the DESER and ESER to the same value allows only those codes to be entered into the Event Queue and summarized on the ESB bit (bit 5) of the Status Byte Register. Use the *ESE command to set the ESER.*

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

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

DIAg:LOOP:OPTion

Sets the self-test loop option.

Group Calibration and Diagnostic

Syntax

```
DIAg:LOOP:OPTion {ALWAYS|FAIL|ONFAIL|ONCE|NTIMES}
```

Arguments

`ALWAYS` continues looping until the self tests (diagnostics) are stopped via the front panel or by an oscilloscope command.

`FAIL` causes looping until the first self test (diagnostic) failure or until self tests (diagnostics) are stopped.

`ONFAIL` causes looping on a specific test group as long as a FAIL status is returned from the test.

`ONCE` executes self test (diagnostics test) sequence once.

`NTIMES` runs “n” number of loops.

Examples

`DIAG:LOOP:OPTION ONCE` runs one loop of self tests.

DIAg:LOOP:OPTion:NTIMes

Sets the self-test loop option to run N times.

Group

Calibration and Diagnostic

Syntax

```
DIAg:LOOP:OPTion:NTIMes <NR1>
DIAg:LOOP:OPTion:NTIMes?
```

Arguments

`<NR1>` is the number of self-test loops.

Examples

`DIAG:LOOP:OPTION:NTIMES 3` sets the self-test loop to run three times.

`DIAG:LOOP:OPTION:NTIMES?` might return `:DIAG:LOOP:OPTION:NTIMES 5`, indicating that the self-test loop is set to run five times.

DIAg:LOOP:STOP (No Query Form)

Stops the self-test at the end of the current loop.

Group

Calibration and Diagnostic

Syntax

```
DIAg:LOOP:STOP
```

Examples

`DIAG:LOOP:STOP` stops the self test at the end of the current loop.

DIAg:RESUlt:FLAg? (Query Only)

Returns the pass/fail status from the last self-test sequence execution. Use this query to determine which test(s) has failed.

Group Calibration and Diagnostic

Syntax `DIAg:RESUlt:FLAg?`

Related Commands DIAg:RESUlt:LOG?

Returns `PASS` indicates that all of the selected self (diagnostic) tests have passed.

`FAIL` indicates that at least one of the selected self (diagnostic) tests has failed.

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

DIAg:RESUlt:LOG? (Query Only)

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

Group Calibration and Diagnostic

Syntax `DIAg:RESUlt:LOG?`

Related Commands DIAg:RESUlt:FLAg?

Returns `<QString>` in the following format:

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

Examples `DIAG:RESULT:LOG?` might return :`DIAG:RESULT:LOG "NOT RUN--CPU,NOT RUN--DISPLAY,NOT RUN--FPANEL,NOT RUN--IO,NOT RUN--ACQ,NOT RUN--RO M,NOT RUN--APPKEY"`

DIAg:SELect (No Query Form)

Sets the type of diagnostics grouping.

Group Calibration and Diagnostic

Syntax `DIAg:SELect {ALL|APPKey|CPU|DISplay|FPAnel|IO|ROM|ACQ}`

Arguments `ALL` runs all diagnostic groups.

`APPKey` runs just the application key diagnostic group.

`CPU` runs just the CPU diagnostic group.

`DISplay` runs just the display circuit diagnostic group.

`FPAnel` runs just the front panel diagnostic group.

`IO` runs just the IO board diagnostic group.

`ROM` runs just the IO board diagnostic group.

`ACQ` runs just the acquisition system diagnostic group.

DIAg:SELect:<function> (No Query Form)

Runs self-tests on the specified system subsystem.

Group Calibration and Diagnostic

Syntax `DIAg:SELect:<function>`

Arguments `<function>` specifies a single oscilloscope subsystem on which to run self tests (diagnostics). Valid values are:

`ACQ` tests the acquisition system.

`APPKey` tests the application keys.

`CPU` tests the CPU.

`DISplay` tests the display.

`FPAnel` tests the front panel controls.

`IO` tests the IO ports.

`ROM` tests the system read only memory.

Examples `DIAG:SELECT:CPU` sets the oscilloscope to run just CPU tests.

DIAg:STATE (No Query Form)

This command starts or stops the oscilloscope self-test. Depending on the argument, self-test capabilities are either turned on or off.

Group Calibration and Diagnostic

Syntax

```
DIAg:STATE {EXECute|ABORt}
```

Arguments `EXECute` starts diagnostics.

`ABORt` stops diagnostics at the end of the current loop.

Examples `DIAG:STATE EXECute` starts diagnostics.

DISplay? (Query Only)

Returns the current display settings.

Group Display

Syntax

```
DISplay?
```

DISplay:CLOCk

Sets or returns whether the oscilloscope displays the date and time. The query form of this command returns an ON (1) or an OFF (0).

NOTE. *For the MSO/DPO2000/B series, the enumerations enable control of the date/time display. The query form will return 0 when the control is set to NONe or OFF and will return 1 when the control is set to DATETIME or ON.*

Group Display

Syntax

```
DISplay:CLOCk {DATEOnly|TIMEOnly|DATETIME|NONe|ON|OFF|<NR1>}
DISplay:CLOCk?
```

Related Commands DATE, TIME

Arguments `DATEOnly` enables the display of date.

`TIMEOnly` enables the display of time.

`DATETIME or ON` enables the display of both date and time.

`NONe or OFF` disables the display of date and time.

`<NR1>` = 0 disables the display of date and time; any other value enables the display of date and time.

Examples `DISPLAY:CLOCK ON` enables display of date and time.

`DISPLAY:CLOCK?` might return `:DISPLAY:CLOCK 1` indicating that the display of date and time is currently enabled.

DISplay:DIGital:HEIght

Sets or returns the number of available digital waveform position slots.

NOTE. *If the number of live digital channels exceeds the number of slots, their height is reduced to a smaller size, allowing all digital channels to be displayed.*

Group Display

Syntax

```
DISplay:DIGital:HEIght {SMAll|MEDium|LARge}
DISplay:DIGital:HEIght?
```

Arguments `SMAll` sets the height to 40.

`MEDium` sets the height to 20.

`LARge` sets the height to 10.

DISplay:FORMat

Sets or returns the display format.

Group Display

Syntax

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

Arguments

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

`XY` argument displays one waveform against another. Selecting one source causes its corresponding source to be implicitly selected, producing a single trace from the two input waveforms.

Examples

`DISPLAY:FORMAT XY` sets the display format to `XY`.

`DISPLAY:FORMAT?` might return `DISPLAY:FORMAT YT` indicating that the display format is `YT`.

DISplay:GLITch

Controls the display of the glitch capture background of the waveform. Set the intensity with DISplay:INTENSITy:GLITch.

Group

FilterVu

Syntax

```
DISplay:GLITch {ON|OFF|<NR1>}
DISplay:GLITch?
```

Related Commands

DISplay:INTENSITy:GLITch

Arguments

`ON` or `<NR1>` ≠ 0 turns on the glitch capture waveform display.

`OFF` or `<NR1>` = 0 turns off the glitch capture waveform display.

Examples

`DISPLAY:GLITCH?` might return DISPLAY:GLITCH 1 indicating that there is glitch capture waveform display.

`DISPLAY:GLITCH OFF` turns off the glitch capture waveform display.

DISplay:GRAticule

Selects or queries the type of graticule the oscilloscope displays.

Group

Display

Syntax

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

Arguments

`CROSSHair` specifies a frame and cross hairs.

`FRAme` specifies a frame only.

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

`GRId` specifies a frame and grid only.

Examples

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

`DISPLAY:GRATICULE?` might return `:DISPLAY:GRATICULE FULL` indicating that all graticule elements are selected.

DISplay:INTENSITy? (Query Only)

Returns the display intensity settings.

Group

Display

Syntax

```
DISplay:INTENSITy?
```

Examples

`DISPLAY:INTENSITY?` might return `:DISPLAY:INTEnsITY:WAVEFORM 35;GRATICULE 35;BACKLIGHT HIGH;GLITCH CAPTURE BACKGROUND INTENSITY 55`

DISplay:INTENSITy:BACKLight

Sets and returns the waveform backlight intensity settings.

Group

Display

Syntax

```
DISplay:INTENSITy:BACKLight {LOW|MEDium|HIGH}
DISplay:INTENSITy:BACKLight?
```

Examples

`DISPLAY:INTENSITY:BACKLIGHT?` might return `:DISPLAY:INTEnsITY:BACKLIGHT HIGH`

DISplay:INTENSITy:GLITch

Sets the intensity of the glitch capture background of the waveform display. The intensity can be set from 5 to 100% in increments of 5%. The intensity of the glitch capture background can only be adjusted when the glitch capture background is displayed and FilterVu filtering is in use. Otherwise, the intensity is fixed.

Use DISplay:GLITch to turn the glitch capture background ON and OFF.

Use FILTERVu:FREQuency to enable filtering. Filtering is enabled if the frequency chosen is different than the oscilloscope's full bandwidth.

Group FilterVu

Syntax

```
DISplay:INTENSITy:GLITch {<NR1>}
```

Related Commands

Arguments `<NR1>` is the glitch intensity and ranges from 5 to 100 percent.

Examples `DISPLAY:INTENSITY:GLITCH ?` might return `DISPLAY:INTENSITY:GLITCH 5` indicating that the glitch intensity is 5%.

DISplay:INTENSITy:GRAticule

Sets and returns the display graticule intensity settings.

Group Display

Syntax

```
DISplay:INTENSITy:GRAticule <NR1>
DISplay:INTENSITy:GRAticule?
```

Arguments `<NR1>` is the graticule intensity and ranges from 0 to 100 percent.

Examples `DISPLAY:INTENSITY:GRATICULE?` might return `DISPLAY:INTEnsITY:GRATICULE 30`

DISplay:INTENSITy:WAVEform

Sets and returns the display waveform intensity settings.

Group Display

Syntax

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

Arguments `<NR1>` is the waveform intensity and ranges from 1 to 100 percent.

Examples `DISPLAY:INTENSITY:WAVEFORM?` might return

```
DISPLAY:INTEnsITY:WAVEFORM 60
```

as the intensity of the waveforms.

DISplay:PERSistence

Sets or returns the display persistence. This affects the display only.

NOTE. *When Persistence is set to Infinite, it does not mean that the brightness of any pixel should never decrease. The brightness of a pixel is proportionally dependent on the ratio between its intensity (which does NOT decrease at Infinite Persistence) and the maximum value of intensity of any pixel on the screen. If a particular pixel get hit less often than others, its brightness will decrease over time. It will become less bright relative to the pixels that get hit often.*

Group Display

Syntax

```
DISplay:PERSistence {<NR3>|CLEAR|AUTO|MINImum|INFInite}
DISplay:PERSistence?
```

Arguments `<NR3>` specifies the time of the persistence.

`CLEAR` resets the persist time count down and clears the display of acquired points.

`INFInite` displays waveform points until a control change resets the acquisition system. When persistence is set to infinite, it does not mean that the brightness of any pixel should never decrease. The brightness of a pixel is proportionally dependent on the ratio between its intensity, which does NOT decrease at infinite persistence, and the maximum value of intensity of any pixel on the screen. Thus, if a particular pixel gets hit less often than others, its brightness will decrease over time. It will become less bright relative to the pixels that get hit often.

`AUTO` specifies that the oscilloscope automatically determines the best waveform persistence based on the value of waveform intensity (DISPLAY:INTEnsITY:WAVEFORM)

`MINImum` specifies that the waveform persistence is set to the minimum value of 0.0E0.

Examples

`DISPLAY:PERSISTENCE 3` specifies that the waveform points are displayed fading for 3 seconds before they completely disappear.

D<x>

Sets or returns parameters for digital channel <x>, where x is the channel number.

Group

MSO

Syntax

```
D<x>
```

D<x>:LABel

Sets or returns the waveform label for digital channel <x>, where x is the channel number.

Group

MSO

Syntax

```
D<x>:LABel <Qstring>
D<x>:LABel?
```

Arguments

`<Qstring>` is an alpha-numeric string of text, enclosed in quotes, that contains the text label information for the digital channel <x> waveform. The text string is limited to 30 characters.

Examples

`D4:LABEL?` might return `:D4:LABEL "Clk wfm 4"`.

D<x>:POSition

Sets or returns the vertical position for digital channel <x>, where x is the channel number.

NOTE. *Unlike analog channels, which use the center baseline as the zero point, digital channels use the bottom logic level as the zero point. Because of the slot model used to describe digital channel position, the oscilloscope rounds off divisions to the nearest slot.*

Group MSO

Syntax

```
D<x>:POSition <NR3>
D<x>:POSition?
```

Arguments <NR3> is the position of the digital channel in slot units. Use the oscilloscope front-panel controls to place the channel; then query the channel to obtain an exact value for the position.

D<x>:THREshold

Sets or returns the logical threshold for digital channel <x>, where x is the channel number.

Group MSO

Syntax

```
D<x>:THREshold {ECL|TTL|<NR3>}
D<x>:THREshold?
```

Arguments ECL sets the digital threshold for channel <x> to a preset ECL high level of –1.3V.

TTL sets the digital threshold for channel <x> to a preset TTL high level of 1.4V.

<NR3> specifies the digital threshold for channel <x> in volts.

Examples D5:THRESHOLD ECL sets the threshold for D5 digital channel to ECL levels.

D14:threshold? might return :D14:THRESHOLD 1.2500 indicating that the threshold level for the D14 digital channel is 1.25 volts.

*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 detailed discussion on how to use registers, see Registers.

Group Status and Error

Syntax

```
*ESE <NR1>
*ESE?
```

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

Arguments <NR1> specifies the binary bits of the ESER according to this value, which ranges from 0 through 255.

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

NOTE. *Setting the DESER and the ESER to the same values 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.*

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

`*ESE?` might return 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 (reading the SESR clears it). For a detailed discussion on how to use registers, see Registers.

Group Status and Error

Syntax `*ESR?`

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

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

ETHERnet:DHCPbootp

Sets or returns the network initialization search for a DHCP/BOOTP server.

Group Ethernet

Syntax

```
ETHERnet:DHCPbootp {ON|OFF}
ETHERnet:DHCPbootp?
```

Arguments

`ON` enables the oscilloscope to search the network for a DHCP or BOOTP server in order to automatically assign a dynamic IP address to the oscilloscope.

NOTE. *Do not use DHCP/BOOTP searching if your oscilloscope has been assigned a static address on a network. If you set this command to ON, the DHCP/BOOTP search will delete or change your static IP address information.*

`OFF` disables the oscilloscope to search the network for a DHCP or BOOTP server.

Examples

`ETHERNET:DHCPBOOTP ON` sets the oscilloscope to search for a DHCP or BOOTP server and assign a dynamic IP address to the oscilloscope.

ETHERnet:DNS:IPADDress

Sets or returns the network Domain Name Server (Dns) IP address.

Group

Ethernet

Syntax

```
ETHERnet:DNS:IPADDress <QString>
ETHERnet:DNS:IPADDress?
```

Arguments

`<QString>` is a standard IP address value, enclosed in quotes.

Examples

`ETHERNET:DNS:IPADDRESS “128.196.13.352”` sets the Dns IP address that the oscilloscope uses to communicate with the network.

ETHERnet:DOMAINname

Sets or returns the network domain name.

Group

Ethernet

Syntax

```
ETHERnet:DOMAINname <Qstring>
ETHERnet:DOMAINname?
```

Arguments `<QString>` is the network domain name, enclosed in quotes.

Examples `ETHERNET:DOMAINNAME "Alpha1.Mycorp.com"` sets the domain name that the oscilloscope uses to communicate with the network.

ETHERnet:ENET:ADDress? (Query Only)

Returns the Ethernet address value assigned to the oscilloscope. This is assigned at the factory and can not be changed.

Group Ethernet

Syntax

```
ETHERnet:ENET:ADDress?
```

Examples `ETHERNET:ENET:ADDRESS?` returns an Ethernet address such as `08:00:11:01:02:03`

ETHERnet:GATEWay:IPADDress

Sets or returns the remote interface gateway IP address.

Group Ethernet

Syntax

```
ETHERnet:GATEWay:IPADDress <QString>
ETHERnet:GATEWay:IPADDress?
```

Arguments `<QString>` is a standard IP address value, enclosed in quotes.

Examples `ETHERNET:GATEWAY:IPADDRESS "128.143.16.1"` sets the gateway IP address.

ETHERnet:HTTPPort

Sets or returns the remote interface HTTP port value.

Group Ethernet

Syntax

```
ETHERnet:HTTPPort <QString>
ETHERnet:HTTPPort?
```

Arguments

`<QString>` is an integer port number, enclosed in quotes.

NOTE. *Consider the following if you are using the e*Scope™ control software. If you don't enter a port address in the URL, then the* `ETHERnet:HTTPPort` *value must be set to "80", which is the default port for HTTP protocol. If you use a URL with a port address (for example: http://DPO2004-04WKL4:1234), the port number is specified by the number after the colon. Set the* `ETHERnet:HTTPPort` *value to this same number.*

Examples

`ETHERNET:HTTPPORT` "80" sets the HTTP port value to 80.

ETHERnet:IPADDress

Sets or returns the IP address assigned to the oscilloscope.

Group

Ethernet

Syntax

```
ETHERnet:IPADDress <QString>
ETHERnet:IPADDress?
```

Arguments

<QString> is a standard IP address value, enclosed in quotes.

Examples

`ETHERNET:IPADDRESS` "123.121.13.214" sets the oscilloscope's IP address.

ETHERnet:NAME

Sets or returns the network name assigned to the oscilloscope.

Group

Ethernet

Syntax

```
ETHERnet:NAME <QString>
ETHERnet:NAME?
```

Arguments

`<QString>` is the network name assigned to the oscilloscope, enclosed in quotes.

Examples `ETHERNET:NAME` “labscope1” sets the oscilloscope's network name.

ETHERnet:PASSWord

Sets or returns the HTTP Ethernet access password. If a password is set, the user must enter the password before the user's Web browser can access the oscilloscope.

Group Ethernet

Syntax

```
ETHERnet:PASSWord <new>
ETHERnet:PASSWord?
```

Arguments `<new>` is a new password, enclosed in quotes.

Examples `ETHERNET:PASSWORD ”ZEN53”` replaces the current Ethernet password with the new password ZEN53.

`ETHERNET:PASSWORD?` might return :ETHERNET:PASSWORD “ZEN53”.

ETHERnet:PING (No Query Form)

Causes the oscilloscope to ping the gateway IP address.

Group Ethernet

Syntax

```
ETHERnet:PING EXECute
```

Examples `ETHERNET:PING EXECUTE` causes the oscilloscope to ping the gateway IP address.

ETHERnet:PING:STATUS? (Query Only)

Returns the results from sending the ETHERnet:PING command to ping the gateway IP address.

Group Ethernet

Syntax

```
ETHERnet:PING:STATUS?
```

Returns

`OK` is returned if the computer at the gateway IP address answers.

`NORESPOnsE` is returned if the computer at the gateway IP address does not answer.

`INPROGRESS` is returned if the ping operation is still executing.

ETHERnet:SUBNETMask

Sets or returns the remote interface subnet mask value.

Group

Ethernet

Syntax

```
ETHERnet:SUBNETMask <QString>
ETHERnet:SUBNETMask?
```

Arguments

`<QString>` is the subnet mask value, enclosed in quotes.

Examples

`ETHERNET:SUBNETMASK "255.255.255.0"` sets the subnet mask value using standard IP address notation format.

EVENT? (Query Only)

Returns an event code from the Event Queue that provides information about the results of the last *ESR? read. EVENT? also removes the returned value from the Event Queue.

Group

Status and Error

Syntax

`EVENT?`

Related Commands

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

Examples

`EVENT?` might return `:EVENT 110`, showing that there was an error in a command header. (See page 3-12, *Messages*.)

EVMsg? (Query Only)

Removes a single event code from the Event Queue that is associated with the results of the last *ESR? read and returns the event code along with an explanatory message. For information, see Event Handling.

Group Status and Error

Syntax `EVMsg?`

Related Commands ALLEv?

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

Returns The event code and message in the following format:

<Event Code><Comma><QString>[<Event Code><Comma><QString>...]<QString>::= <Message>;[<Command>] where <Command> is the command that caused the error and may be returned when a command error is detected by the oscilloscope. As much of the command will be returned as possible without exceeding the 60 character limit of the <Message> and <Command> string combined. The command string is right-justified.

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

EVQty? (Query Only)

Returns the number of event codes in the Event Queue. This is useful when using the ALLEv? query, which returns the exact number of events.

Group Status and Error

Syntax `EVQty?`

Related Commands ALLEv?, EVENT?, EVMsg?

Examples `EVQTY?` might return `:EVQTY 3`, indicating that the number of event codes in the Event Queue.

FACtory (No Query Form)

Resets the oscilloscope to its factory default settings.

This command does the following:

- Clears the Event Status Enable Register
- Clears the Service Request Enable Register
- Sets the Device Event Status Enable Register to 255
- Purges all defined aliases
- Enables all Command Headers
- Sets the macro defined by *DDT to a "zero-length field"
- Clears the pending operation flag and associated operations

This command does not reset the following:

- Communication settings
- Sate of the VXI-11 (Ethernet IEEE Std 488.2) interface.
- Calibration data that affects device specifications
- Protected user data
- Stored settings
- Power On Status Clear Flag
- Oscilloscope password

Group Save and Recall

Syntax `FACtory`

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

Arguments None

Examples `FACTORY` resets the oscilloscope to its factory default settings.

FILESystem

Returns the directory listing of the current working directory and the number of bytes of free space available. This query is the same as the FILESystem:DIR? query and the FILESystem:FREESpace? query.

Group File System

Syntax

```
FILESystem
FILESystem?
```

Related Commands FILESystem:CWD, FILESystem:DELEte, FILESystem:DIR?, FILESystem:REName

Arguments None

Examples `FILESYSTEM?` might return

```
:FILESYSTEM:DIR "TEK00000.BMP","GLITCH1.PNG","TEMP.TMP",
"FILE1.WFM","FILE2.WFM", "MATH1.WFM"," REF1.WFM","REF2.WFM"
```

FILESystem:CWD

Sets or returns the current working directory (CWD) for FILESystem commands. The default working directory is "E:/". Anytime you use this command to change the directory, the directory that you specify is retained as the current working directory until you either change the directory or you delete the directory. If you delete the current working directory, the oscilloscope resets current working directory to the default directory (E:) the next time the oscilloscope is powered on or the next time you execute a file system command.

This command supports the permutations of file and directory names supported by Microsoft Windows:

- Relative path names; for example, "./temp"
- Absolute path names; for example, "E:/Wfms"
- Implied relative path names; for example "NEWFILE.TXT" becomes "E:/TEKSCOPE/NEWFILE.TXT" if the current working directory is "E:/TEKSCOPE"

Group File System

Syntax

```
FILESystem:CWD {<new working directory path>}
```

Arguments

`<new working directory path>` is a quoted string that defines the current working; a directory name can have up to 8 characters with an extension of up to 3 characters.

Examples

`FILESYSTEM:CWD "E:/TEKSCOPE/IMAGES"` sets the current working directory to images.

`FILESYSTEM:CWD?` might return

`:FILESYSTEM:CWD "E:/TEKSCOPE/WAVEFORMS"` indicating that the current working directory is set to waveforms.

FILESystem:DELEte (No Query Form)

This command deletes a named file. If you specify a directory name, it will delete the directory and all of its contents, the same as the RMDir command. You can also specify the filename as *.* to delete all of the files in the current or specified directory.

Group

File System

Syntax

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

Related Commands

FILESystem:CWD

FILESystem:RMDir

Arguments

`<file path>` is a quoted string that defines the file name and path. If the file path is within the current working directory, you need only specify the file name.

The argument `*.*` will delete all files and subdirectories within the current working directory.

Examples

`FILESYSTEM:DELETE "NOT_MINE.SET"` deletes the file named NOT_MINE.SET from the current working directory.

FILESystem:DIR? (Query Only)

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

Group File System

Syntax `FILESystem:DIR?`

Related Commands FILESystem:CWD, FILESystem:MKDir

Arguments None

Returns `FILESystem:DIR?` returns a list of files and directories in the current working directory.

Examples `FILESYSTEM:DIR?` might return

```
:FILESYSTEM:DIR
"TEK00000.PNG","CANSETUP.SET","WFM1.ISF","MYIMAGES"
```

FILESystem:FORMat (No Query Form)

Formats a mass storage device. This command should be used with extreme caution as it causes all data on the specified mass storage device to be lost. Drive letters (e.g., E:) are case sensitive and must be upper case. For all other FILESYSTEM commands, drives letters are not case sensitive. Example: FILES:FORMAT "E:/" Formats the USB flash drive installed in the oscilloscope's front panel USB port.

Group File System

Syntax `FILESystem:FORMat`

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

Examples `FILESYSTEM:FORMAT "E:/"`

Formats the USB flash drive installed in the oscilloscope's front panel USB port.

FILESystem:FREESpace? (Query Only)

Returns the number of bytes of free space on the current drive.

Group File System

Syntax `FILESystem:FREESpace?`

Related Commands FILESystem:FREESpace?, FILESystem:CWD

FILESystem:MKDir (No Query Form)

Creates a new folder.

Group File System

Syntax `FILESystem:MKDir <directory path>`

Related Commands FILESystem:CWD, FILESystem:DIR?

Arguments `<directory path>` is a quoted string that specifies the directory to create

Examples `FILESYSTEM:MKDIR "E:/NewDirectory"` creates the directory named *NEWDIRECTORY* at the root of the E drive.

These two commands create the directory MYNEWSUBDIRECTORY within the existing directory mydirectory at the root of the E drive:

```
FILESYSTEM:CWD "E:/MyDirectory";:FILESYSTEM:MKDIR
"MyNewSubDirectory"
```

This, of course, assumes that *E:/MYDIRECTORY* already existed and was not a read-only directory.

FILESystem:READFile (No Query Form)

Writes the contents of the specified file to the specified interface. If the file does not exist or is not readable, an appropriate error event is posted.

Group File System

Syntax `FILESystem:READFile <QString>`

Related Commands FILESystem:CWD

Arguments `<QString>` is a quoted string that defines the file name and path. If the file path is within the current working directory, specify only the file name.

Examples `FILESYSTEM:READFILE "E:/TEST_DATA/TEK00016CH1.CSV"` reads the content of the specified file, if the file exists and is readable, and sends the content of the file to the current interface.

FILESystem:REName (No Query Form)

Assigns a new name to an existing file.

Group File System

Syntax

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

Related Commands FILESystem:CWD

Arguments `<old file path>` is a quoted string that defines the file name and path. If the file path is within the current working directory, you need only specify the file name.

`<new file path>` is a quoted string that defines the file name and path. If the file path is within the current working directory, you need only specify the file name.

Examples `FILESYSTEM:RENAME "E:/TEK00000.SET","D:/MYSETTING.SET"` gives the file named TEK00000.SET the new name of MYSETTING.SET. The file remains in the root directory on the D drive.

FILESystem:RMDir (No Query Form)

Deletes a named directory. This command deletes the specified directory and all of its contents. The directory must not be a read-only directory.

Group File System

Syntax

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

Related Commands FILESystem:CWD

Arguments `<directory path>` is a quoted string that defines the directory name and path. If the file path is within the current working directory, you need only specify the file name.

Examples `FILESYSTEM:RMDIR "E:/OldDirectory"` removes the directory named olddirectory from the root of the E drive.

FILESystem:WRITEFile (No Query Form)

Writes the specified block data to a file in the oscilloscope current working directory. If the specified file does not exist or is not readable, an appropriate error event is posted. The maximum length of the block data is 262144 bytes.

Group File System

Syntax

```
FILESystem:WRITEFile <file path>, <data>
```

Related Commands FILESystem:CWD

Arguments `<file path>` is the quoted string that defines the file name and path. If the path is within the current working directory, specify the file name only.

`<data>` can be either DEFINITE LENGTH encoding or INDEFINITE LENGTH ARBITRARY BLOCK PROGRAM DATA encoding as described in IEEE488.2.

FILTERVu:FREQuency

Sets or queries the FilterVu frequency to the closest value supported with the current acquisition settings.

Group FilterVu

Syntax

```
FILTERVu:FREQuency?
FILTERVu:FREQuency <NR3>
```

Related Commands FILTERVu:FREQuency:AVAILable?

Arguments <NR3> is the integer format for the current FilterVu frequency setting.

Examples `FILTERVU:FREQUENCY?` might return `FILTERVU:FREQUENCY 100000000` indicating that the FilterVu frequency is 100 MHz.

FILTERVu:FREQuency:AVAILable? (Query Only)

Returns a comma separated list of filter frequencies available based on the current instrument settings. The source waveform (as specified by the DATa:SOUrce) must be turned on for this query to generate a response.

Group FilterVu

Syntax `FILTERVu:FREQuency:AVAILable?`

Related Commands FILTERVu:FREQuency

Examples `FILTERVU:FREQUENCY:AVAILABLE?` might return `FILTERVU:FREQUENCY:AVAILABLE 200000000,150000000,75000000,42000000,21000000` indicating that the available FilterVu frequencies are 200 MHz, 150 MHz, 75 MHz, 42 MHz and 21 MHz.

FPAnel:PRESS (No Query Form)

Simulates the action of pressing a specified front-panel button.

When the front panel is locked, the front-panel button and multipurpose knob operations are suspended. The `FPAnel:PRESS` and the FPAnel:TURN commands will also not work. You can work around this by using the appropriate programmatic interface commands, instead of the front-panel commands.

Group Miscellaneous

Syntax `FPAnel:PRESS <button>`

Arguments <button> is the name of a front-panel button. Most of the argument names associate directly with their front panel buttons. For example, AUTOSet is for the Autoset button. The <button> enumeration arguments and their associations with the front panel buttons are listed below.

Table 2-37: FPAnel:PRESS arguments

Argument	Description
ACQuire	**Acquire** button
SAVERecall	Save/Recall **Menu** button
MEASurement	**Measure** button
UTILity	**Utility** button
SEArch	**Search** button
TESt	**Test** button
MATh	**M** button
REF	**R** button
TRIGger	Trigger **Menu** button
CH1	Channel1 select button
CH2	Channel2 select button
CH3	Channel2 select button
CH4	Channel4 select button
B1	Bus1 select button
B2	Bus2 select button
D15D0	Digital channels select buttons
DEFaultsetup	**Default Setup** button
MENUOff	**Menu Off** button
SELect	**Select** button
FINe	**Fine** button
CURsor	**Cursors** button
INTensity	**Intensity** button
MARk	Mark **Set/Clear** button
PREv	Previous arrow button
PAUse	Play/pause button
NEXt	Next arrow button
RUnstop	**Run/Stop** button
SINGleseq	**Single** button
AUTOset	**Autoset** button
SETTO50	Set Trigger to 50%
FORCetrig	**Force Trig** button
HARDcopy	Hardcopy button
SAVEBUtton	**Save** button
BMENU1	Screen left-most bottom menu button
BMENU2	Screen bottom menu button
BMENU3	Screen bottom menu button
BMENU4	Screen bottom menu button

Table 2-37: FPAnel:PRESS arguments (cont.)

Argument	Description
BMENU5	Screen bottom menu button
BMENU6	Screen bottom menu button
BMENU7	Screen right-most bottom menu button
RMENU1	Screen top-most side menu button
RMENU2	Screen side menu button
RMENU3	Screen side menu button
RMENU4	Screen side menu button
RMENU5	Screen bottom-most side menu button
MAGnify	Magnify (zoom) button (not the zoom/pan knob)
FITERVu (2000/B series only)	FilterVu frequency button
WFMOnly (2000/B series only)	Waveform data button

Examples

`FPANEL:PRESS AUTOSET` executes the oscilloscope Autoset function.

FPAnel:TURN (No Query Form)

Simulates the action of turning a specified front-panel control knob.

When the front panel is locked, the front-panel button and multipurpose knob operations are suspended. The FPAnel:PRESS and `FPAnel:TURN` commands will also not work, and, they will not generate an error. You can work around this by using the appropriate programmatic interface commands, instead of the front-panel commands. For example, to set the trigger level to 50%, you could use TRIGger:A SETLevel. To force a trigger, you could use TRIGger FORCe.

Group

Miscellaneous

Syntax

`FPAnel:TURN <knob>,[<n>]`

Arguments

`<knob>` is the name of a rotating control.

A comma (,) separates the control knob argument from the numeric optional rotation value argument. In the absence of the numeric rotation value argument, the default is 1 (clockwise). You do not need a white space between the arguments and the comma.

`<n>` represents the rotation direction and magnitude of rotation. Negative values represent a counterclockwise knob rotation, and positive values represent a

clockwise rotation. The magnitude of <n> specifies the amount of the turn, where <n> = 1 represents turning the knob one unit, <n> = 2 represents turning the knob two units, <n> = 4 represents turning the knob four units, and so on. The range of units depends on which front panel knob is specified.

Table 2-38: FPAnel:TURN arguments

Argument	Description
GPKNOB1	**Multipurpose a** knob
GPKNOB2	**Multipurpose b** knob
HORZPos	Horizontal **Position** knob
HORZScale	Horizontal **Scale** knob
PANKNOB1	Outer pan knob
TRIGLevel	Trigger **Level** knob
VERTPOS<n>	Vertical **Position** knob
VERTSCALE<n>	Vertical **Scale** knob
ZOOM	Inner zoom knob

Examples

`FPANEL:TURN TRIGLEVEL,10` duplicates turning the front-panel Trigger Level knob clockwise by 10 units.

GPIBUsb:ID? (Query Only)

Returns the identification string of the connected TEK-USB-488 adaptor module and firmware version. If a TEK-USB-488.2 module is not connected, the system returns “Not detected”.

Group

Miscellaneous

Syntax

```
GPIBUsb:ID?
```

HARDCopy (No Query Form)

Sends a hard copy of the screen display to the currently active printer using the current palette and layout settings.

Group

Hard Copy

Syntax

```
HARDCopy {START}
HARDCopy?
```

Related Commands *WAI, *CLS

Arguments START sends a block of data representing the current screen image to the requested port. The data sent is in the image format specified by SAVe:IMAGe:FILEFormat, and the compression level is controlled by whatever format has been selected (BMP and TIFF are uncompressed, while PNG is compressed).

Examples HARDCOPY initiates a screen copy to the active printer.

HARDCopy:INKSaver

Changes hard copy output to print traces and graticule on a white background while retaining waveform color information (except for channel 1, which prints as dark blue because yellow does not show up well and is difficult to see on a white background). This option can significantly reduce print time and quantities of ink required compared with WYSIWYG dark background images.

Group Hard Copy

Syntax

```
HARDCopy:INKSaver?
```

Arguments ON or <NR1> ≠ 0 sets the ink saver mode on.

OFF or <NR1> = 0 sets the ink saver mode off.

Examples HARDCOPY:INKSAVER ON will cause subsequent hard copy output to display the screen on a white background.

HARDCopy:PREVIEW (No Query Form)

Displays a preview of the current screen contents with the InkSaver palette applied.

Group Hard Copy

Syntax

```
HARDCopy:PREVIEW {ON|OFF|<NR1>}
```

Arguments

ON or <NR1> ≠ 0 turns preview mode on.

OFF or <NR1> = 0 turns preview mode off.

HEADer

Sets or returns the Response Header Enable State that causes the oscilloscope to either include or omit headers on query responses.

NOTE. *This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk); these commands never return headers. This command does affect the Response Header Enable State of both the USBTMC and VXI-11 interfaces. Refer to the Introduction for additional information.*

Group

Miscellaneous

Syntax

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

Related Commands

VERBose

Arguments

OFF sets the Response Header Enable State to false. This causes the oscilloscope to omit headers on query responses, so that only the argument is returned.

ON sets the Response Header Enable State to true. This causes the oscilloscope to include headers on applicable query responses. You can then use the query response as a command.

<NR1> = 0 sets the Response Header Enable State to false; any other value sets this state to true.

Examples

HEADER OFF specifies that the oscilloscope omits headers on query responses, so that only the argument is returned.

HEADER? might return :HEADER 1 indicating that the oscilloscope is including headers on applicable query responses.

HORizontal? (Query Only)

Returns all settings for the horizontal commands.

Group

Horizontal

Syntax

```
HORizontal?
```

Examples

`HORIZONTAL?` might return the following horizontal settings `:HORIZONTAL:DIGITAL:RECORDLENGTH:MAIN 100.0000E+3;:HORIZONTAL:DIGITAL:SAMPLERATE:MAIN 1.0000E+9;:HORIZONTAL:MAIN:UNITS "s";:HORIZONTAL:POSITION 50.0000;SAMPLERATE 1.0000E+9;SCALE 4.0000E-6;RECORDLENGTH 100000;DELAY:MODE 1;TIME 0.0E+0`

HORizontal:ACQLENGTH? (Query Only)

Returns the record length.

Group

Horizontal

Syntax

```
HORizontal:ACQLENGTH?
```

Related Commands

HORizontal:RECOrdlength

Examples

`HORIZONTAL:ACQLENGTH?` might return `HORizontal:ACQLENGTH?` `:HORIZONTAL:ACQLENGTH 1.0000E+6` indicating that the record length is 1 million points.

HORizontal:DELay:MODe

Sets or returns the horizontal delay mode.

Group

Horizontal

Syntax

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

Related Commands

HORizontal:POSition

Arguments

`OFF` sets the Horizontal Delay Mode to off. This causes the HORizontal:POSition command to horizontally position the waveform.

`ON` sets the Horizontal Delay Mode to on. This causes the HORizontal:DELay:TIMe command to horizontally position the waveform.

<NR1> = 0 sets the Horizontal Delay Mode to off; any other value sets this mode to on.

Examples

HORIZONTAL:DELAY:MODE OFF sets the Horizontal Delay Mode to off, allowing the HORizontal:POSition command to horizontally position the waveform.

HORIZONTAL:DELAY:MODE? might return HORIZONTAL:DELAY:MODE OFF indicating that the HORizontal:POSition command horizontally positions the waveform.

HORizontal:DELay:TIMe

Sets or returns the horizontal delay time. The amount of time the acquisition is delayed depends on sample rate and record length.

Group

Horizontal

Syntax

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

Arguments

NR3 is the delay in seconds.

Examples

HORizontal:DELay:TIME 0.3 sets the delay of acquisition data so that the resulting waveform is centered 300 ms after the trigger occurs.

HORizontal:DIGital:RECOrdlength:MAIN? (Query Only)

Returns the record length of the main digital acquisition.

Group

Horizontal

Syntax

```
HORizontal:DIGital:RECOrdlength:MAIN?
```

HORizontal:DIGital:SAMPLERate:MAIN? (Query Only)

Returns the sample rate of the main digital acquisition.

Group

Horizontal

Syntax `HORizontal:DIGital:SAMPLERate:MAIN?`

HORizontal:POSition

Sets or returns the horizontal position. If Horizontal Delay Mode is turned off, this command is equivalent to adjusting the HORIZONTAL POSITION knob on the front panel. When Horizontal Delay Mode is on, this command stores a new horizontal position that is used when Horizontal Delay Mode is turned off.

Group Horizontal

Syntax

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

Arguments `<NR3>` is the horizontal position expressed as the percentage of the waveform displayed left of the center of the graticule.

Examples `HORIZONTAL:POSITION 50` sets the horizontal position to 50%.

`HORIZONTAL:POSITION?` might return `:HORIZONTAL:POSITION 100` indicating that the horizontal position is set to 100%.

HORizontal:PREViewstate? (Query Only)

Returns a boolean value to indicate whether the acquisition system is in the preview state.

Group Horizontal

Syntax `HORizontal:PREViewstate?`

Returns `<NR1>` = 1 if the acquisition system is in the preview state.

`<NR1>` = 0 if the acquisition system is not in the preview state.

HORizontal:RECOrdlength

Sets the horizontal record length of acquired waveforms. The query form of this command returns the current horizontal record length.

Group Horizontal

Syntax

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

Arguments <NR1> represents the supported values for horizontal record lengths, which are: 100000 or 1000000.

Examples HORIZONTAL:RECORDLENGTH 100000 specifies that 100000 data points will be acquired for each record.

HORIZONTAL:RECORDLENGTH? might return :HORIZONTAL:RECOrdlength 100000 indicating that the horizontal record length is equal to 100000 data points.

HORizontal:RESOlution

Sets or returns the horizontal record length of acquired waveforms. The sample rate is automatically adjusted at the same time to maintain a constant time per division. The query form of this command returns the current horizontal record length.

Group Horizontal

Syntax

```
HORizontal:RESOlution <NR1>
HORizontal:RESOlution?
```

Arguments <NR1> represents the supported values for horizontal record lengths.

HORizontal:SAMPLERate

Returns the current horizontal sample rate.

Group Horizontal

Syntax

```
HORizontal:SAMPLERate?
```

HORizontal:SCAle

Sets or returns the time base horizontal scale.

Group Horizontal

Syntax

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

Arguments <NR3> specifies the range from 2 ns to 100 s, depending on the oscilloscope model.

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

HORIZONTAL:SCALE? might return :HORIZONTAL:MAIN:SCALE 2.0000E-06 indicating that the main scale is currently set to 2 µs per division.

HORizontal:TRIGger:POSition

Sets the horizontal position when delay mode is OFF. It is similar to HORizontal:POSition.

Group Horizontal

Syntax

```
HORizontal:TRIGger:POSition
HORizontal:TRIGger:POSition?
```

Examples HORIZONTAL:TRIGGER:POSITION? might return HORIZONTAL:TRIGGER:POSITION 50 indicating that the horizontal trigger position is set to 50%.

ID? (Query Only)

Returns identifying information about the oscilloscope and related firmware.

Group Miscellaneous

Syntax ID?

Related Commands *IDN?

Examples ID? might return TEK/DPO2024,CF:91.1CT,FV:v1.0000. This indicates the oscilloscope model number, configured format, and firmware version number.

*IDN? (Query Only)

Returns the oscilloscope identification code.

Group Miscellaneous

Syntax `*IDN?`

Related Commands ID?

Examples `*IDN?` might return `:TEKTRONIX,DPO2024,SN123456789,CF:91.1CT FV:v1.00000` indicating that the oscilloscope model number, serial number, configured number, and firmware version number.

LANGuage

Sets or returns the user interface display language. This command only affects the oscilloscope displayed language. Remote commands and their responses are always in English.

Group Miscellaneous

Syntax

```
LANGuage
{ENGLish|FRENch|GERMan|ITALian|SPANish|PORTUguese|JAPAnese|
KOREan|RUSSian|SIMPlifiedchinese|TRADitionalchinese}
LANGuage?
```

Examples `LANGUAGE?` might return `:LANGUAGE ENGLISH`.

LOCk

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

When the front panel is locked, neither theFPAnel:PRESS nor the FPAnel:TURN commands will work. They will not generate an error event either. You can work around this by using the appropriate programmatic interface commands, instead of the front-panel commands. For example, to set the trigger level to 50%, you could use TRIGger:A SETLevel. To force a trigger, you could use TRIGger FORCe.

Group Miscellaneous

Syntax

```
LOCk {ALL|NONe}
LOCk?
```

Related Commands UNLock

Arguments `ALL` disables all front-panel controls.

`NONe` enables all front-panel controls. This is equivalent to the UNLock `ALL` command.

Examples `LOCK ALL` locks the front-panel controls.

`LOCK?` might return `:LOCK NONE` indicating that the front-panel controls are enabled by this command.

*LRN? (Query Only)

Returns the commands that list the oscilloscope settings except for configuration information for the calibration values, the WFMInpre? query, and the WFMOutpre? query. This query allows you to record or "learn" the current oscilloscope settings. You can use these commands to return the oscilloscope to the state it was in when you made the *LRN? query. This command is identical to the SET? Command.

Group Miscellaneous

Syntax `*LRN?`

Related Commands SET?

Examples `*LRN?` might return a long response, part of which could be as follows:

```
:ACQUIRE:STOPAFTER RUnsTOP;STATE 1;MODE SAMPLE;NUMENV
INFINITE;NUMAVG 16;SAMPLINGMODE RT;:HEADER 1;:LOCK
NONE;:LANGUAGE ENGLISH;:VERBOSE 1;:ALIAS:STATE
0;:DISPLAY:COLOR:PALETTE NORMAL;:DISPLAY:STYLE:DOTSONLY
0;:DISPLAY:PERSISTENCE 0.0000;CLOCK 1;GRATICULE
FULL;INTEnsITY:WAVEFORM 30;GRATICULE 75;BACKLIGHT
HIGH;:HARDCOPY :INKSAVER OFF;LAYOUT LANDSCAPE;PREVIEW
0;:SAVE:IMAGE:FILEFORMAT BMP;:SAVE:WAVEFORM:FILEFORMAT
```

```
INTERNAL;:SAVE:ASSIGN:TYPE SETUP;:TRIGGER:A:MODE
AUTO;TYPE EDGE;LE VEL 20.0000E-3;LEVEL:CH1
20.0000E-3;CH2 0.0000;CH3 0.0000;CH4 0.0000;:TRIGGER:A:
UPPERTHRESHOLD:CH1 1.4000;CH2 800.0000E-3;CH3
800.0000E-3;CH4 800.0000E-3;:TRIGG ER:A:LOWERTHRESHOLD:CH1
20.0000E-3;CH2 0.0000;CH3 0.0000;CH4 0.0000;:TRIGGER:A:H
OLDOFF:TIME 20.0000E-9;:TRIGGER:A:EDGE:SOURCE
CH1;COUPLING DC;SLOPE RISE;:TRIGGER:A:LOGIC:CLASS
SETHOLD;FUNCTION AND;THRESHOLD:CH1 20.0000E-3;CH2
0.0000;CH3 0.0 000;CH4 0.0000;:TRIGGER:A:LOGIC:INPUT:CH1
X;CH2 X;CH3 X;CH4 X;CLOCK:SOURCE NONE; EDGE
RISE;:TRIGGER:A:LOGIC:PATTERN:INPUT:CH1 X;CH2
X;CH3 X;CH4 X;:TRIGGER:A:LOGIC:PATTERN:WHEN
TRUE;WHEN:LESSLIMIT 4.0000E-9;LIMIT
4.0000E-9;:TRIGGER:A:SETHOLD:CLOCK:SOURCE CH1;EDGE
RISE;THRESHOLD 20.0000E-3;:TRIGGER:A:SETHOLD:DATA:SOURCE
CH2;THRESHOLD 0.0000;:TRIGGER:A:SETHOLD:HOLDTIME
4.0000E-9;SETTIME 4.0000E-9; :TRIGGER:A:PULSE:CLASS
TRAnsITION;:TRIGGER:A:PULSEWIDTH:SOURCE
CH1;POLARITY POSITIVE;WHEN LESSTHAN;WIDTH
4.0000E-9;:TRIGGER:A:RUNT:SOURCE CH1;POLARITY POSITIVE; WHEN
OCCURS;WIDTH 4.0000E-9;:TRIGGER:A:TRAnsITION:SOURCE CH1
```

...

MARK

Moves to the next or previous reference mark on the waveform. Returns the current mark position.

Group Mark

Syntax

```
MARK {NEXT|PREVious}
MARK?
```

Arguments `NEXT` moves to the next reference mark on the right.

`PREVious` moves to the next reference mark on the left.

MARK:CREATE (No Query Form)

Creates a mark on a specified waveform or all waveforms in a column.

Group Mark

Syntax

```
MARK:CREATE {CH<x>|MATH|B<x>|REF<x>|COLUMN|DIGItal}
```

Arguments

`CH<x>` creates the mark on a channel waveform, where <x> is the channel number.

`MATH` creates the mark on the math waveform.

`B<x>` creates the mark on a bus waveform, where <x> is the bus number.

`REF<x>` creates the mark on a reference waveform, where <x> is the reference waveform number.

`COLUMN` creates marks on all waveforms in the current zoom pixel column.

`DIGItal` creates marks on a digital waveform. (An error will result if no digital channel is turned on.)

MARK:DELEte (No Query Form)

Deletes a mark on a particular waveform, all waveforms in a column, the selected mark, or all marks.

Group

Mark

Syntax

```
MARK:DELEte {CH<x>|MATH|B<x>|REF<x>|COLUMN|DIGItal}
```

Arguments

`CH<x>` deletes the mark on a channel waveform, where <x> is the channel number.

`MATH` deletes the mark on the math waveform.

`B<x>` deletes the mark on a bus waveform, where <x> is the bus number.

`REF<x>` deletes the mark on a reference waveform, where <x> is the reference waveform number.

`COLUMN` deletes marks on all waveforms in the current zoom pixel column.

`DIGItal` deletes all marks on all digital channels.

MARK:FREE? (Query Only)

Returns how many marks are available for use.

Group

Mark

Syntax

```
MARK:FREE?
```

MARK:SELected:END? (Query Only)

Returns the end of the selected mark, 0 to 100% of the waveform.

Group Mark

Syntax `MARK:SELected:END?`

MARK:SELected:FOCUS? (Query Only)

Returns the focus of the selected mark, 0 to 100% of the waveform.

Group Mark

Syntax `MARK:SELected:FOCUS?`

MARK:SELected:MARKSINCOLumn? (Query Only)

Returns the number of marks in the current zoom pixel column.

Group Mark

Syntax `MARK:SELected:MARKSINCOLumn?`

MARK:SELected:OWNer? (Query Only)

Returns the owner of the selected mark.

Group Mark

Syntax `MARK:SELected:OWNer?`

Returns `<QString>` is the owner of the mark.

Examples `MARK:SELECTED:OWNER?` might return: `USER, SEARCH1`

MARK:SELected:SOURCE? (Query Only)

Returns the source waveform for the selected mark.

Group Mark

Syntax `MARK:SELected:SOURCE?`

MARK:SELected:STARt? (Query Only)

Returns the starting point of the selected mark, 0 to 100% of the waveform.

Group Mark

Syntax `MARK:SELected:STARt?`

MARK:SELected:STATe? (Query Only)

Returns the on or off state of the selected mark. The selected mark is at or near the center of the screen. If you press the front-panel Set/Clear button, this mark will disappear.

Group Mark

Syntax `MARK:SELected:STATe?`

MARK:SELected:ZOOm:POSition? (Query Only)

Returns the position of the selected mark, 0 to 100% of the zoom overview window.

Group Mark

Syntax `MARK:SELected:ZOOm:POSition?`

MARK:TOTal? (Query Only)

Returns how many marks are currently in use.

Group Mark

Syntax

```
MARK:TOTal?
```

MATH[1]:LABel

Sets or queries the waveform label for the math waveform.

Group Math

Syntax

```
MATH[1]:LABel <QString>
MATH[1]:LABel?
```

Arguments `<QString>` is the quoted string used as the label for the math waveform.

Examples `MATH:LABEL "Output"` sets the label for the math waveform to Output.

`MATH:LABEL?` might return `MATH:LABEL "Sum of channel 1 and channel 2"` indicating that the current label for the math waveform.

MATH[1]? (Query Only)

Returns the definition of the math waveform. The returned data depends on the setting of the MATH[1]:TYPe command.

Group Math

Syntax

```
MATH[1]?
```

Related Commands MATH[1]:TYPe

Examples `MATH?` or `MATH1?` might return `:MATH:TYPE DUAL;DEFINE "CH1+CH2";VERTICAL:SCALE 100.0000E-3;POSITION 0.0000;UNITS`

```
"V";:MATH:HORIZONTAL:SCALE 4.0000E-6;POSITION 50.0000;UNITS
"s";:MATH:SPECTRAL:MAG DB;WINDOW HANNING;:MATH:LABEL“”
```

MATH[1]:DEFine

Sets or returns the current math function as a text string.

Dual math is defined if the string is of the form `<wfm> <operation> <wfm>`, where the <wfm>s are any combination of live channels or reference waveforms, <operation> is any of +, -, or *, and the MATH[1]:TYPe is DUAL.

FFT math is defined if the string is in the form `FFT(<wfm>)`, where <wfm> is any live channel or reference waveform, and theMATH[1]:TYPe is FFT.

On the front panel, the Dual Wfm Math, and FFT menus contain controls that allow building equivalent math expressions to those described above.

Group

Math

Syntax

```
MATH[1]:DEFine <QString>
MATH[1]:DEFine?
```

Related Commands

MATH[1]:TYPe

Arguments

`<QString>` quoted string argument is the mathematical expression that defines the waveform.

Table 2-39: Advanced Math expression elements

Expression	Description
CH1–CH4, REF1, REF2	Specifies a waveform data source.
FFT(	Executes a Fast Fourier Transform. The FFT operator must be the first (left-most) operator in an expression. All these operations must end with a right parenthesis.
AMPlitude(, AREa(, BURst(, CARea(	Executes the selected measurement operation on the waveform (active or reference) that follows. All these operations must end with a right parenthesis.
+ , - , *	Executes an addition, subtraction, or multiplication operation on the following expression.
<, >, <=, >=, ==, ≠, ||, &&	Executes relational and logical operations.

Examples

`MATH1:DEFINE "CH1+CH2"` adds the Ch 1 waveform and Ch 2 waveform, storing the results in Math 1.

`MATH:DEFINE?` might return `:MATH1:DEFINE "CH2*REF2"` as the expression that defines Math 1.

MATH[1]:HORizontal:POSition

Sets or returns the math horizontal display position for FFT or (non-live), live math reference waveforms.

Group

Math

Syntax

```
MATH[1]:HORizontal:POSition <NR3>
MATH[1]:HORizontal:POSition?
```

Arguments

`<NR3>` is the % of the math waveform that precedes center screen. It can vary from 0.0 to 100.0.

Examples

`MATH:HORIZONTAL:POSITION 10` sets the horizontal position to 10% pretrigger

MATH[1]:HORizontal:SCAle

Sets or returns the math horizontal display scale for FFT or (non-live), live math waveforms. The horizontal scale of a dual math waveform with a channel source waveform is set through the HORizontal:SCAle command.

Group

Math

Syntax

```
MATH[1]:HORizontal:SCAle <NR3>
MATH[1]:HORizontal:SCAle?
```

Arguments

`<NR3>` is the math horizontal scale in seconds.

Examples

`MATH:HORIZONTAL:SCALE?` might return `MATH:HORIZONTAL:SCALE 2.0E-4` indicating that the math horizontal scale is 200 µ

MATH[1]:HORizontal:UNIts

Returns the math waveform horizontal measurement unit value.

Group Math

Syntax

```
MATH[1]:HORizontal:UNIts?
```

Examples `MATH:HORIZONTAL:UNITS?` might return `MATH:HORIZONTAL:UNITS "?"`

indicating that the math horizontal unit label for unknown values is the default question mark unit.

MATH[1]:SPECTral:GATing:INDICators

Enables or disables the display of indicators that show the portion of the source waveform record used to compute the math FFT waveform.

Group Math

Syntax

```
MATH[1]:SPECTral:GATing:INDICators {<NR1>|OFF|ON}
MATH[1]:SPECTral:GATing:INDICators?
```

Arguments `OFF` turns off the display of indicators.

`ON` turns on the display of indicators.

`<NR1>` = 0 turns off the display of indicators. Any other value turns on the display of indicators.

MATH[1]:SPECTral:GATing:INDICators:END? (Query Only)

Returns the ending point in the source waveform record used for computing the FFT

Group Math

Syntax

```
MATH[1]:SPECTral:GATing:INDICators:END?
```

Returns A value between 0 and 1, which represents the fractional portion of the source record.

MATH[1]:SPECTral:GATing:INDICators:STARt? (Query Only)

Returns the starting point in the source waveform record used for computing the FFT math waveform.

Group Math

Syntax

```
MATH[1]:SPECTral:GATing:INDICators:STARt?
```

Returns A value between 0 and 1, which represents the fractional portion of the source record.

MATH[1]:SPECTral:MAG

Sets or returns the units of the Spectral Magnification function in the math string.

Group Math

Syntax

```
MATH[1]:SPECTral:MAG {LINEAr|DB}
MATH[1]:SPECTral:MAG?
```

Arguments `LINEAR` sets the SpectralMag units to linear.

`DB` sets the SpectralMag units to decibels.

Examples `MATH1:SPECTRAL:MAG DB` sets the SpectralMag units for Math1 to decibels.

`MATH1:SPECTRAL:MAG?` might return `:MATH1:SPECTRAL:MAG DB` indicating that the SpectralMag units for Math1 are set to decibels.

MATH[1]:SPECTral:NYQUISTFreq? (Query Only)

Returns the Nyquist frequency of the FFT math waveforms.

Group Math

Syntax

```
MATH[1]:SPECTral:NYQUISTFreq?
```

MATH[1]:SPECTral:WINdow

Sets or returns the window function for the spectral analyzer input data for the specified math waveform. A spectral window determines what the filter shape of the spectral analyzer will be in the frequency domain. It can be described by a mathematical function that is multiplied point-by-point times the input data to the spectral analyzer.

Group

Math

Syntax

```
MATH[1]:SPECTral:WINdow
{RECTangular|HAMming|HANning|BLAckmanharris}
MATH[1]:SPECTral:WINdow?
```

Arguments

`RECTangular` window function is equivalent to multiplying all gate data by one.

`HAMming` window function is based on a cosine series.

`HANning` window function is based on a cosine series.

`BLAckmanharris` window function is based on a cosine series.

Examples

`MATH1:SPECTRAL:WINDOW HANNING` applies a Hanning window to the spectral analyzer input data.

`MATH1:SPECTRAL:WINDOW?` might return `:MATH1:SPECTRAL:WINDOW HAMMING` indicating that the window function used to multiply the spectral analyzer input data is the Hamming window.

MATH[1]:TYPe

This command specifies the math waveform type - either dual or FFT. This command is typically used along with MATH[1]:DEFine, which specifies the current mathematical expression as a text string which defines the waveform.

Group

Math

Syntax

```
MATH[1]:TYPe {DUAL|FFT}
MATH[1]:TYPe?
```

Related Commands MATH[1]:DEFine

Arguments `DUAL` sets the math waveform mode to dual waveform math.

`FFT` sets the math waveform mode to FFT math.

Examples `MATH:TYPE FFT` sets the math waveform mode to FFT.

`MATH:TYPE FFT;:MATH:DEFINE "FFT( CH1 )"` sets the math type to FFT and displays an FFT waveform of the channel 1 waveform, using the current FFT scale and window settings.

MATH[1]:VERTical:POSition

Sets or returns the vertical position of the currently selected math type.

Group Math

Syntax

```
MATH[1]:VERTical:POSition <NR3>
MATH[1]:VERTical:POSition?
```

Related Commands CH<x>:POSition, REF<x>:VERTical:POSition

Arguments `<NR3>` is the desired position in divisions from the center graticule.

Examples `MATH1:VERTICAL:POSITION 1.3E+00` positions the Math 1 input signal 1.3 divisions higher than a position of 0.

`MATH1:VERTICAL:POSITION?` might return `:MATH1:VERTICAL:POSITION -1.3000E+00` indicating that the current position of Math 1 is 1.3 divisions below the center graticule.

MATH[1]:VERTical:SCAle

Sets or returns the vertical scale of the currently selected math type.

Group Math

Syntax

```
MATH[1]:VERTical:SCAle <NR3>
MATH[1]:VERTical:SCAle?
```

Related Commands CH<x>:SCAle, REF<x>:VERTical:SCAle

Arguments <NR3> is the scale-per-division in the current math vertical units. The range is from 1.0E-09 through 5.0.

Examples `MATH1:VERTICAL:SCALE 100E-03` sets the Math scale to 100 mV per division.

`MATH:VERTICAL:SCALE?` might return `:MATH:VERTICAL:SCALE 1.0000E+00` indicating that the current scale setting of Math is 1 V per division.

MATH[1]:VERTical:UNIts? (Query Only)

Returns the math waveform vertical measurement unit value.

Group Math

Syntax `MATH[1]:VERTical:UNIts?`

Examples `MATH:VERTICAL:UNITS?` might return `MATH:VERTICAL:UNITS "V"` indicating that the math vertical unit label is volts.

MEASUrement? (Query Only)

Returns all measurement parameters.

Group Measurement

Syntax `MEASUrement?`

Examples `MEASUREMENT?` might return `:MEASUREMENT:IMMED:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE;:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH2;:MEASUREMENT:MEAS1:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE;:MEASUREMENT:MEAS1:STATE 1;TYPE FREQUENCY;UNITS "Hz";SOURCE1 CH1;SOURCE2 CH2;COUNT 0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM 0.0 000;STDDEV 0.0000;:MEASUREMENT:MEAS2:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE;:MEASUREMENT:MEAS2:STATE 1;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH2;COUNT 0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM 0.0000;STDDEV 0.0000;:MEASUREMENT:MEAS3:`

```
DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2
RISE;:MEASUREMENT:MEAS3:STATE 1;TYPE PK2PK;UNITS "V";SOURCE1
CH1;SOURCE2 CH2;COUNT 0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM
0.0000;STDDEV 0.0000;:MEASUREMENT:MEAS4:DELAY:DIRECTION
FORWARDS;EDGE1 RISE;EDGE2 RISE;:MEASUREMENT:MEAS4:STATE
0;TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH2;COUNT
0;MAXIMUM 0.0000;MEAN 0.0000;MINIMUM 0.0000;STDDEV
0.0000;:MEASUREMENT:METHOD AUTO;REFLEVEL:METHOD
PERCENT;ABSOLUTE:HIGH 0.0000;LOW 0.0000;MID1
0.0000;MID2 0.0000;:MEASUREMENT:REFLEVEL:PERCENT:HIGH
90.0000;LOW 10.0000;MID1 50.0000;MID2
50.0000;:MEASUREMENT:INDICATORS:STATE
OFF;NUMHORZ 0;NUMVERT 0;HORZ1 99.0000E +36;HORZ2
99.0000E+36;HORZ3 99.0000E+36;HORZ4 99.0000E+36;VERT1
99.0000E+36;VERT2 99.0000E+36;VERT3 99.0000E+36;VERT4
99.0000E+36;:MEASUREMENT:STATISTICS:MODE OFF;WEIGHTING
32;:MEASUREMENT:GATING SCREEN.
```

MEASUrement:CLEARSNapshot (No Query Form)

Removes the measurement snapshot display.

Group Measurement

Syntax

```
MEASUrement:CLEARSNapshot
```

Related Commands CLEARMenu

MEASUrement:GATing

Specifies or returns the measurement gating setting.

Group Measurement

Syntax

```
MEASUrement:GATing {OFF|SCREen|CURSor}
MEASUrement:GATing?
```

Arguments `OFF` turns off measurement gating (full record).

`SCREen` turns on gating, using the left and right edges of the screen.

`CURSor` limits measurements to the portion of the waveform between the vertical bar cursors, even if they are off screen.

Examples

`MEASUREMENT:GATING CURSOR` turns on measurement gating using the cursors as limits.

`MEASUREMENT:GATING?` might return `:MEASUREMENT:GATING CURSOR` indicating that measurements are limited to the portion of the waveform between the vertical bar cursors.

MEASUrement:IMMed? (Query Only)

Returns all immediate measurement setup parameters.

Group

Measurement

Syntax

`MEASUrement:IMMed?`

Examples

`MEASUREMENT:IMMED?` might return
`:MEASUREMENT:IMMED:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE;:MEASUREMENT:IMMED:TYPE PERIOD;UNITS "s";SOURCE1 CH1;SOURCE2 CH2`

MEASUrement:IMMed:DELay? (Query Only)

Returns information about the immediate delay measurement. This command is equivalent to viewing the delay measurement settings on the measurement readout.

Group

Measurement

Syntax

`MEASUrement:IMMed:DELay?`

Examples

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

MEASUrement:IMMed:DELay:DIRection

Sets or returns the starting point and direction that determines the delay "to" edge when taking an immediate delay measurement.

NOTE. *Use the MEASUrement:IMMed:SOUrce2 command to specify the delay "to" waveform.*

Group Measurement

Syntax

```
MEASUrement:IMMed:DELay:DIRection {BACKWards|FORWards}
MEASUrement:IMMed:DELay:DIRection?
```

Related Commands MEASUrement:IMMed:SOUrce2

Arguments

`BACKWards` starts the search at the end of the waveform and looks for the last rising or falling edge in the waveform.

`FORWards` starts the search at the beginning of the waveform and looks for the first rising or falling edge in the waveform.

Examples

`MEASUREMENT:IMMED:DELAY:DIRECTION FORWARDS` starts searching from the beginning of the waveform record and looks for the first rising or falling edge.

`MEASUREMENT:IMMED:DELAY:DIRECTION?` might return `:MEASUREMENT:IMMED:DELAY:DIRECTION BACKWARDS` indicating that searching begins at the end of the waveform record and looks for the last rising or falling edge.

MEASUrement:IMMed:DELay:EDGE<x>

Sets or returns the slope of the edge the oscilloscope uses for the delay "from" or "to" waveform when taking an immediate delay measurement.

Group Measurement

Syntax

```
MEASUrement:IMMed:DELay:EDGE<x> {FALL|RISe}
MEASUrement:IMMed:DELay:EDGE<x>?
```

Related Commands

MEASUrement:IMMed:SOUrce1

MEASUrement:IMMed:SOUrce2

Arguments

<x> specifies which waveform to use, where <x> = 1 is the "from" waveform, and <x> = 2 is the "to" waveform.

FALL specifies the falling edge.

RISe specifies the rising edge.

Examples

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

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

MEASUrement:IMMed:SOUrce1

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

Digital channels (D<x>) are available as a measurement source for time, edge and pulse measurements such as Period, Frequency, Pos Width, Neg Width, Pos Duty Cycle, Neg Duty Cycle, Pos/Neg Edges and Pos/Neg Pulses, Delay and Phase.

NOTE. *If you do not specify a numerical suffix, the source is assumed to be SOURCE 1.*

Group

Measurement

Syntax

```
MEASUrement:IMMed:SOUrce1 {CH<x>|MATH<y>|REF<x>}
MEASUrement:IMMed:SOUrce1?
```

Related Commands

MEASUrement:IMMed:SOUrce2

Arguments

CH<x> is an input channel waveform. The x variable can be expressed as an integer, where x is the channel number.

MATH<y> is a math waveform. The y variable can be expressed as an integer of 1.

REF<X> is a reference waveform. The x variable can be expressed as an integer, where x is the reference channel number.

Examples

`MEASUREMENT:IMMED:SOURCE1 MATH1`

specifies Math1 as the immediate measurement source.

`MEASUREMENT:IMMED:SOURCE1?` might return `:MEASUREMENT:IMMED:SOURCE1 CH3` indicating that channel 3 is the immediate measurement source.

MEASUrement:IMMed:SOUrce2

Sets or returns the source to measure "to" for phase or delay immediate measurements.

NOTE. *Source2 measurements only apply to phase and delay measurement types, which require both a target (Source1) and reference (Source2) source.*

Group

Measurement

Syntax

```
MEASUrement:IMMed:SOUrce2 {CH<x>|MATH<y>|REF<x>}
MEASUrement:IMMed:SOUrce2?
```

Related Commands

MEASUrement:IMMed:SOUrce1

Arguments

`CH<x>` is an input channel waveform, where x is the channel number.

`MATH<y>` is a math waveform. The y variable can be expressed as an integer of 1.

`REF<X>` is a reference waveform, where x is the reference channel number.

Examples

`MEASUREMENT:IMMED:SOURCE2 REF2` sets the waveform in reference memory location 2 as the delay "to" source when making delay measurements.

`MEASUREMENT:IMMED:SOURCE2?` might return `:MEASUREMENT:IMMED:SOURCE2 MATH1` indicating that Math1 is the immediate measurement source.

MEASUrement:IMMed:SOUrce<x>

For SOURce1: Sets or returns the source for all single channel measurements. For delay or phase measurements, sets or returns the waveform to measure "from".

For SOUrce2: Sets or returns the waveform to measure "to" when taking a delay measurement or phase measurement.

Group Measurement

Syntax

```
MEASUrement:IMMed:SOUrce<x> {CH1|CH2|CH3|CH4|MATH}
MEASUrement:IMMed:SOUrce<x>?
```

Arguments `CH1–CH4` or `MATH` is the source waveform.

MEASUrement:IMMed:TYPe

Sets or returns the immediate measurement type.

Group Measurement

Syntax

```
MEASUrement:IMMed:TYPe
{AMPlitude|AREa|BURst|CARea|CMEan|CRMs|DELay|FALL|FREQuency
|HIGH|LOW|MAXimum|MEAN|MINImum|NDUty|NEDGECount|NOVershoot
|NPULSECount|NWIdth|PEDGECount|PDUty
|PERIod|PHAse|PK2Pk|POVershoot|PPULSECount|PWIdth|RISe|RMS}
MEASUrement:IMMed:TYPe?
```

Arguments `AMPlitude` measures the amplitude of the selected waveform. In other words, it measures the high value less the low value measured over the entire waveform or gated region. This measurement is available only on DPO models.

Amplitude = High – Low

`AREa` measures the voltage over time. The area is over the entire waveform or gated region and is measured in volt-seconds. The area measured above the ground is positive, while the area below ground is negative. This measurement is available only on DPO models.

`BURst` measures the duration of a burst. The measurement is made over the entire waveform or gated region.

`CARea` (cycle area) measures the voltage over time. In other words, it measures, in volt-seconds, the area over the first cycle in the waveform or the first cycle in the gated region. The area measured above the common reference point is positive, while the area below the common reference point is negative. This measurement is available on DPO and MSO models.

`CMEan` (cycle mean) measures the arithmetic mean over the first cycle in the waveform or the first cycle in the gated region. This measurement is available only on DPO and MSO models.

`CRMs` (cycle rms) measures the true Root Mean Square voltage over the first cycle in the waveform or the first cycle in the gated region. This measurement is available only on DPO and MSO models.

`DELay` measures the time between the middle reference (default = 50%) amplitude point of the source waveform and the destination waveform.

`FALL` measures the time taken for the falling edge of the first pulse in the waveform or gated region to fall from a high reference value (default is 90%) to a low reference value (default is 10%). This measurement is available only on DPO models.

`FREQuency` measures the first cycle in the waveform or gated region. Frequency is the reciprocal of the period and is measured in hertz (Hz), where 1 Hz = 1 cycle per second.

`HIGH` measures the High reference (100% level, sometimes called Topline) of a waveform. This measurement is available only on DPO models.

`LOW` measures the Low reference (0% level, sometimes called Baseline) of a waveform. This measurement is available only on DPO models.

`MAXimum` finds the maximum amplitude. This value is the most positive peak voltage found. It is measured over the entire waveform or gated region. This measurement is available only on DPO models.

`MEAN` amplitude measurement finds the arithmetic mean over the entire waveform or gated region. This measurement is available only on DPO models.

`MINImum` finds the minimum amplitude. This value is typically the most negative peak voltage. It is measured over the entire waveform or gated region. This measurement is available only on DPO models.

`NDUty` (negative duty cycle) is the ratio of the negative pulse width to the signal period, expressed as a percentage. The duty cycle is measured on the first cycle in the waveform or gated region.

Negative Duty Cycle = ((Negative Width) / Period) × 100%

`NEDGECount` is the count of falling edges.

`NOVershoot` (negative overshoot) finds the negative overshoot value over the entire waveform or gated region. This measurement is available only on DPO models.

Negative Overshoot = ((Low – Minimum) / Amplitude) × 100%)

`NPULSECount` is the count of negative pulses.

`NWIdth` (negative width) measurement is the distance (time) between the middle reference (default = 50%) amplitude points of a negative pulse. The measurement is made on the first pulse in the waveform or gated region.

`PDUty` (positive duty cycle) is the ratio of the positive pulse width to the signal period, expressed as a percentage. It is measured on the first cycle in the waveform or gated region.

Positive Duty Cycle = ((Positive Width / Period) × 100%

`PEDGECount` is the count of rising edges.

`PERIod` is the time required to complete the first cycle in a waveform or gated region. Period is the reciprocal of frequency and is measured in seconds.

`PHAse` measures the phase difference (amount of time a waveform leads or lags the reference waveform) between two waveforms. The measurement is made between the middle reference points of the two waveforms and is expressed in degrees, where 360° represents one waveform cycle.

`PK2Pk` (peak-to-peak) finds the absolute difference between the maximum and minimum amplitude in the entire waveform or gated region. This measurement is available only on DPO models.

`POVershoot` is the positive overshoot value over the entire waveform or gated region. This measurement is available only on DPO models.

Positive Overshoot = ((Maximum – High) / Amplitude) ×100%

`PPULSECount` is the count of positive pulses.

`PWIdth` (positive width) is the distance (time) between the middle reference (default = 50%) amplitude points of a positive pulse. The measurement is made on the first pulse in the waveform or gated region.

`RISe` timing measurement finds the rise time of the waveform. The rise time is the time it takes for the leading edge of the first pulse encountered to rise from a low reference value (default is 10%) to a high reference value (default is 90%). This measurement is available only on DPO models.

`RMS` amplitude measurement finds the true Root Mean Square voltage in the entire waveform or gated region. This measurement is available only on DPO models.

Examples

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

`MEASUREMENT:IMMED:TYPE?` might return `:MEASUREMENT:IMMED:TYPe RMS` indicating that the immediate measurement is the true Root Mean Square voltage.

MEASUrement:IMMed:UNIts? (Query Only)

Returns the units of the immediate measurement:

VOLTS, VOLTS SQUARED, SEC, HERTZ, PERCENT, DIVS, SAMPLES, OHMS, AMPS, WATTS, MINUTES, DEGREES, UNKNOWN, AMPS SQUARED, HOURS, DAYS, DB, BYTES, INVERSE HERTZ, IRE, V OVER V, V OVER A, VOLTS WATTS, V OVER W, VOLTS DB, V OVER DB, A OVER V, A OVER A, AMPS WATTS, A OVER W, AMPS DB, A OVER DB, WATTS VOLTS, W OVER V, WATTS AMPS, W OVER A, WATTS SQUARED, W OVER W, WATTS DB, W OVER DB, DB VOLTS, DB OVER V, DB AMPS, DB OVER A, DB WATTS, DB OVER W, DB SQUARED, DB OVER DB, VOLTS SEC, AMPS SEC, WATTS SEC, V OVER S, A OVER S, W OVER S

Group Measurement

Syntax `MEASUrement:IMMed:UNIts?`

Examples `MEASUREMENT:IMMED:UNITS?` might return

`:MEASUREMENT:IMMED:UNIts "s"`

indicating that units for the immediate measurement are in seconds.

MEASUrement:IMMed:VALue? (Query Only)

Returns the value of the measurement specified by the MEASUrement:IMMed:TYPe command. The measurement is immediately taken on the source(s) specified by a MEASUrement:IMMed:SOUrce1 command.

NOTE. *A change to HORizontal:MAIn:SCALe or CH<x>:SCALe will not necessarily have taken affect if immediately followed by this command.*

Group Measurement

Syntax `MEASUrement:IMMed:VALue?`

Related Commands MEASUrement:IMMed:TYPe, MEASUrement:IMMed:SOUrce1, *ESR?, ALLEv?

Examples `MEASUREMENT:IMMED:VALUE?` might return `:MEASUREMENT:IMMED:VALUE 9.9000E+37.` If the measurement has an error or warning associated with it, then an item is added to the error queue. The error can be checked for with the *ESR? and ALLEv? commands.

MEASUrement:INDICators? (Query Only)

Returns all measurement indicator parameters.

Group Measurement

Syntax `MEASUrement:INDICators?`

Examples `MEASUREMENT:INDICATORS?` might return `MEASUREMENT:INDICATORS:STATE MEAS1;NUMHORZ 0;NUMVERT 4;HORZ1 7.5E0;HORZ2 -3.400000095367E0;HORZ3 0.0E0;HORZ4 0.0E0;VERT1 -6.351123E-6;VERT2 -3.179753E-6;VERT3 -6.40943E-6;VERT4 -6.403E-6`

MEASUrement:INDICators:HORZ<x>? (Query Only)

Returns the position of the specified horizontal measurement indicator <x>, where <x> can be 1, 2, 3, or 4.

Group Measurement

Syntax `MEASUrement:INDICators:HORZ<x>?`

Examples `MEASUREMENT:INDICATORS:HORZ1?` might return `MEASUREMENT:INDICATORS:HORZ1 -2.0E-3` indicating that horizontal indicator1 has a value of –2 mV.

MEASUrement:INDICators:NUMHORZ? (Query Only)

Returns the number of horizontal measurement indicators currently being displayed.

Group Measurement

Syntax `MEASUrement:INDICators:NUMHORZ?`

Examples `MEASUREMENT:INDICATORS:NUMHORZ?` might return `MEASUREMENT:INDICATORS:NUMHORZ 2` indicating that there are currently

2 horizontal lines drawn on the graticule. The indicators show where the measurement specified by MEASUrement:INDICators:STATE is being performed.

MEASUrement:INDICators:NUMVERT? (Query Only)

Returns the number of vertical measurement indicators currently being displayed.

Group Measurement

Syntax

```
MEASUrement:INDICators:NUMVERT?
```

Examples `MEASUREMENT:INDICATORS:NUMVERT?` might return `MEASUREMENT:INDICATORS:NUMVERT 2` indicating that there are currently 2 vertical lines drawn on the graticule. The indicators show where the measurement specified by MEASUrement:INDICators:STATE is being performed.

MEASUrement:INDICators:STATE

Sets or returns the state of visible measurement indicators.

Group Measurement

Syntax

```
MEASUrement:INDICators:STATE {OFF|MEAS<x>}
MEASUrement:INDICators:STATE?
```

Arguments `OFF` turns the visible measurement indicators off.

`MEAS<x>` displays the visible measurement indicators for measurement <x>, where <x> can be 1, 2, 3, or 4.

NOTE. *There must be an active measurement before you can activate an indicator for a specified measurement.*

Examples `MEASUREMENT:INDICATORS:STATE MEAS2` turns on the display of visible measurement indicators for measurement 2.

`MEASUREMENT:INDICATORS:STATE?` might return `MEASUREMENT:INDICATORS:STATE OFF` indicating that no measurement indicators are active.

MEASUrement:INDICators:VERT<x>? (Query Only)

Returns the value of the specified vertical measurement indicator <x> from the trigger point, where <x> can be 1, 2, 3, or 4. A negative value means that the indicator is positioned earlier in the waveform record than the trigger point.

Group Measurement

Syntax `MEASUrement:INDICators:VERT<x>?`

Examples `MEASUREMENT:INDICATORS:VERT2?` might return `MEASUREMENT:INDICATORS:VERT2 -3.724507E-6` indicating that the second measurement indicator is positioned 3.72 µs before the trigger point.

MEASUrement:MEAS<x>? (Query Only)

Returns all measurement parameters for the specified active measurement <x>.

Group Measurement

Syntax `MEASUrement:MEAS<x>?`

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

Returns the number of values accumulated for this measurement since the last statistical reset. Values may be ignored if they generated an error. Measurements are specified by x, which ranges from 1 through 4.

Group Measurement

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

Examples `MEASUREMENT:MEAS3:COUNT?` might return `:MEASUREMENT:MEAS3:COUNT 3247.`

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

Returns the delay measurement parameters for the measurement specified by <x>, which ranges from 1 through 4.

Group Measurement

Syntax

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

Examples `MEASUREMENT:MEAS1?` might return `:MEASUREMENT:MEAS1:DELAY:DIRECTION FORWARDS;EDGE1 RISE;EDGE2 RISE`.

MEASUrement:MEAS<x>:DELay:DIRection

Sets or returns the starting point and direction that determines the delay "to" edge when taking a delay measurement. Use the MEASUrement:MEAS<x>:SOURCE2 command to specify the waveform.

Group Measurement

Syntax

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

Related Commands MEASUrement:MEAS<x>:SOURCE2

Arguments `BACKWards` means the search starts at the end of the waveform and looks for the last rising or falling edge in the waveform. Use the MEASUrement:MEAS<x>:DELay:EDGE<x> command to specify the slope of the edge.

`FORWards` means the search starts at the beginning of the waveform and looks for the first rising or falling edge in the waveform. Use the MEASUrement:MEAS<x>:DELay:EDGE<x> command to specify the slope of the edge.

Examples `MEASUREMENT:MEAS3:DELAY:DIRECTION BACKWARDS` starts searching from the end of the waveform record.

`MEASUREMENT:MEAS3:DELAY:DIRECTION?` might return `:MEASUREMENT:MEAS3:DELAY:DIRECTION BACKWARDS` indicating that the current search direction is backwards.

MEASUrement:MEAS<x>:DELay:EDGE<x>

Sets or returns the slope of the edge used for the delay "from" or "to" waveform when taking an immediate delay measurement. The waveform is specified by MEASUrement:MEAS<x>:SOURCE[1].

Group Measurement

Syntax

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

Arguments

`<x>` specifies which waveform to use, where <x> = 1 is the "from" waveform, and <x> = 2 is the "to" waveform.

`FALL` specifies the falling edge.

`RISe` specifies the rising edge.

Examples

`MEASUREMENT:MEAS1:DELAY:EDGE1 RISE` specifies that the "from" waveform rising edge be used for the immediate delay measurement.

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

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

Returns the maximum value found for this measurement since the last statistical reset. Measurements are specified by x, which ranges from 1 through 4.

Group Measurement

Syntax

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

Examples

`MEASUREMENT:MEAS3:MAXIMUM?` might return `:MEASUREMENT:MEAS3:MAXIMUM 4.18E-9.`

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

Returns the mean value accumulated for this measurement since the last statistical reset. Measurements are specified by x, which ranges from 1 through 4.

Group Measurement

Syntax

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

Examples `MEASUREMENT:MEAS1:MEAN?` might return `:MEASUREMENT:MEAS1:MEAN 514.71E-09.`

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

Returns the minimum value for this measurement since the last statistical reset. Measurements are specified by <x>, which ranges from 1 through 4.

Group Measurement

Syntax

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

Examples `MEASUREMENT:MEAS1:MINIMUM?` might return `:MEASUREMENT:MEAS1:MINIMUM 1.75E-09.`

MEASUrement:MEAS<x>:SOURCE[1]

Sets or returns the source for all single source measurements and specifies the source to measure "from" when taking a delay measurement or phase measurement. Measurements are specified by <x>, which ranges from 1 through 4.

Group Measurement

Syntax

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

Arguments

`CH<x>` is an input channel waveform, where x is the channel number.

`MATH<y>` is a math waveform, where y is 1.

`REF<x>` is a reference waveform, where x is the reference channel number.

Examples

`MEASUREMENT:MEAS2:SOURCE1 MATH1` specifies Math 1 as the measurement 2 source.

`MEASUREMENT:MEAS1:SOURCE1?` might return `:MEASUREMENT:MEAS1:SOURCE[1] MATH1` indicating that Math1 is the measurement 2 source.

MEASUrement:MEAS<x>:SOURCE2

Sets or returns the reference source to measure "to" when taking a delay measurement or phase measurement. Measurements are specified by <x>, which ranges from 1 through 4.

NOTE. *Source2 measurements only apply to phase and delay measurement types, which require both a target (Source1) and reference (Source2) source.*

Group

Measurement

Syntax

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

Related Commands

MEASUrement:MEAS<x>:TYPe

Arguments

`CH<x>` is an input channel waveform, where x is the channel number.

`MATH<y>` is the math waveform, which is always 1.

`REF<x>` is a reference waveform, where x is the reference channel number.

Examples

`MEASUREMENT:MEAS4:SOURCE2 CH1` specifies CH1 as the delay "to" source when making delay measurement.

`MEASUREMENT:MEAS2:SOURCE2?` might return `:MEASUREMENT:MEAS2:SOURCE2 MATH1` indicating that Math 1 is the measurement 2 source.

MEASUrement:MEAS<x>:SOUrce<x>

For SOURce1: Sets or returns the source for all single channel measurements. For delay or phase measurements, sets or returns the waveform to measure "from".

For SOUrce2: Sets or returns the waveform to measure "to" when taking a delay measurement or phase measurement.

Group Measurement

Syntax

```
MEASUrement:MEAS<x>:SOUrce<x> {CH<x>|MATH|}
MEASUrement:MEAS<x>:SOUrce<x>?
```

Arguments

`CH<x>` is an input channel waveform, where x is the channel number.

`MATH` is the math waveform.

`REF<x>` is a reference waveform, where x is the reference channel number.

MEASUrement:MEAS<x>:STATE

Sets or returns whether the specified measurement slot is computed and displayed. The measurement slot is specified by <x>, which ranges from 1 through 4.

For a measurement to display, you must have selected a source waveform and defined the measurement you want to take and display. You select the measurement using the MEASUrement:MEAS<x>:SOURCE[1] command. You define the measurement type using the MEASUrement:MEAS<x>:TYPe command.

Group Measurement

Syntax

```
MEASUrement:MEAS<x>:STATE {OFF|ON|<NR1>}
MEASUrement:MEAS<x>:STATE?
```

Related Commands MEASUrement:MEAS<x>:SOURCE[1], MEASUrement:MEAS<x>:TYPe

Arguments

`OFF` disables calculation and display of the specified measurement slot.

`ON` enables calculation and display of the specified measurement slot.

`<NR1>` = 0 disables calculation and display of the specified measurement slot; any other value enables calculation and display of the specified measurement slot.

Examples `MEASUREMENT:MEAS2:STATE ON` computes and displays the measurement defined as measurement 2.

`MEASUREMENT:MEAS1:STATE?` might return `:MEASUREMENT:MEAS1:STATE 0` indicating that measurement defined for measurement slot 1 is disabled.

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

Returns the standard deviation of values accumulated for this measurement since the last statistical reset. Measurements are specified by <x>, the measurement slots, from 1 through 4.

Group Measurement

Syntax

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

Examples `MEASUREMENT:MEAS1:STDDEV?` might return `:MEASUREMENT:MEAS1:STDDEV 21.0E-12.`

MEASUrement:MEAS<x>:TYPe

Sets or returns the measurement type defined for the specified measurement slot. The measurement slot is specified by <x>, which ranges from 1 through 4.

Group Measurement

Syntax

```
MEASUrement:MEAS<x>:TYPe
{AMPlitude|AREa|BURst|CARea|CMEan|CRMs|DELay|FALL|FREQuency
|HIGH|LOW|MAXimum|MEAN|MINImum|NDUty|NEDGECount|NOVershoot
|NPULSECount|NWIdth|PDUty|PEDGECount|PERIod|PHAse|PK2Pk
|POVershoot|PPULSECount|PWIdth|RISe|RMS}
MEASUrement:MEAS<x>:TYPe?
```

Arguments `AMPlitude` measures the amplitude of the selected waveform. In other words, it measures the high value less the low value measured over the entire waveform or gated region. This measurement is available only on DPO models.

Amplitude = High – Low

`AREa` measures the voltage over time. The area is over the entire waveform or gated region and is measured in volt-seconds. The area measured above the ground is positive, while the area below ground is negative. This measurement is available only on DPO models.

`BURst` measures the duration of a burst. The measurement is made over the entire waveform or gated region.

`CAREa` (cycle area) measures the voltage over time. In other words, it measures, in volt-seconds, the area over the first cycle in the waveform or the first cycle in the gated region. The area measured above the common reference point is positive, while the area below the common reference point is negative. This measurement is available only on DPO models.

`CMEan` (cycle mean) measures the arithmetic mean over the first cycle in the waveform or the first cycle in the gated region. This measurement is available only on DPO models.

`CRMs` (cycle rms) measures the true Root Mean Square voltage over the first cycle in the waveform or the first cycle in the gated region. This measurement is available only on DPO models.

`DELay` measures the time between the middle reference (default = 50%) amplitude point of the source waveform and the destination waveform. This measurement is available only on DPO models.

`FALL` measures the time taken for the falling edge of the first pulse in the waveform or gated region to fall from a high reference value (default is 90%) to a low reference value (default is 10%). This measurement is available only on DPO models.

`FREQuency` measures the first cycle in the waveform or gated region. Frequency is the reciprocal of the period and is measured in hertz (Hz), where 1 Hz = 1 cycle per second.

`HIGH` measures the High reference (100% level, sometimes called Topline) of a waveform. This measurement is available only on DPO models.

`LOW` measures the Low reference (0% level, sometimes called Baseline) of a waveform. This measurement is available only on DPO models.

`MAXimum` finds the maximum amplitude. This value is the most positive peak voltage found. It is measured over the entire waveform or gated region. This measurement is available only on DPO models.

`MEAN` amplitude measurement finds the arithmetic mean over the entire waveform or gated region. This measurement is available only on DPO models.

`MINImum` finds the minimum amplitude. This value is typically the most negative peak voltage. It is measured over the entire waveform or gated region. This measurement is available only on DPO models.

`NDUty` (negative duty cycle) is the ratio of the negative pulse width to the signal period, expressed as a percentage. The duty cycle is measured on the first cycle in the waveform or gated region.

Negative Duty Cycle = ((Negative Width) / Period) × 100%

`NEDGECount` is the count of negative edges.

`NOVershoot` (negative overshoot) finds the negative overshoot value over the entire waveform or gated region. This measurement is available only on DPO models.

Negative Overshoot = ((Low – Minimum) / Amplitude) × 100%)

`NPULSECount` is the count of negative pulses.

`NWIdth` (negative width) measurement is the distance (time) between the middle reference (default = 50%) amplitude points of a negative pulse. The measurement is made on the first pulse in the waveform or gated region.

`PDUty` (positive duty cycle) is the ratio of the positive pulse width to the signal period, expressed as a percentage. It is measured on the first cycle in the waveform or gated region.

Positive Duty Cycle = ((Positive Width) / Period) × 100%

`PEDGECount` is the count of positive edges.

`PERIod` is the time required to complete the first cycle in a waveform or gated region. Period is the reciprocal of frequency and is measured in seconds.

`PHAse` measures the phase difference (amount of time a waveform leads or lags the reference waveform) between two waveforms. The measurement is made between the middle reference points of the two waveforms and is expressed in degrees, where 360° represents one waveform cycle.

`PK2Pk` (peak-to-peak) finds the absolute difference between the maximum and minimum amplitude in the entire waveform or gated region. This measurement is available only on DPO models.

`POVershoot` is the positive overshoot value over the entire waveform or gated region. This measurement is available only on DPO models.

Positive Overshoot = ((Maximum – High) / Amplitude) ×100%

`PPULSECount` is the count of positive pulses.

`PWIdth` (positive width) is the distance (time) between the middle reference (default = 50%) amplitude points of a positive pulse. The measurement is made on the first pulse in the waveform or gated region.

`RISe` timing measurement finds the rise time of the waveform. The rise time is the time it takes for the leading edge of the first pulse encountered to rise from a low reference value (default is 10%) to a high reference value (default is 90%). This measurement is available only on DPO models.

`RMS` amplitude measurement finds the true Root Mean Square voltage in the entire waveform or gated region. This measurement is available only on DPO models.

Examples

`MEASUREMENT:MEAS2:TYPE FREQUENCY` defines measurement 2 as a measurement of the frequency of a waveform.

`MEASUREMENT:MEAS1:TYPE?` might return `:MEASUREMENT:MEAS1:TYPE RMS` indicating that measurement 1 is defined to measure the RMS value of a waveform.

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

Returns the units associated with the specified measurement. The measurement slots are specified by <x>, which ranges from 1 through 4.

Group Measurement

Syntax

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

Related Commands MEASUrement:MEAS<x>:TYPe

Examples `MEASUREMENT:MEAS1:UNITS?` might return `:MEASUREMENT:MEAS1:UNIts %` indicating units for measurement 1 are set to percent.

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

Returns a calculate value for the measurement specified by <x>, which ranges from 1 through 4.

NOTE. *This is the same value as displayed on-screen. If measurement statistics are enabled, a new value is calculated with every waveform. In addition, this value is updated approximately every 1/3 second. If you are acquiring a long acquisition record, the oscilloscope may take longer to update.*

Group Measurement

Syntax

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

Related Commands MEASUrement:MEAS<x>:UNIts?, *ESR?, ALLEv?

Examples `MEASUREMENT:MEAS1:VALUE?` might return `:MEASUREMENT:MEAS1:VALue 2.8740E-06`. If the measurement has an error or warning associated with it, then an item is added to the error queue. The error can be checked for with the *ESR? and ALLEv? commands.

MEASUrement:METHod

Sets or returns the method used to calculate the 0% and 100% reference level.

Group Measurement

Syntax

```
MEASUrement:METHod {Auto|HIStogram|MINMax}
MEASUrement:METHod?
```

Related Commands MEASUrement:REFLevel:PERCent:HIGH, MEASUrement:REFLevel:PERCent:LOW, MEASUrement:REFLevel:PERCent:MID, MEASUrement:REFLevel:PERCent:MID2

Arguments

`Auto` selects the best method for each data set.

`HIStogram` sets the high and low waveform levels statistically using a histogram algorithm.

`MINMax` uses the highest and lowest values of the waveform record. This selection is best for examining waveforms with no large, flat portions of a common value, such as sine waves and triangle waves.

Examples `MEASUREMENT:METHOD?` might return `:MEASUREMENT:METHOD MINMAX` indicating that the reference levels are set to MIN and MAX.

MEASUrement:REFLevel? (Query Only)

Returns the current reference level parameters.

Group Measurement

Syntax

```
MEASUrement:REFLevel?
```

Examples `MEASUREMENT:REFLEVEL?` might return these reference level settings `:MEASUREMENT:REFLEVEL:METHOD PERCENT;ABSOLUTE:HIGH 0.0000;LOW 0.0000;MID1 0.0000 ;MID2 0.0000;:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90.0000;LOW 10.0000;MID1 50.0000 ;MID2 50.0000`

MEASUrement:REFLevel:ABSolute:HIGH

Sets or returns the high reference level, and is the upper reference level when MEASUrement:REFLevel:METHod is set to Absolute. This command affects the results of rise and fall measurements.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group Measurement

Syntax

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

Related Commands MEASUrement:REFLevel:METHod, MEASUrement:IMMed:TYPe, MEASUrement:MEAS<x>:TYPe

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

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

`MEASUREMENT:REFLEVEL:ABSOLUTE:HIGH?` might return `:MEASUREMENT:REFLEVEL:ABSOLUTE:HIGH 1.7100E+00` indicating that the absolute high reference level is set to 1.71 V.

MEASUrement:REFLevel:ABSolute:LOW

Sets or returns the low reference level, and is the lower reference level when MEASUrement:REFLevel:METHod is set to Absolute.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group Measurement

Syntax

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

Related Commands MEASUrement:REFLevel:METHod, MEASUrement:IMMed:TYPe, MEASUrement:MEAS<x>:TYPe

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

Examples `MEASUREMENT:REFLEVEL:ABSOLUTE:LOW 0.0` sets the low reference level to 0.0 V.

`MEASUREMENT:REFLEVEL:ABSOLUTE:LOW?` might return `:MEASUREMENT:REFLEVEL:ABSOLUTE:LOW 0.0000E+00` indicating that the absolute low reference level is set to 0.0 V.

MEASUrement:REFLevel:ABSolute:MID[1]

Sets or returns the mid reference level, and is the 50% reference level when MEASUrement:REFLevel:METHod is set to Absolute. This command affects the results of period, frequency, delay, and all cyclic measurements.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group Measurement

Syntax

```
MEASUrement:REFLevel:ABSolute:MID[1] <NR3>
MEASUrement:REFLevel:ABSolute:MID[1]?
```

Related Commands MEASUrement:REFLevel:METHod

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

Examples `MEASUREMENT:REFLEVEL:ABSOLUTE:MID 1 .71` sets the mid reference level to 0.71 V.

`MEASUREMENT:REFLEVEL:ABSOLUTE:MID?` might return `:MEASUREMENT:REFLEVEL:ABSOLUTE:MID 0.7100E+00` indicating that the absolute mid1 reference level is set to 0.71 V.

MEASUrement:REFLevel:ABSolute:MID2

Sets or returns the mid reference level for the "to" waveform when taking a delay measurement, and is the 50% reference level when MEASUrement:REFLevel:METHod is set to Absolute. This command affects the results of delay measurements.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group Measurement

Syntax

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

Related Commands MEASUrement:REFLevel:METHod

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

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

`MEASUREMENT:REFLEVEL:ABSOLUTE:MID2?` might return `:MEASUREMENT:REFLEVEL:ABSOLUTE:MID2 0.5000E+00` indicating that the absolute mid2 reference level is set to 0.5 V.

MEASUrement:REFLevel:ABSolute:MID<x>

Sets or returns the mid reference level for channel <x>, where x is the measurement channel.

Group Measurement

Syntax

```
MEASUrement:REFLevel:ABSolute:MID<x> <NR3>
MEASUrement:REFLevel:ABSolute:MID<x>?
```

Arguments `<NR3>` is the mid reference level in volts.

MEASUrement:REFLevel:METHod

Specifies or returns the reference level units used for measurement calculations.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the eight periodic measurements. To change the parameter for individual measurements, use the MEASUrement:MEAS<x>:REFLevel commands.*

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 example, 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?` might return `:MEASUREMENT:REFLEVEL:METHOD PERCENT` indicating that the reference level units used are calculated as a percent relative to HIGH and LOW.

MEASUrement:REFLevel:PERCent:HIGH

Sets or returns the percent (where 100% is equal to HIGH) used to calculate the high reference level when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of rise and fall measurements.

NOTE. *This command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group Measurement

Syntax

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

Related Commands

MEASUrement:REFLevel:METHod, MEASUrement:IMMed:TYPe, MEASUrement:MEAS<x>:TYPe

Arguments

`<NR3>` is the high reference level, ranging from 0 to 100%. The default high reference level is 90%.

Examples

`MEASUREMENT:REFLEVEL:PERCENT:HIGH 95` sets the high reference level to 95% of HIGH.

`MEASUREMENT:REFLEVEL:PERCENT:HIGH?` might return `:MEASUREMENT:REFLEVEL:PERCENT:HIGH 90` indicating that the percentage high reference level is set to 90% of HIGH.

MEASUrement:REFLevel:PERCent:LOW

Sets or returns the percent (where 100% is equal to HIGH) used to calculate the low reference level when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of rise and fall measurements.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group

Measurement

Syntax

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

Related Commands

MEASUrement:REFLevel:METHod, MEASUrement:IMMed:TYPe, MEASUrement:MEAS<x>:TYPe

Arguments

`<NR3>` is the low reference level, ranging from 0 to 100%. The default low reference level is 10%.

Examples

`MEASUREMENT:REFLEVEL:PERCENT:LOW 15` sets the high reference level to 15% of HIGH.

`MEASUREMENT:REFLEVEL:PERCENT:LOW?` might return `:MEASUREMENT:REFLEVEL:PERCENT:LOW 10` indicating that the percentage high reference level is set to 10% of HIGH.

MEASUrement:REFLevel:PERCent:MID[1]

Sets or returns the percent (where 100% is equal to HIGH) that is used to calculate the mid reference level when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of period, frequency, delay, and all cyclic measurements.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group Measurement

Syntax

```
MEASUrement:REFLevel:PERCent:MID[1] <NR3>
MEASUrement:REFLevel:PERCent:MID[1]?
```

Related Commands MEASUrement:REFLevel:METHod

Arguments `<NR3>` is the mid reference level, ranging from 0 to 100%. The default mid reference level is 50%.

Examples `MEASUREMENT:REFLEVEL:PERCENT:MID 1 60` sets the mid reference level to 60% of HIGH.

`MEASUREMENT:REFLEVEL:PERCENT:MID?` might return `:MEASUREMENT:REFLEVEL:PERCENT:MID 65` indicating that the percentage mid reference level is set to 65% of HIGH.

MEASUrement:REFLevel:PERCent:MID2

Sets or returns the percent (where 100% is equal to HIGH) that is used to calculate the mid reference level for the second waveform specified when MEASUrement:REFLevel:METHod is set to Percent. This command affects the results of delay measurements.

NOTE. *this command affects the associated reference level parameter for all MEASurements:IMMed and the four periodic measurements.*

Group Measurement

Syntax

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

Related Commands MEASUrement:REFLevel:METHod

Arguments `<NR3>` is the mid reference level, ranging from 0 to 100%. The default mid reference level is 50%.

Examples `MEASUREMENT:REFLEVEL:PERCENT:MID2 40` sets the mid2 reference level to 40% of HIGH.

`MEASUREMENT:REFLEVEL:PERCENT:MID2?` might return `:MEASUREMENT:REFLEVEL:PERCENT:MID2 45` indicating that the percentage mid2 reference level is set to 45% of HIGH.

MEASUrement:REFLevel:PERCent:MID<x>

Sets or returns the mid reference level for channel <x>, where x is the measurement channel.

Group Measurement

Syntax

```
MEASUrement:REFLevel:PERCent:MID<x> <NR3>
MEASUrement:REFLevel:PERCent:MID<x>?
```

Arguments `<NR3>` is the mid reference level in percent.

MEASUrement:SNAPShot (No Query Form)

Displays the measurement snapshot list on the oscilloscope screen. The list contains the immediate values for all available measurements of the active signal.

Group Measurement

Syntax

```
MEASUrement:SNAPShot
```

MESSage

This command sets or queries message parameters.

Group Miscellaneous

Syntax

```
MESSage
MESSage?
```

Examples `MESSAGE?` might return `MESSAGE:SHOW "TP401";BOX 271,82,292,114;STATE 0` indicating that the message parameters.

MESSage:BOX

Sets or returns the size and position of the message window. This command does not display the message unless MESSage:STATE is on.

X1 and Y1 are the screen coordinates of the top left corner of the message box. X2 and Y2 are the screen coordinates of the bottom right corner of the message box. All four coordinates are returned by the query.

Changing the text in the message box, using the MESSAGE:SHOW command, automatically resizes the message box. If you want a custom message box size, send the MESSAGE:BOX command after changing the text using the MESSAGE:SHOW command.

Message box settings and data are saved and restored in saved setups.

Group Display

Syntax

```
MESSage:BOX <X1>,<Y1>[,<X2>,<Y2>]
MESSage:BOX?
```

Related Commands MESSage:STATE, MESSage:SHOW, MESSage:CLEAR

Arguments `<X1>` and `<X2>` = 0 to 1023, 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 767, 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 16 pixels so the window must be at least that high to fully display characters. `<X2>` and `<Y2>` are optional because the `MESSAGE:SHOW`

command automatically sizes the box to fit the message. All four values are returned in a query.

MESSage:CLEAR (No Query Form)

Removes the message text from the message window.

Group Display

Syntax

```
MESSage:CLEAR
```

Related Commands MESSage:BOX, MESSage:SHOW, MESSage:STATE

Examples MESSage:CLEAR
removes the message from the message window.

MESSage:SHOW

Clears the contents of the message window and displays the new message in the window. Changing the string length causes the message box to automatically resize to fit the text. The box may be resized using the MESSage:BOX command.

Group Display

Syntax

```
MESSage:SHOW <QString>
MESSage:SHOW?
```

Related Commands MESSage:BOX, MESSage:CLEAR, MESSage:STATE

Arguments `<QString>` is the message and can include any of the characters shown in the Character Set, Appendix A. The maximum length of the message is 1000 characters; the instrument ignores longer strings.

The message box size is set to fit the message. You can also set the message area height and width using the MESSage:BOX command. The length of the message that fits in the message area depends on the contents of the message because the width of characters varies.

If the message exceeds the limits of the message box, 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 message box.

The message is left-justified, and is displayed on a single line starting with the top most line in the window. A new line 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. Text which does not fit within the message box is truncated. Defining a message box text string erases any previously displayed text within the message box.

You can send a tab by transmitting a tab character (\t or \x09) followed characters representing the most significant eight bits followed by significant eight bits of a 16-bit number. The number specifies the position relative to the left margin of the message area. For example, to tab send TAB (\t or \x09), NUL (decimal 0), and CR (decimal 13).

For example, using hexadecimal escape sequences, MESSAGE:SHOW '\x09\x01\x17Hello' when sent as a command would cause the 'Hello' to be displayed starting at pixel position 279 relative to the left margin set by the MESSAGE:BOX command. If you want to display characters starting at position 279, then 279 = 0x0117; split the hexadecimal number into two characters 0x01 and 0x17 and send \x09\x01\x17.

Special characters which control decoration are two character sequences where the first character is an escape (0x1b) and the second character is as described below.

Bit 7	0	
Bit 6	If set, inverse video is toggled from current state and the following text is displayed in the new inverse state until the state is toggled again. Remaining bits are ignored	
Bit 5	If set, the color index in the four LSB's (bits 0 through 3) is applied to the foreground or background color depending on the fg/bg bit (bit 4).	
Bit 4	If set, color change is applied to the background, otherwise applies to the foreground.	
Bit 0 – 3	Specifies the color index (0 through 15) to change color as specified below:	
	Index 0	Black (background)
	Index 1	Yellow (Ch 1)
	Index 2	Cyan (Ch 2)
	Index 3	Magenta (Ch 3)
	Index 4	Green (Ch 4)
	Index 5	Red (math)
	Index 6	White (reference)
	Index 7	Orange
	Index 8	Gray (Graticule)
	Index 9	White (text)

	Index 10	Tek blue
	Index 11	Bright blue
	Index 12	Undefined
	Index 13	Blue
	Index 14	Undefined
	Index 15	Dark blue
Bit 4	If set, the foreground color is set to the default foreground color.	
Bit 3	If set, the background color is set to the default background color.	
Bit 2	Undefined	
Bit 1	Undefined	
Bit 0	Undefined	

The ESC (escape) character followed by the @ character turns inverse video on or off and can be embedded in the message string. Example: “abcESC@defESC@ ghi” specifies the string “abcdefghi” where the “def” portion is displayed in inverse video.

Example: “abcESC#defESC)ESC@ghi” specifies the string “abcdefghi” where the “def” portion appears in the channel 3 color (magenta) and the “ghi” portion appears in the normal text color except it’s in inverse video.

An alternate way to enter characters is octal escape sequences. This consists of a backslash followed by numerals in the standard C language printf fashion.

Another way to enter characters is \xnn where the nn is the hexadecimal value of the character to display.

An advantage of these methods is that any controller program can be used. Another advantage is it’s easy to access characters with the high bit set, that is, those characters with a decimal value greater than 127.

An alternate way to enter certain characters is with a backslash followed by a single character (following “standard” Unix) as described in the table below.

n	Newline (carriage return and line feed)
\	Backslash (\\ is required to get a backslash character)
t	Horizontal tab; the next 2 characters specify the pixel column to tab to as explained earlier

If a backslash is followed by an undefined special character, the backslash is ignored and the character following it is accepted as is.

NOTE. *The use of any escape codes other than those described above may produce unpredictable results.*

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 controller program.

MESSage:STATE

Controls the display of the message window on the screen.

Group

Display

Syntax

```
MESSage:STATE {OFF|ON|<NR1>}
MESSage:STATE?
```

Related Commands

MESSage:BOX

MESSage:SHOW, MESSage:CLEAR

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.

NEWpass (No Query Form)

This command 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

Syntax

```
NEWpass <QString>
```

Related Commands

*PUD

PASSWord

Arguments `<QString>` is the new password, which can contain up to 16 characters.

Examples `NEWPASS "mypassword"` creates a new password (mypassword) for accessing your protected data.

*OPC

Generates the operation complete message in the Standard Event Status Register (SESR) when all pending commands that generate an OPC message are complete. The *OPC? query places the ASCII character "1" into the output queue when all such OPC commands are complete. The *OPC? response is not available to read until all pending operations finish. (See page 3-1, *Status and Events*.)

The *OPC command allows you to synchronize the operation of the oscilloscope with your application program. (See page 3-7, *Synchronization Methods*.)

Table 2-40: Commands that Generate an OPC Message

Operation	Command
Single sequence acquisition	ACQuire:STATE {ON|NR 1}
	AUXin:PRObe:DEGAUss EXECute
	CH<x>:PRObe:DEGAUss EXECute
	DIAg:STATE EXECute
	RECAll:SETUp <file path>
	RECAll:WAVEform <file path>,REF<x>
	SAVe:IMAGe <file path>
	SAVe:SETUp <file path>
	SAVe:WAVEform <wfm>, {REF<x>}
	TEKSecure
Hard copy operation	HARDCopy START
Calibration step	CALibrate:FACtory {STARt|PREVious|CONTinue}

Group Status and Error

Syntax
```
*OPC
*OPC?
```

Related Commands BUSY?, *WAI

Examples `*OPC` generates the operation complete message in the SESR at the completion of all pending OPC operations.

*OPC? might return 1 to indicate that all pending OPC operations are finished.

PASSWord(No Query Form)

Enables the *PUD and NEWpass set commands. Sending PASSWord without any arguments disables these same commands. Once the password is successfully entered, the *PUD and NEWpass commands are enabled until the oscilloscope is powered off, or until the FACtory command or the PASSWord command with no arguments is issued.

To change the password, you must first enter the valid password with the PASSWord command and then change to your new password with the NEWpass command. Remember that the password is case sensitive.

Group Miscellaneous

Syntax

```
PASSWord <QString>
```

Related Commands NEWpass, *PUD

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.

PICTBridge:DATEPrint

Enables or disables printing the date on the print output.

Group PictBridge

Syntax

```
PICTBridge:DATEPrint {DEFLT|OFF|ON}
```

Arguments DEFLT is the default setting of the printer.

ON enables the date print on the print output.

OFF disables the date print on the print output.

Examples

`PICTBRIDGE:DATEPRINT?` might return `PICTBRIDGE:DATEPRINT DEFLT` indicating that the default setting of the printer.

`PICTBRIDGE:DATEPRINT ON` enables the date print on the print output.

PICTBridge:DEFault (No Query Form)

Sets the arguments for all PictBridge commands to their default values. The default values are the same as printer default settings.

Group

PictBridge

Syntax

```
PICTBridge:DEFault
```

Examples

`PICTBRIDGE:DEFAULT` sets all the PictBridge commands to their default values.

PICTBridge:IDPrint

Enables or disables printing the oscilloscope model and serial number on the print output.

Group

PictBridge

Syntax

```
PICTBridge:IDPrint {DEFLT|OFF|ON}
PICTBridge:IDPrint?
```

Related Commands

Arguments

`OFF` disables the oscilloscope model and serial number print on the print output.

`ON` enables the oscilloscope model and serial number print on the print output.

`Default` is provided for compatibility with TDS instruments that support pictbridge.

Examples

`PICTBRIDGE:IDPRINT?` might return `PICTBRIDGE:IDPRINT 0` indicating that the ID print is disabled on the print output.

`PICTBRIDGE:IDPRINT ON` enables the printing of oscilloscope model and serial number on the print output.

PICTBridge:IMAGESize

Sets or returns the image print size.

Group PictBridge

Syntax

```
PICTBridge:IMAGESize {DEFLT|IN2P5BY3P25|L|IN4BY6|L2|
IN8BY10|L4|E|CARD|HAGAKIPC|CM6BY8|CM7BY10|CM9BY13|
CM10BY15|CM13BY18|CM15BY21|CM18BY24| A4|LETTER}
```

Arguments

`DEFLT` for a default image print size.

`IN2P5BY3P25` for a 2_5*3_25 image print size.

`L` for a 3_5*5 image print size.

`IN4BY6` for a 4*6 image print size.

`L2` for a 5*7 image print size.

`IN8BY10` for a 8*10 image print size.

`L4` for a 254 MM*178 MM image print size.

`E` for a 110 MM*74 MM image print size.

`CARD` for a 89 MM*55MM image print size.

`HAGAKIPC` for a 100 MM*148 MM image print size.

`CM6BY8` for a 6 CM*8 CM image print size.

`CM7BY10` for a 7 CM*10 CM image print size.

`CM9BY13` for a 9 CM*13 CM image print size.

`CM10BY15` for a 10 CM*15 CM image print size.

`CM13BY18` or a 13 CM*18 CM image print size.

`CM15BY21` for a 15 CM*21 CM image print size.

`CM18BY24` for a 18 CM*24 CM image print size.

`A4` for a A4 image print size.

`LETTER` for a Letter image print size.

Examples

`PICTBRIDGE:IMAGESIZE`? might return `PICTBRIDGE:IMAGESIZE DEFLT` indicating that the default image print size.

`PICTBRIDGE:IMAGESIZE CARD` sets the image print size to 89 MM* 55 MM.

PICTBridge:PAPERSize

Sets the output print paper size.

Group PictBridge

Syntax

```
PICTBridge:PAPERSize
{DEFLT|L|L2|HAGAKIPCARD|MM54BY86|MM100BY150
|IN4BY6|IN8BY10|LETTER|IN11BY17|A0|A1|A2|A3|A4|A5|A6|A7|A8|A9|
B0|B1|B2|B3|
B4|B5|B6|B7|B8|B9|ROLL89MM|ROLL127MM|ROLL100MM|ROLL210MM}
```

Arguments

`DEFLT` for a default paper size.

`L` for a paper size L

`L2` for a paper size 2L.

`HAGAKIPCARD` for a paper size Hagaki.

`MM54BY86` for a card paper size.

`MM100BY150` for paper size of 100*150 MM.

`IN4BY6` for a paper size of 4*6.

`IN8BY10` for a paper size of 8*10.

`LETTER` for a letter paper size.

`IN11BY17` for a paper size of 11*17.

`A0` for a A0 paper size.

`A1` for a A1 paper size.

`A2` for a A2 paper size.

`A3` for a A3 paper size.

`A4` for a A4 paper size.

`A5` for a A5 paper size.

`A6` for a A6 paper size.

`A7` for a A7 paper size.

`A8` for a A8 paper size.

`A9` for a A9 paper size.

`B0` for a B0 paper size.

`B1` for a B1 paper size.

`B2` for a B2 paper size.

`B3` for a B3 paper size.

`B4` for a B4 paper size.

`B5` for a B5 paper size.

`B6` for a B6 paper size.

`B7` for a B7 paper size.

`B8` for a B8 paper size.

`B9` for a B9 paper size.

`ROLL89MM` for a 89 MM Roll paper size.

`ROLL127MM` for a 127 MM Roll paper size.

`ROLL100MM` for a 100 MM Roll paper size.

`ROLL210MM` for a 210 MM Roll paper size.

Examples

`PICTBRIDGE:PAPERSIZEL` sets the paper size to L.

`PICTBRIDGE:PAPERSIZE?` might return `PICTBRIDGE:PAPERSIZE DEFLT` indicating that the paper size is set to Default.

PICTBridge:PAPERType

Sets or returns the paper type.

Group

PictBridge

Syntax

`PICTBridge:PAPERType {DEFLT|PLAIN|PHOTO|FASTPHOTO}`

Arguments

`DEFLT` for a default print paper type.

`PLAIN` for a plain print paper type.

`PHOTO` for a photo print paper type.

`FASTPHOTO` for a fastphoto print paper type.

Examples

`PICTBRIDGE:PAPERTYPE?` might return `PICTBRIDGE:PAPERTYPE DEFLT` indicating that the default paper type.

`PICTBRIDGE:PAPERTYPE PLAIN` sets the paper type to Plain.

PICTBridge:PRINTQual

Sets or returns the output print quality.

Group PictBridge

Syntax

```
PICTBridge:PRINTQual {DEFLT|NRMAL|FINE|DRAFT}
```

Arguments

`DEFLT` for the default quality print.

`NRMAL` for a normal quality print.

`FINE` for a fine quality print.

`DRAFT` for a draft quality print.

Examples

`PICTBRIDGE:PRINTQUAL?` might return `PICTBRIDGE:PRINTQUAL DEFLT` indicating that the default quality print.

`PICTBRIDGE:PRINTQUAL FINE` sets the print quality to Fine.

*PSC

Sets or returns the power-on status flag that controls the automatic power-on handling of the DESER, SRER, and ESER registers. When *PSC is true, the DESER register is set to 255 and the SRER and ESER registers are set to 0 at power-on. When *PSC is false, the current values in the DESER, SRER, and ESER registers are preserved in nonvolatile memory when power is shut off and are restored at power-on.

Group Status and Error

Syntax

```
*PSC {OFF|ON|NR1>}
*PSC?
```

Related Commands DESE, *ESE, FACtory, *RST, *SRE

Arguments

`OFF` sets the power-on status clear flag to false.

`ON` sets the power-on status clear flag to true.

`<NR1>` = 0 sets the power-on status clear flag to false. This disables the power-on clear allowing the oscilloscope to possibly assert SRQ after power-on; any other

value sets the power-on status clear flag to true, enabling the power-on status clear preventing any SRQ assertion after power on.

Examples

`*PSC 0` sets the power-on status clear flag to false.

`*PSC?` might return 1 to indicate that the power-on status clear flag is set to true.

*PUD

Sets or returns a string of Protected User Data. This data is protected by the PASSWord command. You can modify it only by first entering the correct password. This password is not necessary to query the data.

Group

Status and Error

Syntax

```
*PUD {<Block>|<QString>}
*PUD?
```

Related Commands

PASSWord

Arguments

`<Block>` is a block containing up to 300 ASCII characters.

`<QString>` is a string containing up to 300 ASCII characters.

Examples

`*PUD #229This oscilloscope belongs to me` stores the string "This oscilloscope belongs to me" in the user protected data area.

`*PUD?` might return `#221PROPERTY OF COMPANY X`

*RCL (No Query Form)

This command restores the state of the oscilloscope from a copy of the settings stored in memory (The settings are stored using the *SAV command).

Group

Save and Recall

Syntax

```
*RCL <NR1>
```

Related Commands

FACtory, *LRN?, RECAll:SETUp, *RST, *SAV, SAVe:SETUp

Arguments

`<NR1>` is a value in the range from 1 to 10, which specifies a saved setup storage location.

Examples

`*RCL 3` restores the oscilloscope from a copy of the settings stored in memory location 3.

RECAll:SETUp (No Query Form)

Restores the state of the oscilloscope from a copy of the settings stored in memory. The settings are stored using the `*SAV` command.

Group

Save and Recall

Syntax

```
RECAll:SETUp {FACtory|<NR1>|<file path>}
```

Related Commands

FACtory, *RCL, *RST, *SAV, SAVe:SETUp, FILESystem:CWD

Arguments

`FACtory` restores the factory setup.

`<NR1>` is a value in the range from 1 to 10, which specifies a saved setup storage location.

`<file path>` specifies a location for an oscilloscope setup file. <file path> is a quoted string that defines the file name and path. Input the file path using the form `<drive>:/<dir>/<filename>.<extension>` and one or <dir>s are optional. If you do not specify them, the oscilloscope will read the file from the default directory (see FILESystem:CWD). <filename> stands for a filename; the use of wildcard characters in filenames is not supported. Filename extensions are not required, but highly recommended.

Examples

`RECALL:SETUP FACTORY` recalls (and makes current) the oscilloscope setup to its factory defaults.

`RECALL:SETUP 2` recalls the oscilloscope setup from setup storage location 2.

`RECALL:SETUP "TEK00000.SET"` recalls the setup from the file TEK00000.SET in the default directory for setups (E:/tekScope/setups).

RECAll:WAVEform (No Query Form)

This command (no query form) recalls a stored waveform to a reference location.

Group Save and Recall

Syntax `RECAll:WAVEform <file path>,REF<x>`

Related Commands SAVe:WAVEform, FILESystem:CWD, FILESystem

Arguments `REF<x>` specifies a location in internal reference memory. Reference memory location values range from 1 through 4.

`<file path>` specifies a location for an oscilloscope setup file. <file path> is a quoted string that defines the file name and path. Input the file path using the form `<drive>:/<dir>/<filename>.<extension>` and one or <dir>s are optional. If you do not specify them, the oscilloscope will read the file from the default directory (see FILESystem:CWD). <filename> stands for a filename; the use of wildcard characters in filenames is not supported. Filename extensions are not required, but highly recommended.

Examples `RECALL:WAVEFORM "tek00000.isf",REF1` recalls the waveform stored in the file named tek00000.isf from the current directory for waveforms to the reference location 1.

REF<x>? (Query Only)

Returns reference waveform data for the channel specified by <x>, where x is the reference channel number.

Group Vertical

Syntax `REF<x>?`

REF<x>:DATE? (Query Only)

Returns the date that reference waveform data for channel <x> was copied into the internal reference memory, where x is the reference channel number.

Group Vertical

Syntax `REF<x>:DATE?`

REF<x>:HORizontal:DELay:TIMe

Sets or returns the horizontal delay time for reference waveform <x>, where x is the reference channel number. The delay time is expressed in seconds and is limited to ±5 times the reference horizontal scale.

Group Vertical

Syntax

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

Arguments `<NR3>` is the delay time in seconds.

Examples `REF2:HORIZONTAL:DELAY:TIME 4.0E-6` sets the horizontal delay time for the REF2 waveform to 4 µs.

REF<x>:HORizontal:SCAle

Sets or returns the horizontal scale for reference waveform <x>, where x is the reference channel number.

Group Vertical

Syntax

```
REF<x>:HORizontal:SCAle <NR3>
REF<x>:HORizontal:SCAle?
```

Arguments `<NR3>` is the horizontal scale in seconds.

Examples `REF1:HORIZONTAL:SCALE?` might return `REF1:HORIZONTAL:SCALE 4.0E-4`.

REF<x>:LABel

Sets or returns the reference waveform label for the channel specified by <x>, where x is the reference channel number.

Group Vertical

Syntax

```
REF<x>:LABel <Qstring>
REF<x>:LABel?
```

Arguments

`<Qstring>` is an alpha-numeric string of text, enclosed in quotes, that contains the label text for the reference channel <x> waveform. The text string is limited to 30 characters.

Examples

`REF2:LABEL?` might return `:REF2:LABEL "Clk Wfm 2"`.

REF<x>:POSition

Sets or returns the vertical position for channel <x>, where x is the reference channel number.

Group

Vertical

Syntax

```
REF<x>:POSition <NR3>
REF<x>:POSition?
```

Arguments

`<NR3>` is the vertical position in volts.

REF<x>:SCAle

Sets or returns the vertical scale for the channel specified by <x>, where x is the reference channel number.

Group

Save and Recall

Syntax

```
REF<x>:SCAle <NR3>
REF<x>:SCAle?
```

Arguments

`NR3` is the vertical scale in volts

REF<x>:TIMe? (Query Only)

Returns the time that reference waveform data was copied into the internal reference memory for reference channel <x>, where x is the reference channel number.

Group Vertical

Syntax `REF<x>:TIMe?`

Examples `REF2:TIME?` might return "`16:54:05`".

REF<x>:VERTical:POSition

Sets or returns the vertical position of the reference waveform specified by <x>, where x is the reference channel number.

Increasing the position value of a waveform causes the waveform to move up, and decreasing the position value causes the waveform to move down. Position adjusts only the display position of a waveform. The position value determines the vertical graticule coordinate at which signal values are displayed. For example, if the position for Reference 3 is set to 2.0, the signal represented by that reference will be displayed at 2.0 divisions above the center of the screen.

Group Vertical

Syntax

```
REF<x>:VERTical:POSition <NR3>
REF<x>:VERTical:POSition?
```

Related Commands CH<x>:POSition, MATH[1]:VERTical:POSition

Arguments `<NR3>` is the desired position, in divisions from the center horizontal graticule. The range is from –5.0 to 5.0 divisions.

Examples `REF2:VERTICAL:POSITION 1.3E+00` positions the Reference 2 input signal 1.3 divisions above the center horizontal graticule.

`REF1:VERTICAL:POSITION?` might return `:REF1:VERTICAL:POSITION -1.3000E+00` indicating that the current position of Reference 1 is 1.3 divisions below the center horizontal graticule.

REF<x>:VERTical:SCAle

Sets or returns the vertical scale for the reference waveform specified by <x>, where x is the reference channel number.

Each waveform has a vertical scale parameter. For a signal with constant amplitude, increasing the Scale causes the waveform to be displayed smaller. Decreasing the scale causes the waveform to be displayed larger.

Scale affects all waveforms, but affects reference and math waveforms differently from channel waveforms:

- For reference and math waveforms, this setting controls the display only, graphically scaling these waveforms and having no affect on the acquisition hardware.
- For channel waveforms, this setting controls the vertical size of the acquisition window as well as the display scale. The range and resolution of scale values depends on the probe attached and any other external factors you have specified.

Group Vertical

Syntax

```
REF<x>:VERTical:SCAle <NR3>
REF<x>:VERTical:SCAle?
```

Related Commands CH<x>:SCAle, MATH[1]:VERTical:SCAle

Arguments `<NR3>` is the gain in user units-per-division.

Examples `REF2:VERTICAL:SCALE 100E-03` sets the Reference 2 scale to 100 mV per division.

`REF2:VERTICAL:SCALE?` might return `:REF2:VERTICAL:SCALE 1.0000e+00` indicating that the current vertical scale setting for Reference 2 is 1 V per division.

REM (No Query Form)

Embeds a comment within programs as a means of internally documenting the programs. The oscilloscope ignores these embedded comment lines.

Group Miscellaneous

Syntax

```
REM <QString>
```

Arguments `<QString>` is a string that can contain a maximum of 80 characters.

Examples `REM "This is a comment"` is a comment string that the oscilloscope will ignore.

*RST (No Query Form)

Resets the oscilloscope to the factory default settings. The *RST command does not alter the following:

- Calibration data that affect device specifications
- The Output Queue
- The Service Request Enable Register setting
- The Power-on status clear flag setting
- Alias definitions
- Stored settings
- The *PUD? Response
- Any of the values associated with the DATA command.
- Oscilloscope password

Group Status and Error

Syntax `*RST`

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

Arguments None

Examples `*RST` resets the oscilloscope settings to factory defaults.

*SAV (No Query Form)

Stores the state of the oscilloscope to a specified memory location. You can use the *RCL command to restore the oscilloscope to this saved state at a later time.

Group Save and Recall

Syntax `*SAV <NR1>`

Related Commands *RCL, RECAll:SETUp, SAVe:SETUp

Arguments `<NR1>` specifies a location in which to save the state of the oscilloscope. Location values range from 1 through 10. Using an out-of-range location value causes an execution error. Any settings that have been stored previously at this location will be overwritten.

Examples `*SAV 2` saves the current oscilloscope state in memory location 2.

SAVe:ASSIgn:TYPe

Sets or returns the assignment of the data to be saved when the front-panel Save button is pressed.

Group Save and Recall

Syntax

```
SAVe:ASSIgn:TYPe {IMAGe|WAVEform|SETUp}
SAVe:ASSIgn:TYPe?
```

Arguments `IMAGe` assigns the Save button to save screen images.

`WAVEform` assigns the Save button to save waveforms.

`SETUp` assigns the Save button to save setups.

SAVe:EVENTtable:BUS<x> (No Query Form)

Saves the data from bus<x> to a specified file and location; where x is the bus number

Group Save and Recall

Syntax

```
SAVe:EVENTtable:BUS<x> <file path>
```

Arguments `<file path>` is a quoted string that defines the file name and path location where the event table will be stored.

NOTE. *<filename> stands for a filename of up to 125 characters, followed by a period (".") and the three-character extension. Waveform files should have a .csv extension for comma-separated spreadsheet format files.*

SAVe:IMAGe (No Query Form)

Saves a capture of the screen image into the specified file. Supported image formats are PNG, Windows Bitmap, and TIFF. If an extension for a supported file type is added to the file name, then the corresponding format will be used. If no supported extension is added to the file, the format to use will be determined by the value obtained from the :SAVe:IMAGe:FILEFormat? query.

Group Save and Recall

Syntax

```
SAVe:IMAGe <file path>
```

Related Commands SAVe:ASSIgn:TYPe

Arguments `<file path>` is a filename, including path, where the image will be saved. If you do not specify a directory, the oscilloscope will store the file in the current working directory. File name extensions are not required but are highly recommended. The images will be saved in E:/.

SAVe:IMAGe:FILEFormat

Sets or returns the file format to use for saving screen images when the file type cannot be determined from the given file name or when screen images are captured by using the front panel.

Group Save and Recall

Syntax

```
SAVe:IMAGe:FILEFormat {PNG|BMP|TIFf}
SAVe:IMAGe:FILEFormat?
```

Related Commands SAVe:IMAGe

Arguments `PNG` saves the file in Portable Network Graphics format.

`BMP` saves the file in Microsoft Windows bitmap format.

`TIFf` saves the file in Tagged Image File Format.

SAVe:IMAGe:LAYout

Sets or returns the layout to use for saved screen images.

Group Save and Recall

Syntax

```
SAVe:IMAGe:LAYout {LANdscape|PORTRait}
SAVe:IMAGe:LAYout?
```

Arguments

`LANdscape` specifies that screen images are saved in landscape format.

`PORTRait` specifies that screen images are saved in portrait format.

SAVe:SETUp (No Query Form)

Stores the state of the oscilloscope to a specified memory location. You can later use the *RCL command to restore the oscilloscope to this saved state.

Group Save and Recall

Syntax

```
SAVe:SETUp {<file path>|<NR1>}
```

Related Commands *RCL, RECAll:SETUp, *SAV

Arguments

`<file path>` is the target location for storing the setup file. <file path> is a quoted string that defines the file name and path. Input the file path using the form <drive>:<dir>/<filename>. <extension> and one or <dir>s are optional. If you do not specify them, the oscilloscope will store the file in the current working directory. <filename> stands for a filename. (Use of wildcard characters in filenames is not supported.) Filename extensions are not required but are highly recommended. For setups, use the extension ".SET".

`<NR1>` specifies a location for saving the current front-panel setup. The front-panel setup value ranges from 1 to 10. Using an out-of-range value causes an execution error. Any settings that have been stored previously at this location will be overwritten.

Examples

`SAVE:SETUP 5` saves the current oscilloscope setup in memory location 5.

`SAVE:SETUP "TEK00000.SET"` saves the current oscilloscope setup in the file TEK00000.SET in the current working directory.

SAVe:WAVEform (No Query Form)

This command saves a specified waveform or all displayed waveforms (excluding serial bus waveforms). Only individual analog waveforms (CH<x>, MATH and REF<x>) can be saved to reference memory locations.

You can save all displayed waveforms, excluding serial bus waveforms, to a single CSV file when the `SAVE:WAVEFORM:FILEFORMAT` is set to `SPREADSHEET`.

You can save all displayed waveforms, excluding serial bus waveforms to consecutive ISF (internal save format) files when the `SAVE:WAVEFORM:FILEFORMAT` is set to `INTERNAL`.

NOTE. *For MSO models, reduced resolution waveform data is not available for the digital channels.*

Group Save and Recall

Syntax

```
SAVe:WAVEform[<wfm>,{REF<x>}] | [<wfm>, <QString>] | [ALL,
<QString>]
```

Related Commands RECAll:WAVEform, SAVe:WAVEform:FILEFormat

Arguments

`<wfm>, <REF<x>>` saves the specified waveform to the specified reference memory location. <wfm> can be any live analog channel (where <x> is the channel number), the MATH1 waveform, or another reference waveform (such as REF1).

`<wfm>, <QString>` saves the specified waveform to the file specified in the quoted string argument. Any live channel (such as CH1), the MATH1 waveform, any reference waveform can be saved to a file.

`ALL, <QString>` saves all displayed waveforms, excluding serial bus waveforms, to a single CSV file specified by the quoted string argument when the `SAVE:WAVEFORM:FILEFORMAT` is set to `SPREADSHEET`, or saves all displayed waveforms, excluding serial bus waveforms to individual ISF (internal save format) files with a file name prefix specified by the argument with an underscore (_) and the waveform ID (such as CH1, REF1, MATH) appended to the file name(s).

Examples `SAVE:WAVEFORM CH1,REF1` saves the CH1 waveform in reference memory location 1.

`:SAVE:WAVEFORM:FILEFORMAT SPREADSHEET; :SAVE:WAVEFORM ALL, "E:/test_folder/test1_all.csv"` saves all displayed waveforms (excluding serial bus waveforms) to `E:/test_folder/test1_all.csv`.

`:SAVe:WAVEform:FILEFormat INTERNal; :SAVe:WAVEform ALL, "E:/test_folder/test1"` saves all displayed waveforms (excluding serial bus waveforms) to individual files named `E:/test_folder/test1_<wfm>.isf` (for example `test1_CH1.isf`).

SAVe:WAVEform:FILEFormat

Specifies or returns the file format for saved waveforms. Waveform header and timing information is included in the resulting file of non-internal formats. The oscilloscope saves DPO waveforms as a 500 x 200 matrix, with the first row corresponding to the most recently acquired data. The values specified by DATa:STARt and DATa:STOP determine the range of waveform data to output. In the event that DATa:STOP value is greater than the current record length, the current record length determines the last output value.

Group

Save and Recall

Syntax

```
SAVe:WAVEform:FILEFormat {INTERNal|SPREADSheet}
SAVe:WAVEform:FILEFormat?
```

Related Commands

CURVe, DATa, DATa:STARt, DATa:STOP, SAVe:WAVEform, WFMInpre:NR_Pt, WFMOutpre:NR_Pt?

Arguments

`INTERNal` specifies that waveforms are saved in an internal format, using a .isf filename extension. These files can be recalled as reference waveforms. When this argument is specified, the settings specified via the DATa:STARt and DATa:STOP commands have no meaning as the entire waveform is saved.

`SPREADSheet` specifies that waveform data is saved in a format that contains comma delimited values. These waveform data files are named using the .csv filename extension. Saving waveforms in CSV format enables spreadsheet programs to import the data.

Examples

`SAVE:WAVEFORM:FILEFORMAT INTERNAL` specifies that the internal file format is the format used for saving waveforms.

`SAVE:WAVEFORM:FILEFORMAT?` might return `:SAVE:WAVEFORM:FILEFORMAT INTERNAL` indicating that waveforms are saved using the internal format.

SAVe:WAVEform:GATIng

Specifies whether save waveform operations should save the entire waveform (NONe) or a specified portion of the waveform.

Group Save and Recall

Syntax

```
SAVe:WAVEform:GATIng {NONe|CURSors|SCREEN}
SAVe:WAVEform:GATIng?
```

Arguments

`CURSors` turns on cursors and the gates are the waveform record points at the cursor positions.

`NONe` saves the entire waveform.

`SCREEN`, if zoom is on, the gates are the start and end waveform record points of the zoom (upper) graticule, otherwise the gates are the start and end waveform record points of the main graticule.

Examples `SAVE:WAVEFORM:GATING CURSors` specifies that, when the waveform gating is set to cursors, save waveform operations should save the waveform points between the cursors. If cursors are turned off, waveform gating automatically reverts to NONE.

SAVe:WAVEform:SPREADSheet:RESOlution

Specifies whether to save the full (LRL) or reduced (thumb) resolution waveform to a CSV file.

Group Waveform Enhanced

Syntax

```
SAVe:WAVEform:SPREADSheet:RESOlution {FULL|REDUced}
SAVe:WAVEform:SPREADSheet:RESOlution?
```

Arguments

`FULL` specifies that full resolution waveform is saved to a CSV file.

`REDUced` specifies that reduced resolution waveform is saved to a CSV file.

Examples `SAVE:WAVEFORM:SPREADSHEET:RESOLUTION?` might return `SAVE:WAVEFORM:SPREADSHEET:RESOLUTIONFULL` indicating that full resolution waveform is saved to a CSV file.

SAVE:WAVEFORM:SPREADSHEET:RESOLUTIONREDUced sets the reduced resolution waveform to be saved to a CSV file.

SEARCH? (Query Only)

Returns all search-related settings.

Group Search

Syntax SEARCH?

Examples SEARCH? might return:

```
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:CONDITION SS;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:CONDITION SS;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:DATA:MOSI:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:DATA:MOSI:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:DATA:MISO:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:DATA:MISO:VALUE
“XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:DATA:SIZE1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:CONDITION TXSTART;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:CONDITION TXSTART;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:RX:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:RX:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C: RX:DATA:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:RX:DATA:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:TX:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:TX:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:TX:DATA:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:TX:DATA:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:CONDITION START;
:SEARCH:SEARCH1:T RIGGER:A:BUS:B2:I2C:CONDITION START;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:DATA:SIZE 1;
```

```
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:ADDRESS:MODE ADDR7;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:ADDRESS:MODE ADDR7;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:ADDRESS:TYPE USER;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:ADDRESS:TYPE USER;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:ADDRESS:VALUE
"XXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:ADDRESS:VALUE
"XXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:CONDITION SOF;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:CONDITION SOF;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:FRAMETYPE DATA;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:FRAMETYPE DATA;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A: BUS:B1:CAN:DATA:QUALIFIER EQUAL;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:DATA:QUALIF IER EQUAL;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:IDENTIFIER:MODE
STANDARD;
:SEARCH: SEARCH1:TRIGGER:A:BUS:B2:CAN:IDENTIFIER:MODE
STANDARD;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:IDENTIFIER:VALUE
"XXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:IDENTIFIER:VALUE
"XXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:CONDITION SYNCFIELD;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:CONDITION SYNCFIELD;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:VALUE"XXXXXXXX";
:SEARCH:SEARCH1:TRI GGER:A:BUS:B2:LIN:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:HIVALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:DATA:HIVALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER EQUAL;
:SEARCH:SEARCH1:TRIG GER:A:BUS:B2:LIN:DATA:QUALIFIER EQUAL;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:IDENTIFIER:VALUE
"XXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:IDENTIFIER:VALUE
"XXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:ERRTYPE SYNC;
```

```
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:ERRTYPE SYNC;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:PARALLEL:VALUE
"XXXXXXXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:PARALLEL:VALUE
"XXXXXXXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:SOURCE
B1;:SEARCH:SEARCH1:TRIGGER:A:TYPE EDGE;
LEVEL 0.0E+0;LEVEL:CH1 0.0E+0; CH20.0E+0;CH30.0E+0;CH4
0.0E+0;MATH 0.0E+0;
REF1 0.0E+0 ;REF2
0.0E+0;:SEARCH:SEARCH1:TRIGGER:A:UPPERTHRESHOLD:CH1 1.4000;
CH2 1.4000; CH3 1.4000;CH4 1.4000;MATH 492.0000E-3;REF1
492.0000E-3;REF2 492.0000E-3;
:SEARCH:SEARCH1:TRIGGER:A:LOWERTHRESHOLD:CH1 0.0E+0;CH2
0.0E+0; CH30.0E+0;
CH4 0.0E+0;MATH 0.0E+0;REF1 0.0E+0;REF2 0.0E+0;
:SEARCH:SEARCH1:TRIGGER:A:EDGE:SOURCE CH1;SLOPE RISE;
:SEARCH:SEARCH1:TRIGGER:A:LOGIC:FUNCTION AND;
THRESHOLD:CH1 0.0E+0;CH2 0.0E+0; CH3 0.0E+0;CH4 0.0E+0;MATH
0.0E+0;REF1 0.0E+0; REF20.0E+0;
:SEARCH:SEARCH1:TRIGGE R:A:LOGIC:INPUT:CH1 X;CH2 X;CH3 X;CH4
X;MATH X;REF1 X;REF2 X;
REF3 X;REF4 X;CLOCK:SOURCE NONE;EDGE RISE;
:SEARCH:SEARCH1:TRIGGER:A:LOGIC:INPUT:D0 X;D1 X;D2 X;
D3 X ;D4 X;D5 X;D6 X;D7 X;D8 X;D9 X;D10 X;D11 X;D12 X;D13
X;D14 X;D15 X;
:SEARCH: SEARCH1:TRIGGER:A:LOGIC:PATTERN:INPUT:CH1 X;CH2
X;CH3 X;
CH4 X;MATH X;REF1 X;REF2 X;RE F3 X;REF4 X;D0 X;D1 X;D2 X;D3
X;D4 X;
D5 X;D6 X;D7 X ;D8 X;D9X;D10 X;D11 X;D12 X; D13 X;D14 X;D15
X;
:SEARCH:SEARCH1:TRIGGER:A:LOGIC:PATTERN:WHEN
TRUE;WHEN:LESSLIMIT 2.0000E-9;MORELIMIT 2.0000E-9;
:SEARCH:SEARCH1:TRIGGER:A:PULSEWIDTH:POLARITY POSITIVE;WHEN
LESSTHAN;WIDTH 8.0000E-9;
:SEARCH:SEARCH1:TRIGGER:A:RUNT:POLARITY POSITIVE;WHEN
OCCURS;WIDTH 8.0000E-9;
:SEARCH:SEARCH1:TRIGGER:A:TRANSITION:POLARITY POSITIVE;WHEN
SLOWER;DELTATIME 8.0000E-9;
:SEARCH:SEARCH1:TRIGGER:A:SETHOLD:CLOCK:SOURCE CH1;EDGE
RISE;THRESHOLD 0.0E+0;
:SEARCH:SEARCH1:TRIGGER:A:SETHOLD:DATA:SOURCE NONE;THRESHOLD
9.9100E+37;
:SEARCH:SEARCH1:TRIGGER:A:SETHOLD:HOLDTIME 2.000
0E-9;SETTIME 2.0000E-9;
```

```
THRESHOLD:CH1 0.0E+0;CH2 0.0E+0;CH3 0.0E+0;CH4 0.0E+0; MATH
0.0E+0;REF1 0.0E+0;REF2 0.0E+0;
:SEARCH:SEARCH1:STATE 0
```

SEARCH:SEARCH<x>:COPy (No Query Form)

Copies the search criteria to the trigger, or the trigger criteria to a search. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:COPy {SEARCHtotrigger|TRIGgertosearch}
```

Arguments

`SEARCHtotrigger` copies the search criteria to the trigger

`TRIGgertosearch` copies the trigger criteria to the search

SEARCH:SEARCH<x>:STATE

Sets the search state to on or off. <x> is the search number, which is always 1. The query form returns the search state.

Group Search

Syntax

```
SEARCH:SEARCH<x>:STATE {<NR1>|OFF|ON}
SEARCH:SEARCH<x>:STATE?
```

Arguments

`OFF` or `<NR1>` = 0 sets the search state to off.

`ON` or `<NR1>` ≠ 0 sets the search state to on.

SEARCH:SEARCH<x>:TOTAL? (Query Only)

Returns the total number of matches for the search. The total number of matches may be more than the number of marks placed. <x> is the search number, which is always 1.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TOTAL?
```

Returns

<NR1> is the total number of matches.

SEARCH:SEARCH<x>:TRIGger:A:BUS? (Query Only)

Queries the SEARCH:SEARCH<x>:TRIGger:A:BUS? settings.

Group

Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS?
SEARCH:SEARCH<x>:TRIGger:A:BUS??
```

Examples

SEARCH:SEARCH1:TRIGGER:A:BUS? might return

```
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:CONDITION SS;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:CONDITION SS;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:DATA:MOSI:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:DATA:MOSI:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:DATA:MISO:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:DATA:MISO:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:SPI:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:SPI:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:CONDITION TXSTART;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:CONDITION TXSTART;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:RX:DATA:SIZE 1;
:SEARCH:SEARCH1 :TRIGGER:A:BUS:B2:RS232C:RX:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:RX:DATA:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:RX:DATA:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:TX:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:TX:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:RS232C:TX:DATA:VALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:RS232C:TX:DATA:VALUE "
XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:CONDITION START;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:CONDITION START;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:DATA:VALUE "XXXXXXXX";
```

```
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:DATA:VALUE "XXXXXXXX";
:SEARCH1:TRIGGER:A:BUS:B1:I2C:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C: DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:ADDRESS:MODE ADDR7;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:ADDRESS:MODE ADDR 7;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:ADDRESS:TYPE USER;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:ADDRESS:TYPE USER;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:I2C:ADDRESS:VALUE
"XXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:I2C:ADDRESS:VALUE
"XXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:CONDITION SOF;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:CONDITION SOF;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:FRAMETYPE DATA;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:FRAMETYPE DATA;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:  CAN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:DATA:DIRECTION NOCARE;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:DATA:QUALIFIER EQUAL;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:DATA:QUALIFIER EQUAL;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:IDENTIFIER:MODE
STANDARD;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:IDENTIFIER:MODE
STANDARD;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:CAN:IDENTIFIER:VALUE
"XXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:CAN:IDENTIFIER:VALUE
"XXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:CONDITION SYNCFIELD;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:CONDITION SYNCFIELD;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:DATA:SIZE 1;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:DATA:VALUE "XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:HIVALUE
"XXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:DATA:HIVALUE "XXXXXXXX
";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER EQUAL;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:DATA:QUALIFIER EQUAL;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:IDENTIFIER:VALUE
"XXXXXX";
```

```
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:IDENTIFIER:VALUE
"XXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:ERRTYPE SYNC;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:LIN:ERRTYPE SYNC;
:SEARCH:SEARCH1:TRIGGER:A:BUS:B1:PARALLEL:VALUE
"XXXXXXXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:B2:PARALLEL:VALUE
"XXXXXXXXXXXXXXXX";
:SEARCH:SEARCH1:TRIGGER:A:BUS:SOURCE B1
```

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:CONDition

Sets or returns the search condition for a CAN trigger search. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions

This command requires a DPO2AUTO application module.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:CONDition
{SOF|FRAMEtype|IDentifier|DATA|IDANDDATA|EOF|ACKMISS|ERROR}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:CONDition?
```

Arguments

`SOF` specifies a search based on the start of frame.

`FRAMEtype` specifies a search based on the frame type.

`IDentifier` specifies a search based on the frame identifier.

`DATA` specifies a search based on the frame data.

`IDANDDATA` specifies a search based on the frame identifier and data.

`EOF` specifies a search base on the end of frame.

`ACKMISS` specifies a search based on the missing ACK field.

`ERROR` specifies a search based on a bit stuffing error.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:DIRection

Sets or returns the CAN search to be valid for Read, Write, or Either condition if the criteria is IDentifier. `SEARCH<x>` is the search number and `B<x>` is the bus number. This only applies if the search condition is IDentifier.

Conditions This command requires a DPO2AUTO application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:DIRection
{READ|WRITE|NOCARE}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:DIRection?
```

Related Commands SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:CONDition

Arguments `READ` specifies the read condition.

`WRITE` specifies the write condition.

`NOCARE` specifies either a read or write condition.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier

Sets or returns the CAN data qualifier for a search. `SEARCH<x>` is the search number and `B<x>` is the bus number. This only applies if the trigger condition is IDANDDATA or DATA.

Conditions This command requires a DPO2AUTO application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier
{LESSThan|MOREThan|UNEQual|LESSEQual|MOREEQual|EQual}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier?
```

Related Commands SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue

Arguments `LESSThan` searches for bus data less than the value specified by SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue.

`MOREThan` searches for bus data more than the value specified by SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue.

`UNEQual` searches for bus data not equal to the value specified by SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue.

LESSEQual searches for bus data less than or equal to the value specified by SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue.

MOREEQual searches for bus data more than or equal to the value specified by SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue.

EQual searches for bus data equal to the value specified by SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:SIZe

Sets or returns the length of the data string in bytes to be used for a CAN search if the search condition is DATA or IDANDDATA. SEARCH<x> is the search number and B<x> is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:SIZe <NR1>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:SIZe?
```

Arguments <NR1> is the data string length in bytes.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue

Sets or returns the binary data string to be used for a CAN search if the search condition is ID or IDANDDATA. SEARCH<x> is the search number and B<x> is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue <bin>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue?
```

Related Commands SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier

Arguments <bin> is the data in binary format.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:FRAMEtype

Sets or returns the CAN Frame Type to be used if the trigger search condition is Frame Type. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:FRAMEtype
{DATA|REMote|ERRor|OVERLoad}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:FRAMEtype?
```

Arguments

`DATA` specifies a data frame.

`REMote` specifies a remote frame.

`ERRor` specifies an error frame.

`OVERLoad` specifies an overload frame.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe

Sets or returns the CAN addressing mode for a trigger search to a standard or extended format. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:
ADDRess}:MODe {STandard|EXTended}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:
ADDRess}:MODe?
```

Arguments

`STandard` specifies an 11-bit identifier field.

`EXTended` specifies a 29-bit identifier field.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue

Sets or returns the binary address string to be used for a CAN trigger search if the search condition is ID or IDANDDATA. SEARCH<x> is the search number and B<x> is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:
ADDRess}:VALue <bin>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:
ADDRess}:VALue?
```

Arguments <bin> is the address in binary format.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe

Sets or returns the I2C address mode to 7 or 10-bit. SEARCH<x> is the search number and B<x> is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe
{ADDR7|ADDR10}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe?
```

Arguments ADDR7 specifies 7-bit addresses.

ADDR10 specifies 10-bit addresses.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe

Sets or returns the I2C address type. SEARCH<x> is the search number and B<x> is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe
{GENeralcall|STARtbyte|HSmode|EEPROM|USER}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe?
```

Arguments

`GENeralcall` specifies the GENeralcall address type.

`STARtbyte` specifies the STARtbyte address type.

`HSmode` specifies the HSmode address type

`EEPROM` specifies the EEPROM address type.

`USER` specifies a user address.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue

Sets or returns the binary address string to be used for an I2C trigger search if the search condition is ADDR or ADDRANDDATA. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue <bin>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue?
```

Arguments `<bin>` is the address in binary format.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:CONDition

Sets or returns the search condition for an I2C trigger search. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:CONDition
{STARt|STOP|REPEATstart|ACKMISS|ADDRess|DATA|ADDRANDDATA}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:CONDition?
```

Arguments

`STARt` specifies a search based on a start condition.

`STOP` specifies a search based on a stop condition.

`REPEATstart` specifies a search based on a repeat of start condition.

`ACKMISS` specifies a search based on a missing acknowledgement condition.

`ADDRess` specifies a search based on an address.

`DATA` specifies a search based on a data condition.

`ADDRANDDATA` specifies a search based on an address and data condition.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:DIRection

Sets or returns the I2C search condition to be valid on a Read, Write, or Either condition. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:DIRection
{READ|WRITE|NOCARE}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:DIRection?
```

Arguments

`READ` specifies a read condition.

`WRITE` specifies a write condition.

`NOCARE` specifies either a read or write condition.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:SIZe

Sets or returns the length of the data string in bytes to be used for an I2C trigger search if the search condition is DATA or ADDRANDDATA. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:SIZe <NR1>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:SIZe?
```

Arguments `<NR1>` is the data string length in bytes.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:VALue

Sets or returns the binary data string to be used for an I2C trigger search if the search condition is DATA or ADDRANDDATA. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:VALue <bin>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:VALue?
```

Arguments `<bin>` is the data in binary format.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:CONDition

Sets or returns the search condition for a LIN search.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:CONDition
{SYNCField|IDentifier|DATA|IDANDDATA|WAKEup|SLEEP|ERROR}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:CONDition?
```

Arguments `SYNCField` specifies to search on the sync field.

`IDentifier` specifies to search on the identifier.

`DATA` specifies to search on the data.

`IDANDDATA` specifies to search on the identifier and the data.

`WAKEup` specifies to search on wake up.

`SLEEP` specifies to search on sleep.

`ERROR` specifies to search on errors.

Examples

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:CONDITION?` might return `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:CONDITION SYNCFIELD` indicating a search on the sync field.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue

Sets or returns the binary data string to be used for LIN searches if the search condition is ID or IDANDDATA.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue
<QString>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue?
```

Arguments

`<QString>` is a quoted string of 1s, 0s, or Xs representing the binary data string to be used for LIN searches if the search condition is ID or IDANDDATA.

Examples

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:HIVALUE?` might return `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:HIVALUE "XXXXXXXX"` indicating that the high value is don't care.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier

Sets or returns the LIN data qualifier. This only applies if the trigger condition is IDANDDATA or DATA.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier
{LESSThan|MOREThan|EQUal|UNEQual|
LESSEQual|MOREEQual|INrange|OUTrange}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier?
```

Arguments

`LESSThan` sets the LIN data qualifier to less than.

`MOREThan` sets the LIN data qualifier to greater than.

`EQUal` sets the LIN data qualifier to equal.

`UNEQual` sets the LIN data qualifier to not equal.

`LESSEQual` sets the LIN data qualifier to less than or equal.

`MOREEQual` sets the LIN data qualifier to more than or equal.

`INrange` sets the LIN data qualifier to in range.

`OUTrange` sets the LIN data qualifier to out of range.

Examples

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER LESSThan` sets the data qualifier to lessthan.

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER?` might return `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER EQUAL` indicating that the data qualifier is equal.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:SIZe

Sets or returns the length of the data string in bytes to be used for LIN Search, if search condition is DATA or IDANDDATA.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:SIZe <NR1>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:SIZe?
```

Arguments

`<NR1>` is the length of the data in bytes.

Examples

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:SIZE 8` sets the LIN data size is 8 bytes.

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:SIZE?` might return `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:SIZE 1` indicating that the LIN data size is 1 byte.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:VALue

Sets or returns the binary data string used for a LIN search if the search condition is ID or IDANDDATA.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:VALue <QString>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:VALue?
```

Arguments `<QString>` is the binary data string for the search.

Examples `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:VALUE?` might return `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:DATA:VALUE "XXXXXXXX"` indicating that the data value is don't care.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:ERRTYPE

Sets or returns the error type used for a LIN Search.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:ERRTYPE
{SYNC|PARity|CHecksum}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:ERRTYPE?
```

Arguments `SYNC` specifies a sync error type.

`PARity` specifies a parity error type.

`CHecksum` specifies a checksum error type.

Examples

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:ERRTYPE?` might return `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:ERRTYPE SYNC` indicating a SYNC error type.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue

Sets or returns the binary address string used for LIN search if search condition is ID or IDANDDATA.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue
<QString>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue?
```

Arguments

`<QString>` is a quoted string specifying the binary address string to be used for LIN search if search condition is ID or IDANDDATA.

Examples

`SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:IDENTIFIER:VALUE?` might return `SEARCH:SEARCH1:TRIGGER:A:BUS:B1:LIN:IDENTIFIER:VALUE "XXXXXX"` indicating that the binary address is undefined.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:PARallel:VALue

Sets or returns the binary data string to be used for a parallel trigger search. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:PARallel:VALue <QString>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:PARallel:VALue?
```

Arguments

`<QString>` is the binary data string.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:CONDition

Sets or returns the condition for a RS232 trigger search. SEARCH<x> is the search number and B<x> is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:CONDition
{RXSTArt|RXDATA|RXENDPacket|RXPARity|TXSTArt|
TXDATA|TXENDPacket|TXPARity}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:CONDition?
```

Arguments

RXSTArt specifies a search based on the RX Start Bit.

RXDATA specifies a search based on RX Data.

RXENDPacket specifies a search based on the RX End of Packet condition.

RXPARity specifies a search based on the RX parity.

TXSTArt specifies a search base on the TX Start Bit.

TXDATA specifies a search based on TX Data.

TXENDPacket specifies a search based on the TX End of Packet condition.

TXPARity specifies a search based on the TX parity.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe

Sets or returns the length of the data string for a RS232 trigger search if the trigger condition is RX. SEARCH<x> is the search number and B<x> is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe?
```

Arguments <NR1> is the length of the data string in Bytes.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue

Sets or returns the binary data string for a RS232 trigger search if the condition involves RX. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue?
```

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe

Sets or returns the length of the data string to be used for a RS232 trigger search if the Trigger condition is TX. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe?
```

Arguments `<NR1>` is the length of the data string in Bytes.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue

Sets or returns the binary data string to be used for a RS232 trigger search if the condition involves TX. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue?
```

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:CONDition

Sets or returns the search condition for a SPI trigger search. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:CONDition
{SS|MISO|MOSI|MISOMOSI}
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:CONDition?
```

Arguments

`SS` specifies a search based on the Slave Selection condition.

`MISO` specifies a search based on the Master-In Slave-Out condition.

`MOSI` specifies a search based on the Master-Out Slave-In condition.

`MISOMOSI` specifies a search based on the Master-In Slave-Out and Master-Out Slave-In conditions.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MISO|:IN}:VALue

Sets or returns the binary data string for an SPI trigger search if the search condition is MISO or MISOMOSI. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MISO|:IN}:
VALue <bin>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MISO|:IN}:
VALue?
```

Arguments

`<bin>` is the data string in binary format.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MOSI|:OUT}:VALue

Sets or returns the binary data string for an SPI trigger search if search the condition is MOSI, or MISOMOSI. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MOSI|:OUT}:
VALue <bin>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MOSI|:OUT}:
VALue?
```

Arguments `<bin>` is the data in binary format.

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa:SIZe

Sets or returns the length of the data string for an SPI trigger search if the search condition is MISO, MOSI, or MISOMOSI. `SEARCH<x>` is the search number and `B<x>` is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa:SIZe <NR1>
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa:SIZe?
```

Arguments `<NR1>` is the data string length in bytes.

SEARCH:SEARCH<x>:TRIGger:A:BUS:SOUrce

Sets or returns a bus serial search. <x> is the search number.

Conditions This command requires a DPO2AUTO or DPO2EMBD application module.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:BUS:SOUrce {B1|B2}
SEARCH:SEARCH<x>:TRIGger:A:BUS:SOUrce?
```

Arguments `B1` specifies the Bus 1 source.

`B2` specifies the Bus 2 source.

SEARCH:SEARCH<x>:TRIGger:A:EDGE:SLOpe

Sets or returns the slope for an edge trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SLOpe {RISe|FALL}
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SLOpe?
```

Arguments `RISe` specifies a rising edge.

`FALL` specifies a falling edge.

SEARCH:SEARCH<x>:TRIGger:A:EDGE:SOUrce

Sets or returns the source waveform for an edge trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SOUrce
{CH1|CH2|CH3|CH4|MATH}
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SOUrce?
```

Arguments `CH<x>` specifies one input channel as the edge source, where <x> is the channel number.

`MATH` specifies the math waveform as the search source.

SEARCH:SEARCH<x>:TRIGger:A:LEVel

Sets or returns the level for an edge trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LEVel {<NR3>|TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:LEVel?
```

Arguments

`<NR3>` specifies the trigger level, in volts.

`TTL` specifies a preset `TTL` high level of 1.4 V.

`ECL` specifies a preset `ECL` high level of –1.3 V

SEARCH:SEARCH<x>:TRIGger:A:LEVel:CH<x>

Sets or returns the level for an edge trigger search to determine where to place a mark. `SEARCH<x>` is the search number and `CH<x>` is the channel number. Each channel can have an independent level.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LEVel:CH<x> {<NR3>|TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:LEVel:CH<x>?
```

Arguments

`<NR3>` specifies the trigger level in volts.

`TTL` specifies a preset `TTL` high level of 1.4 V.

`ECL` specifies a preset `ECL` high level of –1.3 V

SEARCH:SEARCH<x>:TRIGger:A:LEVel:MATH[1]

Sets or returns the math waveform level for an edge trigger search to determine where to place a mark. <x> is the search number. The value of MATH is 1 for all oscilloscopes.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LEVel:MATH {TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:LEVel:MATH?
```

Arguments

`TTL` specifies a preset `TTL` high level of 1.4 V.

`ECL` specifies a preset `ECL` high level of –1.3 V

SEARCH:SEARCH<x>:TRIGger:A:LEVel:REF<x>

Sets or returns the specified reference waveform level for an edge trigger search to determine where to place a mark. `SEARCH<x>` is the search number and `REF<x>` is the reference channel number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LEVel:REF<x> {TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:LEVel:REF<x>?
```

Arguments

`TTL` specifies a preset `TTL` high level of 1.4 V.

`ECL` specifies a preset `ECL` high level of –1.3 V

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:FUNCtion

Sets or returns the logic operator for a logic trigger search to determine where to place a mark. <x> is the search number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:FUNCtion {AND|NANd}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:FUNCtion?
```

Arguments

`AND` places a mark if all conditions are true.

`NANd` places a mark if any of the conditions is false.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CH<x>

Sets or returns the Boolean logic criteria for a logic trigger search to determine where to place a mark. `SEARCH<x>` is the search number and `CH<x>` is the channel number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CH<x> {HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CH<x>?
```

Arguments

`HIGH` specifies the logic high.

`LOW` specifies the logic low.

`X` specifies a "don't care" state.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:EDGE

Sets or returns whether the clock edge is a rising or falling for a logic search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:EDGE
{FALL|RISe|EITher}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:EDGE?
```

Arguments

`RISe` specifies a rising edge.

`FALL` specifies a falling edge.

`EITher` specifies either a falling or rising edge.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:SOUrce

Sets or returns the clock source definition for a logic trigger search. <x> is the search number. If a clock source is defined, then the logic search is determined by the state of the other inputs at the clock transition. If no clock source is defined, then the logic search is determined only by the state of the inputs.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:SOUrce
{CH<x>|MATH|REF|D<x>|NONe}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:SOUrce?
```

Arguments

CH<x> specifies a channel input as the clock source, where <x> = 1, 2, 3, or 4.

MATH specifies the math waveform as the clock source.

REF specifies the reference waveform as the clock source.

D<x> specifies the digital waveform as the clock source, where <x>=0–15

NONe specifies no clock source.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:D<x>

Sets or returns the waveform logic criteria for a trigger search. SEARCH<x> is the search number and D<x> is the digital channel number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:D<x> {HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:D<x>?
```

Arguments

HIGH specifies a high logic level.

LOW specifies a low logic level.

X specifies a “don't care” condition.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:MATH

Sets or returns the Boolean logic criteria for a logic trigger search to determine where to place a mark. <x> is the search number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:MATH {HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:MATH?
```

Arguments

HIGH specifies a high logic level.

LOW specifies a low logic level.

X specifies a “don't care” condition.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:REF<x>

Sets or returns the Boolean logic criteria for a logic trigger search to determine where to place a mark. SEARCH<x> is the search number and REF<x> is the reference channel number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:REF<x> {HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:REF<x>?
```

Arguments

HIGH specifies a high logic level.

LOW specifies a low logic level.

X specifies a “don't care” condition.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:CH<x>

Sets or returns the logic criteria for a logic pattern trigger search to determine where to place a mark. SEARCH<x> is the search number and CH<x> is the channel number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:CH<x>
{HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:CH<x>?
```

Arguments

HIGH specifies a high logic level.

LOW specifies a low logic level.

X specifies a “don't care” condition.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:D<x>

Sets or returns the logic criteria for a logic pattern search to determine where to place a mark. SEARCH<x> is the search number and D<x> is the digital channel number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:D<x>
{HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:D<x>?
```

Arguments

`HIGH` specifies a high logic level.

`LOW` specifies a low logic level.

`X` specifies a “don't care” condition.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:MATH

Sets or returns the Boolean logic criteria for a logic pattern trigger search to determine where to place a mark. <x> is the search number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:MATH
{HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:MATH?
```

Arguments

`HIGH` specifies a high logic level.

`LOW` specifies a low logic level.

`X` specifies a “don't care” condition.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:REF<x>

Sets or returns the Boolean logic criteria for a pattern trigger search to determine where to place a mark. `SEARCH<x>` is the search number and `REF<x>` is the reference channel number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:REF<x>
{HIGH|LOW|X}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:REF<x>?
```

Arguments

`HIGH` specifies a high logic level.

`LOW` specifies a low logic level.

`X` specifies a “don't care” condition.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn

Sets or returns the condition for generating a logic pattern trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn
{TRUe|FALSe|LESSThan|MOREThan|Than|EQUal|UNEQual}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn?
```

Arguments

`TRUe` places a mark when the pattern becomes true.

`FALSe` places a mark when the pattern becomes false.

`LESSThan` places a mark if the specific pattern is true less than the time set by the SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit command.

`MOREThan` places a mark if the specific pattern is true more than the time set by the SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit command.

`EQUal` places a mark if the specific pattern is true longer than the time set by the SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit command, but less than the specified time set by the SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit command.

`UNEQual` places a mark if the specific pattern is true less than the time set by the SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit command, or longer than the specified time set by the SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit command.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit

Sets or returns the maximum time that the selected pattern may be true and still generate an A logic pattern search to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit
<NR3>
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit?
```

Arguments `<NR3>` specifies the maximum amount of time to hold the pattern true.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit

Sets or returns the minimum time that the selected pattern may be true and still generate an A logic pattern search to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit
<NR3>
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit?
```

Arguments <NR3> specifies the minimum amount of time to hold the pattern true.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:CH<x>

Sets or returns the channel threshold level for a logic trigger search to determine where to place a mark. SEARCH<x> is the search number and CH<x> is the channel number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:CH<x>
{<NR3>|TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:CH<x>?
```

Arguments <NR3> specifies the threshold voltage level.

TTL specifies a preset TTL high level of 1.4 V.

ECL specifies a preset ECL high level of –1.3 V.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:MATH

Sets or returns the math waveform threshold level for a logic trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:MATH
{TTL|ECL|<NR3>}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:MATH?
```

Arguments

TTL specifies a preset TTL high level of 1.4 V.

ECL specifies a preset ECL high level of –1.3 V.

<NR3> specifies the threshold voltage level.

SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:REF<x>

Sets or returns the reference waveform threshold level for a logic trigger search to determine where to place a mark. SEARCH<x> is the search number and REF<x> is the reference channel number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:REF<x>
{TTL|ECL|<NR3>}
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:REF<x>?
```

Arguments

TTL specifies a preset TTL high level of 1.4 V.

ECL specifies a preset ECL high level of –1.3 V.

<NR3> specifies the threshold voltage level.

SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:CH<x>

Sets or returns the channel waveform lower threshold to determine where to place a mark. This setting is applied to all channel searches that use a lower threshold. SEARCH<x> is the search number and CH<x> is the channel number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:CH<x>
{TTL|ECL|<NR3>}
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:CH<x>?
```

Arguments

TTL specifies a preset TTL level of 1.4 V.

ECL specifies a preset ECLlevel of –1.3 V.

SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:MATH

Sets or returns the math waveform lower threshold to determine where to place a mark. This setting is applied to all math searches that use a lower threshold.
<x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:MATH
{TTL|ECL|<NR3>}
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:MATH?
```

Arguments

`TTL` specifies a preset `TTL` level of 1.4 V.

`ECL` specifies a preset `ECL` level of –1.3 V.

`<NR3>` specifies the threshold voltage level.

SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:REF<x>

Sets or returns the reference waveform lower threshold to determine where to place a mark. This setting is applied to all reference searches that use a lower threshold.
`SEARCH<x>` is the search number and `REF<x>` is the reference channel number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:REF<x>
{TTL|ECL|<NR3>}
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:REF<x>?
```

Arguments

`TTL` specifies a preset `TTL` level of 1.4 V.

`ECL` specifies a preset `ECL` level of –1.3 V.

`<NR3>` specifies the threshold voltage level.

SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:POLarity

Sets or returns the polarity for a pulse trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:POLarity
{NEGative|POSitive}
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:POLarity?
```

Arguments `POSITIVe` places a mark only when the polarity of the pulse is positive.

`NEGative` places a mark only when the polarity of the pulse is negative.

SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:SOUrce

Sets or returns the source waveform for a pulse trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:SOUrce
{CH1|CH2|CH3|CH4|MATH|REF}
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:SOUrce?
```

Arguments `CH<x>` specifies one input channel as the edge source, where <x> = 1, 2, 3 or 4.

`MATH` specifies the math waveform as the search source.

`REF` specifies the reference waveform as the search source.

SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WHEn

Sets or returns the condition for generating a pulse width search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WHEn
{LESSthan|MOREthan|EQual|UNEQual}
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WHEn?
```

Arguments `LESSThan` places a mark if the pulse width is less than the time set by the SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth command.

`MOREthan` places a mark if the pulse width is more than the time set by the SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth command.

`EQUal` places a mark if the pulse width is equal to the time set by the SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth command within a tolerance of ±5%.

`UNEQual` places a mark if the pulse width is unequal to the time the time set by the SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth command within a tolerance of ±5%.

SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth

Sets or returns the pulse width setting for a pulse width trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth <NR3>
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth?
```

Arguments `<NR3>` is the pulse width.

SEARCH:SEARCH<x>:TRIGger:A:RUNT:POLarity

Sets or returns the polarity setting for a runt trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:RUNT:POLarity
{EITher|NEGative|POSitive}
SEARCH:SEARCH<x>:TRIGger:A:RUNT:POLarity?
```

Arguments `POSitive` places a mark when the rising edge crosses the low threshold and the falling edge re-crosses the low threshold without either edge ever crossing the high threshold.

`NEGative` places a mark when the falling edge crosses the high threshold and the rising edge re-crosses the high threshold without either edge ever crossing the low threshold.

`EITher` places a mark on a runt of either polarity.

SEARCH:SEARCH<x>:TRIGger:A:RUNT:SOUrce

Sets or returns the source setting for a runt trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:RUNT:SOUrce
{CH1|CH2|CH3|CH4|MATH|REF}
SEARCH:SEARCH<x>:TRIGger:A:RUNT:SOUrce?
```

Arguments

`CH1–CH4` specifies an input channel as the edge source.

`MATH` specifies the math waveform as the search source.

`REF` specifies the reference waveform as the search source.

SEARCH:SEARCH<x>:TRIGger:A:RUNT:WHEn

Sets or returns the condition setting for a runt trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:RUNT:WHEn
{LESSthan|MOREthan|EQual|UNEQual|OCCURS}
SEARCH:SEARCH<x>:TRIGger:A:RUNT:WHEn?
```

Arguments

`OCCURS` argument specifies a trigger event if a runt of any detectable width occurs.

`LESSthan` argument sets the oscilloscope to trigger if the a runt pulse is detected with width less than the time set by the SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth command.

`MOREthan` argument sets the oscilloscope to trigger if the a runt pulse is detected with width more than the time set by the SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth command.

`EQUal` argument sets the oscilloscope to trigger when the pattern is true for a time period equal to the time period specified in SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth within a ±5% tolerance.

`NOTEQual` argument sets the oscilloscope to trigger when the pattern is true for a time period greater than or less than (but not equal) the time period specified in SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth within a ±5% tolerance.

SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth

Sets or returns the width setting for a runt trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth <NR3>
SEARCH:SEARCH<x>:TRIGger:A:RUNT:WIDth?
```

Arguments `<NR3>` specifies the minimum width, in seconds.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:EDGE

Sets or returns the clock slope setting for a setup/hold trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:EDGE {FALL|RISe}
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:EDGE?
```

Arguments `FALL` specifies polarity as the clock falling edge.

`RISe` specifies polarity as the clock rising edge.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:SOUrce

Sets or returns the clock source setting for an setup/hold trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:SOUrce
{CH1|CH2|CH3|CH4|MATH|REF}
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:SOUrce?
```

Related Commands SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:SOUrce

Arguments

`CH1–CH4` specifies an input channel as the edge source.

`MATH` specifies the math waveform as the search source.

`REF` specifies the reference waveform as the search source.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:THReshold

Sets or returns the clock threshold setting for an setup/hold trigger search to determine where to place a mark. <x> is the search number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:THReshold
{<NR3>|TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:THReshold?
```

Arguments

`TTL` specifies a preset `TTL` high level of 1.4 V.

`ECL` specifies a preset `ECL` high level of –1.3 V.

`<NR3>` is the clock level, in volts.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:SOUrce

Sets or returns the data source setting for an setup/hold trigger search to determine where to place a mark. <x> is the search number. You cannot specify the same source for both clock and data.

Group

Search

Syntax

DPO Models:

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:SOUrce
{CH1|CH2|CH3|CH4|MATH|REF|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:SOUrce?
```

Related Commands

SEARCH:SEARCH<x>:TRIGger:A:SETHold:CLOCk:SOUrce

Arguments

DPO Models:

`CH1–CH4` specifies an input channel as the search source.

MATH specifies the math waveform as the search source.

REF specifies the reference waveform as the search source.

D0–D15 specifies the digital waveform as the search source.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:THReshold

Sets or returns the data threshold setting for an setup/hold trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:THReshold
{<NR3>|TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:THReshold?
```

Arguments TTL specifies a preset TTL high level of 1.4 V.

ECL specifies a preset ECL high level of –1.3 V.

<NR3> is the clock level, in volts.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:HOLDTime

Sets or returns the hold time setting for an setup/hold trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:HOLDTime <NR3>
SEARCH:SEARCH<x>:TRIGger:A:SETHold:HOLDTime?
```

Arguments <NR3> specifies the hold time setting in seconds. Positive values for hold time occur after the clock edge. Negative values occur before the clock edge.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:SETTime

Sets or returns the setup time setting for an setup/hold trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:SETTime <NR3>
SEARCH:SEARCH<x>:TRIGger:A:SETHold:SETTime?
```

Arguments <NR3> specifies the setup time for setup and hold violation triggering.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:CH<x>

Sets or returns the trigger search setup and hold lower threshold to determine where to place a mark. Search<x> is the search number, which is always 1. CH<x> is the channel number.

Conditions MSO oscilloscopes only.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:CH<x> <NR3>
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:CH<x>?
```

Arguments <NR3> is the lower threshold in volts.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold{:MATH|:MATH1}

Sets or returns the trigger search setup and hold lower threshold to determine where to place a mark. Where <x> is the search number, which is always 1.

Conditions MSO oscilloscopes only.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold{:MATH|:MATH1}
<NR3>
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold{:MATH|:MATH1}?
```

Arguments <NR3> is the lower threshold in volts.

SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:REF<x>

Sets or returns the trigger search setup and hold lower threshold to determine where to place a mark. Search<x> is the search number, which is always 1. REF<x> is the reference waveform number.

Conditions MSO oscilloscopes only.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:REF<x> <NR3>
SEARCH:SEARCH<x>:TRIGger:A:SETHold:THReshold:REF<x>?
```

Arguments <NR3> is the lower threshold in volts.

SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:DELTatime

Sets or returns the transition time setting for an transition trigger search to determine where to place a mark.<x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:DELTatime
<NR3>
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:DELTatime?
```

Arguments <NR3> specifies the transition time, in seconds.

SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:POLarity

Sets or returns the polarity setting for an transition trigger search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:POLarity
{EITher|NEGative|POSitive}
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:POLarity?
```

Arguments

`POSitive` specifies that a pulse edge must traverse from the lower (most negative) to higher (most positive) level for transition triggering to occur.

`NEGative` specifies that a pulse edge must traverse from the upper (most positive) to lower (most negative) level for transition triggering to occur.

`EITher` specifies either positive or negative polarity.

SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:SOUrce

Sets or returns the source setting for an transition trigger search to determine where to place a mark. <x> is the search number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:SOUrce
{CH1|CH2|CH3|CH4|MATH}
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:SOUrce?
```

Arguments

`CH1–CH4` specifies one input channel as the edge source.

`MATH` specifies the math waveform as the search source.

SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:WHEn

Sets or returns the condition setting for an transition trigger search to determine where to place a mark. <x> is the search number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:WHEn
{SLOWer|FASTer|EQual|UNEQual}
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:RISEFall}:WHEn?
```

Arguments

`FASTer` sets the trigger to occur when the transitioning signal is faster than the set volts/second rate.

`SLOWer` sets the trigger to occur when the transitioning signal is slower than the set volts/second rate.

`EQual` sets the trigger to occur when the transitioning signal is equal to the set volts/second rate within a ±5% tolerance.

UNEQual sets the trigger to occur when the transitioning signal is not equal to the set volts/second rate ±5%.

SEARCH:SEARCH<x>:TRIGger:A:TYPe

Sets or returns the trigger type setting for a search to determine where to place a mark. <x> is the search number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:TYPe
{EDGe|SETHold|PULSEWidth|RUNt|TRAnsition|LOGIc|BUS (with the
appropriate application module installed)}
SEARCH:SEARCH<x>:TRIGger:A:TYPe?
```

Arguments

RUNt triggers when a pulse crosses the first preset voltage threshold but does not cross the second preset threshold before recrossing the first. The thresholds are set with the SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:CH<x> and SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:CH<x> commands.

PULSEWIdth triggers when a pulse is found that has the specified polarity and is either inside or outside the limits as specified by SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit and SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit. The polarity is selected using the SEARCH:SEARCH<x>:TRIGger:A:RUNT:POLarity command.

TRAnsition triggers when a pulse crosses both thresholds in the same direction as the specified polarity and the transition time between the two threshold crossings is greater or less than the specified time delta.

SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:CH<x>

Sets or returns the channel waveform upper threshold to determine where to place a mark. This setting is applied to all channel searches that uses an upper threshold.SEARCH<x> is the search number and CH<x> is the channel number.

Group Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:CH<x> {TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:CH<x>?
```

Arguments

TTL specifies a preset TTL high level of 1.4 V.

ECL specifies a preset ECL high level of –1.3 V.

SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:MATH

Sets or returns the math waveform upper threshold to determine where to place a mark. This setting is applied to all math waveform searches that uses an upper threshold. <x> is the search number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:MATH {TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:MATH?
```

Arguments

TTL specifies a preset TTL high level of 1.4 V.

ECL specifies a preset ECL high level of –1.3 V.

SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:REF<x>

Sets or returns the reference waveform upper threshold to determine where to place a mark. This setting is applied to all reference waveform searches that uses an upper threshold. SEARCH<x> is the search number and REF<x> is the reference channel number.

Group

Search

Syntax

```
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:REF<x> {TTL|ECL}
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:REF<x>?
```

Arguments

TTL specifies a preset TTL high level of 1.4 V.

ECL specifies a preset ECL high level of –1.3 V.

SELect

Sets or returns the selected waveform display (controlled by the front-panel) on or off.

Group

Vertical

Syntax

```
SELect {ON|OFF}
```

Arguments

`ON` turns the selected waveform display on.

`OFF` turns the selected waveform display off.

Examples

`SELECT?` might return the following

```
:SELECT:BUS1 0;BUS2 0;CH1 1;CH2 0;CH3 0;CH4 0;MATH 0;REF1
0;REF2 0;D0 0;D1 0;D2 0;D3 0;D4 0;D5 0;D6 0;D7 0;D8 0;D9
0;D10 0;D11 0;D12 0;D13 0;D14 0;D15 0,CONTROL CH1
```

SELect:BUS<x>

This command turns on and off the display of the waveform for <x>, where x is the bus number. The query returns whether the channel is on or off but does not indicate whether it is the selected waveform.

Group

Vertical

Syntax

```
SELect:BUS<x> {<NR1>|OFF|ON}
SELect:BUS<x>?
```

SELect:CH<x>

Turns the display of the channel <x> waveform on or off, where <x > is the channel number. This command also resets the acquisition. The query returns whether the channel is on or off but does not indicate whether it is the selected waveform.

Group

Vertical

Syntax

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

Arguments

`ON` turns on the display of the specified waveform. This waveform also becomes the selected waveform.

`OFF` turns off the display of the specified waveform.

`<NR1>` = 0 turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

`SELECT:CH2 ON` turns the channel 2 waveform display on, and selects channel 2.

`SELECT:CH1?` might return `:SELECT:CH1 1` indicating that channel 1 is being displayed.

SELect:CONTROl

Sets or returns the waveform that is the recipient of future channel-related commands, for example, the cursor commands. The command form also performs the equivalent of a SELect:CH<x> ON command, as well as the Math, Reference, and Bus variations of that command.

Group

Vertical

Syntax

```
SELect:CONTROl {CH<x>|MATH|BUS<x>}
SELect:CONTROl?
```

Arguments

`CH<x>` specifies a channel waveform as the waveform affected by the front-panel controls. <x> is the channel number.

`MATH` specifies the math waveform as the waveform that is affected by the front-panel controls.

`BUS<x>` specifies a bus waveform as the waveform affected by the front-panel controls. <x> specifies the bus number.

Returns

`NONE` if all the channels are turned off. `NONE` is ignored on input.

Examples

`SELECT:CONTROL CH2` resets acquisition displays on channel 2, and causes the selected waveform to be the implied object of waveform commands.

`SELECT:CONTROL?` might return `:SELECT:CONTROL MATH` indicating that math is the implied object of waveform commands.

SELect:D<x>

Turns on the display of the digital channel <x> and resets the acquisition. <x > is the channel number. The query returns whether the channel is on or off but does not indicate whether it is the selected waveform.

Group

Vertical

Syntax

```
SELect:D<x> {<NR1>|OFF|ON}
SELect:D<x>?
```

Arguments

ON turns on the display of the specified waveform. This waveform also becomes the selected waveform.

OFF turns off the display of the specified waveform.

<NR1> = 0 turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

SELECT:D2 ON turns the digital channel 2 waveform display on, and selects digital channel 2.

SELECT:D2? might return :SELECT:D2 1 indicating that digital channel 2 is being displayed.

SELect:MATH[1]

Turns on and off the display of the math waveform. The query returns whether the math waveform is on or off but does not indicate whether it is the selected waveform.

Group

Vertical

Syntax

```
SELect:MATH[1] {ON|OFF|<NR1>}
SELect:MATH[1]?
```

Arguments

ON turns on the display of the specified waveform. This waveform also becomes the selected waveform.

OFF turns off the display of the specified waveform.

<NR1> = 0 turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

SELECT:MATH ON turns the math waveform display on, and selects it.

SELECT:MATH? might return :SELECT:MATH 1 indicating that the math waveform is being displayed.

SELect:REF<x>

Turns on and off the display of the reference waveform <x>. The <x > variable represents the reference channel number. The query returns whether the channel is on or off.

Group Vertical

Syntax

```
SELect:REF<x> {ON|OFF|<NR1>}
SELect:REF<x>?
```

Arguments

`ON` turns on the display of the specified waveform. This waveform also becomes the selected waveform.

`OFF` turns off the display of the specified waveform.

`<NR1>` = 0 turns off the display of the specified waveform; any other value turns on the display of the specified waveform.

Examples

`SELECT:REF2 ON` turns the channel 2 waveform display on, and selects reference waveform 2.

`SELECT:REF2?` might return `:SELECT:REF2 1` indicating that reference waveform 2 is being displayed.

SET? (Query Only)

Returns the commands that list the oscilloscope settings except for configuration information for the calibration values, the WFMInpre? query, and the WFMOutpre? query. This query allows you to record or "learn" the current oscilloscope settings. You can use these commands to return the oscilloscope to the state it was in when you made the SET? query. The SET? query always returns command headers, regardless of the setting of the HEADer command. This is because the returned commands are intended to be sent back to the oscilloscope as a command string. The VERBose command can still be used to specify whether the returned headers should be abbreviated or full-length.

This command is identical to the *LRN? command.

Group Miscellaneous

Syntax

```
SET?
```

Related Commands HEADer, *LRN?, VERBose

Examples SET? returns a long response, part of which could be as follows: :SET :ACQUIRE:STOPAFTER RUnsTOP;STATE 1;MODE SAMPLE;NUMENV INFINITE;NUMAVG 16;SAMPLINGMODE RT;:HEADER 1;:LOCK NONE;:LANGUAGE ENGLISH;:VERBOSE 1;:ALIAS:STATE 0;: DISPLAY:COLOR:PALETTE NORMAL;:DISPLAY:PERSISTENCE 0.0000;CLOCK 1;GRATICULE FULL;INTEnsITY:WAVEFORM 30;GRATICULE 75;BACKLIGHT HIGH;:HARDCOPY:INKSAVER OFF;LAYOUT LANDSCAPE;PREVIEW 0; :SAVE:IMAGE:FILEFORMAT BMP;:SAVE:WAVEFORM:FILEFORMAT INTERNAL;:SAVE:ASSIGN:TYPE SETUP;:TRIGGER:A:MODE AUTO;TYPE EDGE;LEVEL 20.0000E-3;LEVEL:CH1 20.0000E-3;CH2 0.0000;CH3 0.0000;CH4 0.0000;:TRIGGER:A:UPPERTHRESHOLD:CH1 1.4000;CH2 800.0000E-3;CH3 800.0000E-3;CH4 800.0000E-3;: TRIGGER:A:LOWERTHRESHOLD:CH1 20.0000E-3;CH2 0.0000;CH3 0.0000;CH4 0.0000;:TRIGGER:A:HOLDOFF:TIME 20.0000E-9;:TRIGGER:A:EDGE:SOURCE CH1;COUPLING DC;SLOPE RISE;:TRIGGER:A:LOGIC:CLASS SETHOLD;FUNCTION AND;THRESHOLD:CH1 20.0000E-3;CH2 0.0000;CH3 0.0000;CH4 0.0000;:TRIGGER:A:LOGIC:INPUT:CH1 X;CH2 X;CH3 X;CH4 X;CLOCK:SOURCE NONE;EDGE RISE;:TRIGGER:A:LOGIC:PATTERN:INPUT:CH1 X;CH2 X;CH3 X;CH4 X;:TRIGGER:A :LOGIC:PATTERN:WHEN TRUE;WHEN:LESSLIMIT 4.0000E-9;LIMIT 4.0000E-9;:TRIGGER:A :SETHOLD:CLOCK:SOURCE CH1;EDGE RISE;THRESHOLD 20.0000E-3;:TRIGGER:A:SETHOLD:DATA:SOURCE CH2;

SETUP<x>:DATE? (Query Only)

Returns the date when the oscilloscope setup was saved for the specified channel <x>.

Group Save and Recall

Syntax SETUP<x>:DATE?

Examples SETUP4:DATE? might return SETUP4:DATE: 04-18-06 which is the setup date for channel 4.

SETUP<x>:LABEL

Sets or returns the setup label for the specified channel <x>.

Group Save and Recall

Syntax

```
SETUP<x>:LABEL <Qstring>
```

Arguments `<Qstring>` is an alpha-numeric string of characters, enclosed in quotes, that defines the label text for SETUP<x>. The length of the string is limited to 30 characters.

Examples `SETUP:LABEL?` might return `SETUP1:LABEL: TEST 2` which is the label setup for channel 1.

SETUP<x>:TIME? (Query Only)

Returns the time when the oscilloscope setup was saved for the specified channel <x>.

Group Save and Recall

Syntax

```
SETUP<x>:TIME?
```

Examples `SETUP2:TIME?` might return "`SETUP2:TIME: 15:24:07` which is the setup time for channel 2.

*SRE

The *SRE (Service Request Enable) command sets or returns the bits in the Service Request Enable Register. For information, refer to Registers.

Group Status and Error

Syntax

```
*SRE <NR1>
*SRE?
```

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

Arguments

<NR1> is a value in the range from 0 through 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 the previous power cycle value through the current power cycle.

Examples

*SRE 48 sets the bits in the SRER to binary 00110000.

*SRE? might return 32, showing that the bits in the SRER have the binary value of 00100000.

*STB? (Query Only)

*STB? (Read Status Byte) returns the contents of the Status Byte Register (SBR) using the Master Summary Status (MSS) bit. For information, refer to Registers.

Group

Status and Error

Syntax

*STB?

Related Commands

*CLS, DESE, *ESE, *ESR?, EVENT?, EVMsg?, FACtory, *SRE

Returns

<NR1>

Examples

*STB? might return 96, showing that the SBR contains the binary value 01100000.

TEKSecure (No Query Form)

This command initializes both waveform and setup memories, overwriting any previously stored data. These are the WFMInpre?, WFMOutpre?, DATa, DATa:RESOlution and DATa:COMPosition command values after the TEKSecure operation.

```
:WFMOUTPRE:BYT_NR 1 :WFMOUTPRE:BIT_NR 8 :WFMOUTPRE:ENCDG BIN
:WFMOUTPRE:BN_FMT RI :WFMOUTPRE:BYT_OR MSB :WFMOUTPRE:WFID
"Ch1, DC coupling, 100.0mV/div, 4.000us/div, 10000 points,
Sample mode" :WFMOUTPRE:NR_PT 10000 :WFMOUTPRE:PT_FMT
Y :WFMOUTPRE:XUNIT "s" :WFMOUTPRE:XINCR 4.0000E-9
:WFMOUTPRE:XZERO -20.0000E-6 :WFMOUTPRE:PT_OFF 0
:WFMOUTPRE:YUNIT "V" :WFMOUTPRE:YMULT 4.0000E-3
:WFMOUTPRE:YOFF 0.0000 :WFMOUTPRE:YZERO 0.0000
:WFMINPRE:BYT_NR 1 :WFMINPRE:BIT_NR 8 :WFMINPRE:ENCDG BIN
:WFMINPRE:BN_FMT RI :WFMINPRE:BYT_OR MSB :WFMINPRE:NR_PT
```

```
10000 :WFMINPRE:PT_FMT Y :WFMINPRE:XUNIT "s" :WFMINPRE:XINCR
4.0000E-9 :WFMINPRE:XZERO 0.0000 :WFMINPRE:PT_OFF 0
:WFMINPRE:YUNIT "V" :WFMINPRE:YMULT 4.0000E-3 :WFMINPRE:YOFF
0.0000 :WFMINPRE:YZERO 0.0000 DATA:DESTINATION REF1
DATA:ENCDG RIBINARY DATA:SOURCE CH1 DATA:START 1 DATA:STOP
10000 DATA:WIDTH 1 :DATA:RESOLUTION FULL :DATA:COMPOSITION
COMPOSITE_YT
```

NOTE. *The TEKSecure command can take up to five minutes to complete. The oscilloscope is inoperable during this period.*

Group Miscellaneous

Syntax `TEKSecure`

Examples `TEKSECURE` initializes both waveform and setup memories.

This is a program example of how to generate an SRQ when `TEKSECURE` completes:

```
# Bit 0 of the DESE (Device Event Status Enable Register)
# enables OPC to be reported to the SESR (Standard Event
# Status Register)
DESE 255
# Bit 0 of the ESER (Event Status Enable Register)
# enables OPC to be summarized in the ESB (Event Status #
Bit) of the SBR (Status Byte Register)
*ESE 255
# Bit 5 of the SRE (Service Request Enable Register)
enables
# the generation of SRQ when the ESB bit of the SBR
becomes # TRUE
*SRE 32
TEKSECURE;*OPC
```

When the `TEKSECURE` operation has completed, the OPC bit of the SESR will be TRUE and SRQ will have been generated.

TIME

Sets or returns the time that the oscilloscope displays.

Group Miscellaneous

Syntax

```
TIME <QString>
TIME?
```

Related Commands

DATE

Arguments

`<QString>` is a time in the form "hh:mm:ss" where hh refers to a two-digit hour number, mm refers to a two-digit minute number from 00 to 59, and ss refers to a two-digit second number from 00 to 59.

Examples

`TIME "14:00:00"` sets the time to exactly 2:00 p.m.

`TIME?` might return `:TIME "14:05:17"` indicating that the current time is set to 2:05 p.m. and 17 seconds.

TOTaluptime? (Query Only)

This command returns the total number of hours that the oscilloscope has been powered on since the nonvolatile memory was last programmed (usually since the initial manufacturing process).

Group

Miscellaneous

Syntax

```
TOTaluptime?
```

*TRG (No Query Form)

Performs a group execute trigger on commands defined by *DDT.

Group

Miscellaneous

Syntax

```
*TRG
```

Related Commands

*DDT

Examples

`*TRG` immediately executes all commands that have been defined by *DDT.

TRIGger (No Query Form)

Forces a trigger event to occur.

Group Trigger

Syntax

```
TRIGger FORCe
TRIGger?
```

Arguments `FORCe` creates a trigger event. If TRIGger:STATE is set to READy, the acquisition will complete. Otherwise, this command will be ignored.

Examples `TRIGGER FORCE` forces a trigger event to occur.

TRIGger:A

Sets the A trigger level automatically to 50% of the range of the minimum and maximum values of the trigger input signal. The query returns current A trigger parameters. The trigger level is the voltage threshold through which the trigger source signal must pass to generate a trigger event. This command works for the following cases: Edge Trigger (when source is Not Line), Logic Trigger (when Clock Source is not Off or Logic Pattern is Don't Care), and Pulse Width Trigger.

Group Trigger

Syntax

```
TRIGger:A SETLevel
TRIGger:A?
```

Related Commands TRIGger:A:EDGE?, TRIGger:A:LOGIc?, TRIGger:A:PULse?

Arguments `SETLevel` sets the A trigger level to 50% of the range of the minimum and maximum values of the trigger input signal.

Examples `TRIGGER:A SETLEVEL` sets the A trigger level to 50% of the range of the minimum and maximum values of the trigger input signal.

`TRIGGER:A?` might return a long response with A trigger parameters, some of which could be as follows: `:TRIGGER:A:MODE AUTO;TYPE EDGE;LEVEL 20.0000E-3;LEVEL:CH1 20.0000E-3;CH2 0.0000; CH3 0.0000;CH4 0.0000;:TRIGGER:A:UPPERTHRESHOLD:CH1 1.4000;CH2 800.0000E-3;CH3 8 00.0000E-3;CH4`

```
800.0000E-3;:TRIGGER:A:LOWERTHRESHOLD:CH1 20.0000E-3;CH2
0.0000;CH3 0.0000;CH4 0.0000;:TRIGGER:A:HOLDOFF:TIME
20.0000E-9;:TRIGGER:A:EDGE:SOURCE CH1;COUPLING
DC;SLOPE RISE;:TRIGGER:A:LOGIC:CLASS SETHOLD;FUNCTION
AND;THRESHOLD: CH1 20.0000E-3;CH2 0.0000;CH3
0.0000;CH4 0.0000;:TRIGGER:A:LOGIC:INPUT:CH1
X;CH2 X;CH3 X;CH4 X;CLOCK:SOURCE NONE;EDGE
RISE;:TRIGGER:A:LOGIC:PATTERN:INPUT:CH1 X; CH2
X;CH3 X;CH4 X;:TRIGGER:A:LOGIC:PATTERN:WHEN
TRUE;WHEN:LESSLIMIT 4.0000E-9;MO RELIMIT
4.0000E-9;:TRIGGER:A:SETHOLD:CLOCK:SOURCE CH1;EDGE
RISE;THRESHOLD 20.000 0E-3;:TRIGGER:A:SETHOLD:DATA:SOURCE
CH2;THRESHOLD 0.0000;:TRIGGER:A:SETHOLD:HOLD TIME
4.0000E-9;SETTIME 4.0000E-9;:TRIGGER:A:PULSE:CLASS
TRAnsITION;:TRIGGER:A:PULSEWIDTH:SOURCE
CH1;POLARITY POSITIVE;WHEN LESSTHAN;WIDTH
4.0000E-9;:TRIGGER:A:RUNT:SOURCE CH1;POLARITY POSITIVE;WHEN
OCCURS;WIDTH 4.0000E-9;:TRIGGER:A:TRAnsITION:SOURCE
CH1;POLARITY POSITIVE;WHEN SLOWER;DELTATIME
4.0000E-9;:TRIGGER:A:VIDEO :POLARITY POSITIVE;SOURCE
CH1;STANDARD NTSC;SYNC ALLLINES;HOLDOFF:FIELD 0.0000;:
TRIGGER:A:VIDEO:CUSTOM:FORMAT PROGRESSIVE;SCAN
RATE15K;:TRIGGER:A:VIDEO:LINE 1;H DTV:FORMAT
HD1080I60;:TRIGGER:A:BUS:SOURCE B1;B1:I2C:CONDITION
START;DATA:VALUE "XXXXXXXX";SIZE 1;START 0.0000;DIRECTION
NOCARE;:TRIGGER:A:BUS:B1:I2C:ADDRESS:MODE
ADDR7;TYPE USER;VALUE
"XXXXXXX";:TRIGGER:A:BUS:B1:SPI:CONDITION MOSI;DATA:OUT
:VALUE "XXXXXXXX";:TRIGGER:A:BUS:B1:SPI:DATA:IN:VALUE
"XXXXXXXX";:TRIGGER:A:BUS: B1:SPI:DATA:SIZE 1;START 0.0000;:
```

TRIGger:A:BUS

Sets or returns the trigger type: I2C, CAN, SPI, and RS232. There are up to two serial buses, B1, B2, depending on your instrument model. Each can be independently set to one of the serial trigger types. The serial parameters related to the trigger are broken into two sections: Trigger:A:SERIAL xxx, consisting of parameters the user will change frequently, and BUS:B1:xxx, consisting of parameters the user will specify once (bus definition).

Conditions Requires a DPO2AUTO, DPO2EMBD, or DPO2COMP application module.

Group Trigger

Syntax

```
TRIGger:A:BUS {I2C|SPI|CAN|RS232}
TRIGger:A:BUS?
```

Arguments

`I2C` specifies the Inter-IC bus.

`SPI` specifies the Serial Peripheral Interface bus.

`CAN` specifies the Controller Area Network bus.

TRIGger:A:BUS:B<x>:CAN:CONDition

Sets or returns the CAN trigger condition for bus <x>, where x is the bus number.

Conditions

Requires a DPO2AUTO application module.

Group

Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN:CONDition
{SOF|FRAMEtype|IDentifier|DATA|IDANDDATA|EOF|ACKMISS}
TRIGger:A:BUS:B<x>:CAN:CONDition?
```

Arguments

`SOF` enables triggering on the start of frame.

`FRAMEtype` enables triggering on the type of frame.

`IDentifier` enables triggering on a matching identifier.

`DATA` enables triggering on matching data.

`IDANDDATA` enables triggering on a matching identifier and matching data.

`EOF` enables triggering on the end of frame.

`ACKMISS` enables triggering on a missing acknowledge.

Examples

`TRIGGER:A:BUS:B1:CAN:CONDITION?` might return :TRIGGER:A:BUS:B1:CAN:CONDITION EOF indicating an end of file condition.

`TRIGGER:A:BUS:B1:CAN:CONDITION DATA` enables triggering on matching CAN data.

TRIGger:A:BUS:B<x>:CAN:DATa:DIRection

Sets or returns the CAN trigger data direction to be valid on a Read, Write, or Either condition for bus <x>, where x is the bus number. This applies only, if the trigger condition is ID.

Conditions Requires a DPO2AUTO application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN:DATa:DIRection {READ|WRITE|NOCARE}
TRIGger:A:BUS:B<x>:CAN:DATa:DIRection?
```

Arguments

`READ` sets the CAN data direction to READ.

`WRITE` sets the CAN data direction to WRITE.

`NOCARE` sets the CAN data direction to either.

Examples `TRIGGER:A:BUS:B1:CAN:DATA:DIRECTION WRITE` sets the CAN data direction to Write.

TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier

Sets or returns the CAN data qualifier for bus <x>, where x is the bus number. This applies only, if the trigger condition is IDANDDATA or DATA.

Conditions Requires a DPO2AUTO application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier
{LESSThan|MOREThan|EQUal|UNEQual|LESSEQual|MOREEQual}
TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier?
```

Arguments

`LESSThan` sets the oscilloscope to trigger when the data is less than the qualifier value.

`MOREThan` sets the oscilloscope to trigger when the data is more than the qualifier value.

`EQUal` sets the oscilloscope to trigger when the data is equal to the qualifier value.

`UNEQual` sets the oscilloscope to trigger when the data is not equal to the qualifier value.

`LESSEQual` sets the oscilloscope to trigger when the data is less than or equal to the qualifier value.

`MOREEQual` sets the oscilloscope to trigger when the data is more than or equal to the qualifier value.

Examples

`TRIGGER:A:BUS:B1:CAN:DATA:QUALIFIER LESSTHAN` sets the oscilloscope to trigger when the data is less than the qualifier value.

`TRIGGER:A:BUS:B1:CAN:DATA:QUALIFIER?` might return `:TRIGGER:A:BUS:B1:CAN:DATA:QUALIFIER THAN`, indicating that the oscilloscope is set to trigger when the data is than the qualifier value.

TRIGger:A:BUS:B<x>:CAN:DATa:SIZe

Sets or returns the length of the data string in bytes for a CAN trigger if the condition is DATA or IDANDDATA. Applies to bus <x>, where x is the bus number.

Conditions

This command requires a DPO2AUTO application module.

Group

Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN:DATa:SIZe <NR1>
TRIGger:A:BUS:B<x>:CAN:DATa:SIZe?
```

Arguments

`<NR1>` is the length of the data string in bytes.

TRIGger:A:BUS:B<x>:CAN:DATa:VALue

Sets or returns the binary data string to be used for a CAN trigger if the trigger condition is ID or IDANDDATA. Applies to bus <x>, where x is the bus number.

Conditions

This command requires a DPO2AUTO application module.

Group

Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN:DATa:VALue <QString>
TRIGger:A:BUS:B<x>:CAN:DATa:VALue?
```

Arguments

`<QString>` is the CAN data value in binary format. The only allowed characters in the `QString` are 0, 1, and X.

Examples

`TRIGGER:A:BUS:B1:CAN:DATA:VALUE 1011` sets the CAN data value to 1011.

TRIGger:A:BUS:B<x>:CAN:FRAMEtype

Sets or returns the frame type for a CAN FRAMEType trigger. Applies to bus <x>, where x is the bus number.

Conditions

This command requires a DPO2AUTO application module. This command is only valid when the TRIGger:A:BUS:B<x>:CAN:CONDition is FRAMETYPE.

Group

Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN:FRAMEtype
{DATA|REMote|ERRor|OVERLoad}
TRIGger:A:BUS:B<x>:CAN:FRAMEtype?
```

Arguments

`DATA` specifies a data frame type.

`REMote` specifies a remote frame type.

`ERRor` specifies an error frame type.

`OVERLold` specifies an overload frame type.

Examples

`TRIGGER:A:BUS:B1:CAN:FRAMETYPE DATA` sets the CAN trigger frame type to DATA.

TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe

Sets or returns the CAN addressing mode for bus <x>, where x is the bus number. Use this command to do the following:

- Trigger on ID
- Trigger in IDANDDATA

Conditions This command requires a DPO2AUTO application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe
{STandard|EXTended}
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe?
```

Arguments `STandard` specifies the standard addressing mode.

`EXTended` specifies the extended addressing mode.

TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue

Sets or returns the binary address string used for a CAN trigger if the trigger condition is ID or IDANDDATA. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2AUTO application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue <QString>
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue?
```

Arguments `<QString>` is up to 29 bits specifying the binary CAN identifier value. The only allowed characters in the `QString` are 0, 1, and X.

Examples `TRIGGER:A:BUS:B1:CAN:IDENTIFIER:VALUE 1011` sets the CAN trigger identifier value to 1011.

TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe

Sets or returns the I²C address mode to 7 or 10-bit. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe {ADDR7|ADDR10}
TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe?
```

Arguments

`ADDR7` specifies the 7-bit I²C address mode.

`ADDR10` specifies the 10-bit I²C address mode.

Examples

`TRIGGER:A:BUS:B1:I2C:ADDRESS:MODE ADDR10` sets the I2C address mode to 10-bit.

TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe

Sets or returns the I²C address type. The only supported address type is USER. Applies to bus <x>, where x is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Trigger

Syntax

```
TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe
{GENeralcall|STARtbyte|HSmode|EEPROM|USER}
TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe?
```

Arguments

`GENeralcall` specifies a general call address.

`STARtbyte` specifies a start byte address.

`HSmode` specifies a high-speed mode address.

`EEPROM` specifies an EEPROM address.

`USER` specifies a user address.

TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue

Sets or returns the binary address string used for the I²C trigger if the trigger condition is ADDRESS or ADDRANDDATA. Applies to bus <x>, where x is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Trigger

Syntax

```
TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue <QString>
TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue?
```

Arguments

`<QString>` is up to 7 or 10-bits depending on the address mode that specifies the address. The only allowed characters in the `QString` are 0, 1, and X.

Examples

`TRIGGER:A:BUS:B1:I2C:ADDRESS:VALUE 1011` sets the I²C address value to XXX1011.

TRIGger:A:BUS:B<x>:I2C:CONDition

Sets or returns the trigger condition for an I²C trigger. Applies to bus <x>, where x is the bus number.

Conditions

This command requires a DPO2EMBD application module.

Group

Trigger

Syntax

```
TRIGger:A:BUS:B<x>:I2C:CONDition
{STARt|STOP|REPEATstart|ACKMISS|ADDRess|DATA|ADDRANDDATA}
TRIGger:A:BUS:B<x>:I2C:CONDition?
```

Arguments

`STARt` specifies a search based on start condition.

`STOP` specifies a search based on stop condition.

`REPEATstart` specifies a search based on repeat of start condition.

`ACKMISS` specifies a search based on missing acknowledgement condition.

`ADDRess` specifies a search based on address.

`DATA` specifies a search based on data.

`ADDRANDDATA` specifies a search based on address and data.

Examples

`TRIGGER:A:BUS:B1:I2C:CONDITION START` specifies start as the I²C trigger condition.

TRIGger:A:BUS:B<x>:I2C:DATa:DIRection

Sets or returns the I²C trigger type to be valid on a Read, Write, or Either condition. Read or write is indicated by the R/W bit in the I²C protocol. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:I2C:DATa:DIRection {READ|WRITE|NOCARE}
TRIGger:A:BUS:B<x>:I2C:DATa:DIRection?
```

Arguments

`READ` specifies read as the data direction.

`WRITE` specifies write as the data direction.

`NOCARE` specifies either as the data direction.

Examples `TRIGGER:A:BUS:B1:I2C:DATA:DIRECTION WRITE` specifies write as the I²C data direction.

TRIGger:A:BUS:B<x>:I2C:DATa:SIZe

Sets or returns the length of the data string in bytes to be used for an I2C trigger if the trigger condition is DATA or ADDRANDDATA. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:I2C:DATa:SIZe <NR1>
TRIGger:A:BUS:B<x>:I2C:DATa:SIZe?
```

Arguments `<NR1>` is the length of the data string in bytes.

TRIGger:A:BUS:B<x>:I2C:DATa:VALue

Sets or returns the binary data string used for I2C triggering if the trigger condition is DATA or ADDRANDDATA. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:I2C:DATa:VALue <QString>
TRIGger:A:BUS:B<x>:I2C:DATa:VALue?
```

Arguments `<QString>` is the binary data string, where the number of bits is 8 times the number of bytes specified. The only allowed characters in the string are 0, 1, and X.

TRIGger:A:BUS:B<x>:LIN:CONDition

Sets or returns the trigger condition for LIN.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
TRIGger:A:BUS:B<x>:LIN:CONDition
{SYNCField|IDentifier|DATA|IDANDDATA|WAKEup|SLEEP|ERROR}
TRIGger:A:BUS:B<x>:LIN:CONDition?
```

Arguments `SYNCField` sets the LIN trigger condition to sync field.

`IDentifier` sets the LIN trigger condition to identifier.

`DATA` sets the LIN trigger condition to data.

`IDANDDATA` sets the LIN trigger condition to ID and data.

`WAKEup` sets the LIN trigger condition to wake up.

`SLEEP` sets the LIN trigger condition to sleep.

`ERROR` sets the LIN trigger condition to error.

Examples

`TRIGGER:A:BUS:B1:LIN:CONDITION ERROR` sets the LIN trigger condition to error.

`TRIGGER:A:BUS:B1:LIN:CONDITION?` might return `TRIGGER:A:BUS:B1:LIN:CONDITION SYNCFIELD` indicating that the LIN trigger condition is sync field.

TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue

Sets or returns the binary data string to be used for LIN trigger if trigger condition is ID or IDANDDATA.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue <QString>
TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue?
```

Arguments

`<QString>` is a quoted string that is the binary data string used for LIN trigger if the trigger condition is ID or IDANDDATA.

Examples

`TRIGGER:A:BUS:B1:LIN:DATA:HIVALUE "11001010"` sets the high value to `11001010`.

`TRIGGER:A:BUS:B1:LIN:DATA:HIVALUE?` might return `TRIGGER:A:BUS:B1:LIN:DATA:HIVALUE "XXXXXXXX"` indicating that the high value is don't care.

TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier

Sets or returns the LIN data qualifier. This only applies if the trigger condition is IDANDDATA or DATA.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier
{LESSThan|MOREThan|EQUal|UNEQual|LESSEQual|MOREEQual|INrange|OUTrang
TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier?
```

Arguments

`LESSThan` sets the LIN data qualifier to less than.

`MORETh an` sets the LIN data qualifier to greater than.

`EQUal` sets the LIN data qualifier to equal.

`UNEQual` sets the LIN data qualifier to not equal.

`LESSEQual` sets the LIN data qualifier to less than or equal.

`MOREEQual` sets the LIN data qualifier to greater than or equal.

`INrange` sets the LIN data qualifier to in range.

`OUTrange` sets the LIN data qualifier to out of range.

Examples

`TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER OUTRANGE` sets the data qualifier to out of range.

`TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER?` might return `TRIGGER:A:BUS:B1:LIN:DATA:QUALIFIER EQUAL` indicating that the data qualifier is set to equal.

TRIGger:A:BUS:B<x>:LIN:DATa:SIZe

Sets or returns the length of the data string in bytes to be used for LIN trigger.

Conditions

This command requires a DPO2AUTO application module.

Group

Bus

Syntax

```
TRIGger:A:BUS:B<x>:LIN:DATa:SIZe <NR1>
TRIGger:A:BUS:B<x>:LIN:DATa:SIZe?
```

Arguments

`<NR1>` is the size of the data string in bytes.

Examples

`TRIGGER:A:BUS:B1:LIN:DATA:SIZE 8` sets the data size to 8 bytes.

`TRIGGER:A:BUS:B1:LIN:DATA:SIZE?` might return `TRIGGER:A:BUS:B1:LIN:DATA:SIZE 1` indicating that the data size is 1 byte.

TRIGger:A:BUS:B<x>:LIN:DATa:VALue

Sets or returns the binary data string to be used for LIN trigger condition if trigger condition is ID or IDANDDATA.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
TRIGger:A:BUS:B<x>:LIN:DATa:VALue <QString>
TRIGger:A:BUS:B<x>:LIN:DATa:VALue?
```

Arguments `<QString>` is a quoted string that is the LIN trigger data value.

Examples `TRIGGER:A:BUS:B1:LIN:DATA:VALUE “11001101”`sets the data value to 11001101.

`TRIGGER:A:BUS:B1:LIN:DATA:VALUE?` might return `TRIGGER:A:BUS:B1:LIN:DATA:VALUE "XXXXXXXX"` indicating that the data value is don't care.

TRIGger:A:BUS:B<x>:LIN:ERRTYPE

Sets or returns the error type be used for LIN trigger.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
TRIGger:A:BUS:B<x>:LIN:ERRTYPE {SYNC|PARity|CHecksum}
TRIGger:A:BUS:B<x>:LIN:ERRTYPE?
```

Arguments `SYNC` sets the LIN error type to SYNC.

`PARity` sets the LIN error type to parity.

`CHecksum` sets the LIN error type to checksum.

Examples `TRIGGER:A:BUS:B1:LIN:ERRTYPE CHECKSUM` sets the LIN error type to checksum.

`TRIGGER:A:BUS:B1:LIN:ERRTYPE?` might return `TRIGGER:A:BUS:B1:LIN:ERRTYPE SYNC` indicating that the LIN error type is SYNC.

TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue

Sets or returns the binary address string used for LIN trigger if the trigger condition is ID or IDANDDATA.

Conditions This command requires a DPO2AUTO application module.

Group Bus

Syntax

```
TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue <QString>
TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue?
```

Arguments `<QString>` is the binary address string used for LIN trigger if the trigger condition is ID or IDANDDATA.

Examples `TRIGGER:A:BUS:B1:LIN:IDENTIFIER:VALUE "110010"` sets the identifier value to 110010.

`TRIGGER:A:BUS:B1:LIN:IDENTIFIER:VALUE?` might return `TRIGGER:A:BUS:B1:LIN:IDENTIFIER:VALUE "XXXXXX"` indicating that the identifier value is XXXXXX.

TRIGger:A:BUS:B<x>:PARallel:VALue

Sets or returns the binary data string to be used for a parallel trigger. Applies to bus <x>, where x is the bus number.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:PARallel:VALue <QString>
TRIGger:A:BUS:B<x>:PARallel:VALue?
```

Arguments `<QString>` is the binary data string.

TRIGger:A:BUS:B<x>:RS232C:CONDition

Sets or returns the condition for a RS232C trigger, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:RS232C:CONDition
{RXSTArt|RXDATA|RXENDPacket|RXPARity|TXSTArt|
TXDATA|TXENDPacket|TXPARity|}
TRIGger:A:BUS:B<x>:RS232C:CONDition?
```

Arguments

`RXSTArt` specifies a search based on the RX Start Bit.

`RXDATA` specifies a search based on RX Data.

`RXENDPacket` specifies a search based on the RX End of Packet condition.

`RXPARity` specifies a search based on the RX parity.

`TXSTArt` specifies a search base on the TX Start Bit.

`TXDATA` specifies a search based on TX Data.

`TXENDPacket` specifies a search based on the TX End of Packet condition.

`TXPARity` specifies a search based on the TX parity.

TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe

Sets or returns the length of the data string in Bytes for a RS232 Trigger if the trigger condition is RXDATA. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe <NR1>
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe?
```

Arguments `<NR1>` is the length of the data string in bytes.

TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue

Sets or returns the binary data string for a RS232 trigger if the trigger condition involves RX. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue?
```

Arguments `<Qstring>` is the binary data string to be used for the trigger.

TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe

Sets or returns the length of the data string for a RS232 trigger if the trigger condition is TXDATA. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe <NR1>
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe?
```

Arguments `<NR1>` is the length of the data string in Bytes.

TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue

Sets or returns the binary data string for a RS232 trigger if the condition involves TX. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2COMP application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue?
```

Arguments `<Qstring>` is the binary data string to be used for the trigger.

TRIGger:A:BUS:B<x>:SPI:CONDition

Sets or returns the trigger condition for a SPI trigger. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:SPI:CONDition
{SS|STARTofframe|MISO|MOSI|MISOMOSI}
TRIGger:A:BUS:B<x>:SPI:CONDition?
```

Arguments

`SS` specifies the Slave Selection condition.

`STARTofframe` specifies the start of frame condition.

`MISO` specifies the Master-In Slave-Out condition.

`MOSI` specifies the Master-Out Slave-In condition.

`MISOMOSI` specifies the Master-In Slave-Out and Master-Out Slave-In conditions.

TRIGger:A:BUS:B<x>:SPI:DATa{:IN|:MISO}:VALue

Sets or returns the binary data string to be used for a SPI trigger if the trigger condition is MISO or MISOMOSI. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:SPI:DATa{:IN|:MISO}:VALue <QString>
TRIGger:A:BUS:B<x>:SPI:DATa{:IN|:MISO}:VALue?
```

Arguments `<QString>` is the binary data string, where the number of bits is 8 times the number of bytes specified. The only allowed characters in the string are 0, 1, and X.

TRIGger:A:BUS:B<x>:SPI:DATa{:OUT|:MOSI}:VALue

Sets or returns the binary data string to be used for a SPI trigger if the trigger condition is MOSI or MISOMOSI. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:SPI:DATa{:OUT|:MOSI}:VALue <QString>
TRIGger:A:BUS:B<x>:SPI:DATa{:OUT|:MOSI}:VALue?
```

Arguments `<QString>` is the binary data string with the number of bits specified by the TRIGger:A:BUS:B<x>:SPI:DATa:SIZe command. The only allowed characters in the `QString` are 0, 1, and X.

TRIGger:A:BUS:B<x>:SPI:DATa:SIZe

Sets or returns the length of the data string to be used for a SPI trigger if the trigger condition is MISO, MOSI, or MISOMOSI. Applies to bus <x>, where x is the bus number.

Conditions This command requires a DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:B<x>:SPI:DATa:SIZe <NR1>
TRIGger:A:BUS:B<x>:SPI:DATa:SIZe?
```

Arguments `<NR1>` is the length of the data string in bytes.

TRIGger:A:BUS:SOUrce

Sets or returns the source for a Serial bus trigger.

Conditions This command requires a DPO2AUTO or DPO2EMBD application module.

Group Trigger

Syntax

```
TRIGger:A:BUS:SOUrce {B1|B2}
TRIGger:A:BUS:SOUrce?
```

Arguments

`B1` specifies the Bus 1 source.

`B2` specifies the Bus 2 source.

TRIGger:A:EDGE? (Query Only)

Returns the trigger source, coupling, and slope for the A edge trigger.

Group

Trigger

Syntax

```
TRIGger:A:EDGE?
```

Related Commands

TRIGger:A:PULse?, TRIGger:A:LOGIc?

Examples

`TRIGGER:A:EDGE?` might return `:TRIGGER:A:EDGE:SOURCE CH1;COUPLING DC; SLOPE RISE` indicating that the trigger source, coupling, and slope for the A edge trigger.

TRIGger:A:EDGE:COUPling

Sets or returns the type of coupling for the A edge trigger.

Group

Trigger

Syntax

```
TRIGger:A:EDGE:COUPling {DC|HFRej|LFRej|NOISErej}
TRIGger:A:EDGE:COUPling?
```

Related Commands

TRIGger:A:EDGE:SOUrce, TRIGger:A:EDGE:SLOpe

Arguments

`DC` specifies DC trigger coupling, which passes all input signals to the trigger circuitry.

`HFRej` specifies high-frequency rejection coupling, which attenuates signals above 50 kHz before passing the signals to the trigger circuitry.

`LFRej` specifies low-frequency rejection coupling, which attenuates signals below 50 kHz before passing the signals to the trigger circuitry.

`NOISErej` specifies noise-rejection coupling, which provides stable triggering by increasing the trigger hysteresis. Increased hysteresis reduces the trigger sensitivity to noise but may require greater trigger signal amplitude.

Examples

`TRIGGER:A:EDGE:COUPLING DC` sets the A edge trigger coupling to DC.

`TRIGGER:A:EDGE:COUPLING?` might return `:TRIGGER:A:EDGE:COUPLING DC` indicating that the A edge trigger coupling is set to DC.

TRIGger:A:EDGE:SLOpe

Sets or returns the slope for the A edge trigger.

Group

Trigger

Syntax

```
TRIGger:A:EDGE:SLOpe {RISe|FALL}
TRIGger:A:EDGE:SLOpe?
```

Related Commands

TRIGger:A:EDGE:SOUrce, TRIGger:A:EDGE:COUPling

Arguments

`RISe` specifies to trigger on the rising or positive edge of a signal.

`FALL` specifies to trigger on the falling or negative edge of a signal.

Examples

`TRIGGER:A:EDGE:SLOPE RISE` sets the A edge trigger slope to positive, which triggers on the rising edge of the signal.

`TRIGGER:A:EDGE:SLOPE?` might return `:TRIGGER:A:EDGE:SLOPE FALL` indicating that the A edge trigger slope is negative.

TRIGger:A:EDGE:SOUrce

Sets or returns the source for the A edge trigger.

Group

Trigger

Syntax

```
TRIGger:A:EDGE:SOUrce {{CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|
D14|D15|EXT|LINE|AUX}
TRIGger:A:EDGE:SOUrce?
```

Related Commands TRIGger:A:EDGE:SLOpe, TRIGger:A:EDGE:COUPling

Arguments `CH1–CH4` specifies an analog input channel as the A edge trigger source.

`D0–D15` specifies a digital input channel as the A edge trigger source.

`EXT` specifies an external trigger using the Aux In connector located on the front panel of the oscilloscope.

`LINE` specifies the AC line as the trigger source.

`AUX` specifies the Auxiliary Input as the trigger source (if available on your oscilloscope).

Examples `TRIGGER:A:EDGE:SOURCE CH1` sets channel 1 as the A edge trigger source.

`TRIGGER:A:EDGE:SOURCE?` might return `:TRIGGER:A:EDGE:SOURCE CH1` indicating that channel 1 is the A edge trigger source.

TRIGger:A:HOLDoff? (Query Only)

Returns the A trigger holdoff parameters. These parameters specify the time period during which the trigger circuitry is not looking to generate a trigger event.

Group Trigger

Syntax

```
TRIGger:A:HOLDoff?
```

Related Commands TRIGger:A:HOLDoff:TIMe

Examples `TRIGGER:A:HOLDOFF?` might return `:TRIGGER:A:HOLDOFF:TIME 900.0000E-09;BY DEFAULT` indicating that the A edge trigger holdoff time (by default) is 900 ns.

TRIGger:A:HOLDoff:TIMe

Sets or returns the A trigger holdoff time.

Group Trigger

Syntax

```
TRIGger:A:HOLDoff:TIMe <NR3>
TRIGger:A:HOLDoff:TIMe?
```

Arguments <NR3> specifies the holdoff time in seconds. The range is from 20 ns through 8.0 s.

Examples TRIGGER:A:HOLDOFF:TIME ? might return :TRIGGER:A:HOLDOFFTIME 1.2000E-06 indicating that the A trigger holdoff time is set to 1.2 µs.

TRIGGER:A:HOLDOFF:TIME 10 sets the A trigger holdoff time to 10 s.

TRIGger:A:LEVel

Sets or returns the trigger level for the A trigger.

Group Trigger

Syntax

```
TRIGger:A:LEVel {ECL|TTL|<NR3>}
TRIGger:A:LEVel?
```

Arguments ECL specifies a preset ECL high level of –1.3V.

TTL specifies a preset TTL high level of 1.4V.

<NR3> specifies the trigger level in user units (usually volts).

Examples TRIGGER:A:LEVEL? might return :TRIGGER:A:LEVel 1.3000E+00 indicating that the A edge trigger is set to 1.3 V.

TRIGGER:A:LEVEL TTL sets the A edge trigger to TTL high level, which is 1.4 V.

TRIGger:A:LEVel:AUXin

Sets or returns the trigger level for the AUXIN port.

Group Trigger

Syntax

```
TRIGger:A:LEVel:AUXin {<NR3>|ECL|TTL}
TRIGger:A:LEVel:AUXin?
```

Arguments <NR3> specifies the trigger level, in volts.

ECL specifies a preset ECL trigger level of –1.3 V.

TTL specifies a preset TTL trigger level of 1.4 V.

Examples

`TRIGGER:A:LEVEL:AUXIN ECL` sets the auxiliary input trigger level to -1.3 volts.

`TRIGGER:A:LEVEL:AUXIN?` might return `TRIGGER:A:LEVEL:AUXIN 0.0E+0` indicating that the auxiliary input trigger level is 0.0 volts.

TRIGger:A:LEVel:CH<x>

Sets or returns the trigger level for the specified channel. Each channel can have an independent level.

Group

Trigger

Syntax

```
TRIGger:A:LEVel:CH<x> {<NR3>|TTL|ECL}
TRIGger:A:LEVel:CH<x>?
```

Arguments

`<NR3>` specifies the trigger level in user units (usually volts).

`TTL` specifies a preset `TTL` high level of 1.4 V.

`ECL` specifies a preset `ECL` high level of –1.3 V.

Examples

`TRIGGER:A:LEVEL:CH2?` might return `:TRIGGER:A:LEVEL:CH2 1.3000E+00` indicating that the A edge trigger is set to 1.3 V for channel 2.

`TRIGGER:A:LEVEL:CH3 TTL` sets the A edge trigger to TTL high level for channel 3.

TRIGger:A:LEVel:D<x>

Sets or returns the trigger level for the specified digital channel <x>, where x is the channel number. The digital channels do not have independent trigger levels. Channels D0–D7 share one common trigger level and D8–D15 share another common trigger level. For example: Changing the trigger level for D3 changes the levels for D0–D7.

Group

Trigger

Syntax

```
TRIGger:A:LEVel:D<x> {<NR3>|ECL|TTL}
TRIGger:A:LEVel:D<x>?
```

Arguments

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

`<NR3>` specifies the trigger level in volts.

TRIGger:A:LOGIc? (Query Only)

Returns all of the A logic trigger parameters.

Group

Trigger

Syntax

```
TRIGger:A:LOGIc?
```

Related Commands

TRIGger:A:LOGIc:CLAss

Examples

`TRIGGER:A:LOGIC?` might return `:TRIGGER:A:LOGIC:CLASS SETHOLD;FUNCTION AND;THRESHOLD:CH1 20.0000E-3;CH2 0.0000; CH3 0.0000;CH4 0.0000;:TRIGGER:A:LOGIC:INPUT:CH1 X;CH2 X;CH3 X;CH4 X;CLOCK:SOURCE NONE;EDGE RISE;:TRIGGER:A:LOGIC:PATTERN:INPUT:CH1 X;CH2 X;CH3 X;CH4 X;:TRIGGER :A:LOGIC:PATTERN:WHEN TRUE;WHEN:LESSLIMIT 4.0000E-9;LIMIT 4.0000E-9;:TRIGGER:A:LOGIC:PATTERN:DELTATIME 4.0000E-9`

TRIGger:A:LOGIc:CLAss

Sets or returns the class of the Logic Trigger. This command is used in conjunction with the TRIGger:A:TYPe command.

Group

Trigger

Syntax

```
TRIGger:A:LOGIc:CLAss {LOGIC|SETHold}
TRIGger:A:LOGIc:CLAss?
```

Related Commands

TRIGger:A:TYPe, TRIGger:A:PULse:CLAss

Arguments

`LOGIC` sets the oscilloscope to trigger on logical combinations of the channels.

When the TRIGger:A:LOGIc:INPut:CLOCk:SOUrce is NONE, `LOGIC` sets the oscilloscope to trigger when the specified logical combinations of channels 1, 2,

3, and 4 are met on four-channel oscilloscopes. On two-channel oscilloscopes, only channel 1 and channel 2 are available.

When the TRIGger:A:LOGIc:INPut:CLOCk:SOUrce is set to one of the channels, `LOGIC` sets the oscilloscope to trigger when the specified logical combinations of the remaining channels is true during a transition on the clock channel.

`SETHold` sets the oscilloscope to trigger on setup and hold violations between a data source and a clock source. Use one channel input as the clock signal and a second channel input as the data input. The clocking and data levels are used to determine if a clock or data transition has occurred.

Examples

`TRIGGER:A:LOGIC:CLASS?` might return `:TRIGGER:A:LOGIC:CLASS LOGIC`

`TRIGGER:A:LOGIC:CLASS LOGIC` sets the trigger A logic class to LOGIc, which causes the oscilloscope to trigger when the specified logical combinations of channels 1, 2, 3, and 4 are met.

TRIGger:A:LOGIc:FUNCtion

Sets or returns the logical combination of the input channels for the A pattern and A state logic triggers.

Group

Trigger

Syntax

```
TRIGger:A:LOGIc:FUNCtion {AND|NANd}
TRIGger:A:LOGIc:FUNCtion?
```

Related Commands

TRIGger:A:LOGIc:INPut:CH<x>

Arguments

`AND` specifies to trigger if all conditions are true.

`NANd` specifies to trigger if any of the conditions is false.

Examples

`TRIGGER:A:LOGIC:FUNCTION?` might return `:TRIGGER:A:LOGIC:FUNCTION NAND` which indicates that the oscilloscope will trigger if the AND logic conditions are false.

`TRIGGER:A:LOGIC:FUNCTION AND` sets the logical combination of channels to be true when all conditions are true.

TRIGger:A:LOGIc:INPut? (Query Only)

Returns the logic input values for all channels. If a clock channel is defined, it returns the clock source and edge.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:INPut?
```

Examples `TRIGGER:A:LOGIC:INPUT?` might return `:TRIGGER:A:LOGIC:INPUT:CH1 HIGH;CH2 X;CH3 X` indicating that a logic high is expected on channel 1 while channel 2 and channel three are “don't care.”

TRIGger:A:LOGIc:INPut:CH<x>

Sets or returns the logical input condition for the channel specified by <x>.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:INPut:CH<x> {HIGH|LOW|X}
TRIGger:A:LOGIc:INPut:CH<x>?
```

Arguments `HIGH` specifies the logic high.

`LOW` specifies the logic low.

`X` specifies a "don't care" state.

Examples `TRIGGER:A:LOGIC:INPUT:CH1?` might return `:TRIGGER:LOGIC:INPUT:CH1 X` indicating that the setting for the A logic trigger input to channel 1 does not matter.

`TRIGGER:A:LOGIC:INPUT:CH2 HIGH` sets the A logic trigger input to logic HIGH for channel 2.

TRIGger:A:LOGIc:INPut:CLOCk:EDGE

Sets the polarity of the clock channel.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:INPut:CLOCk:EDGE {FALL|RISe|EITher}
TRIGger:A:LOGIc:INPut:CLOCk:EDGE?
```

Arguments

`RISe` specifies to trigger on the rising or positive edge of a signal.

`FALL` specifies to trigger on the falling or negative edge of a signal.

`EITher` specifies to trigger either on the falling or rising edge of a signal.

TRIGger:A:LOGIc:INPut:CLOCk:SOUrce

Sets or returns the channel to use as the clock source. The clock can be selected as `NONE`. A selection of `None` implies pattern trigger. Any other selection implies state trigger.

Group

Trigger

Syntax

```
TRIGger:A:LOGIc:INPut:CLOCk:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15|NONE}
TRIGger:A:LOGIc:INPut:CLOCk:SOUrce?
```

Arguments

`CH1-CH4` specifies the analog input channel source.

`D0-D15` specifies the digital input channel source.

`NONE` specifies a Pattern trigger.

TRIGger:A:LOGIc:INPut:D<x>

Sets or returns the logic pattern for a trigger on digital channel <x>, where x is the channel number.

Group

Trigger

Syntax

```
TRIGger:A:LOGIc:INPut:D<x> {HIGH|LOW|X}
TRIGger:A:LOGIc:INPut:D<x>?
```

Arguments

`High` specifies the logic high state.

`Low` specifies the logic low state.

`X` specifies a "don't care" state.

TRIGger:A:LOGIc:PATtern? (Query Only)

Returns the conditions used for generating an A logic pattern trigger, with respect to the defined input pattern, and identifies the time that the selected pattern may be true and still generate the trigger.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:PATtern?
```

Examples

```
TRIGGER:A:LOGIC:PATTERN? might return
:TRIGGER:A:LOGIC:PATTERN:INPUT:CH1 HIGH;CH2
LOW;CH3 X;CH4 X;:TRIGGER:A:LOGIC:PATTERN:WHEN
LESSTHAN;WHEN:LESSLIMIT 16.0000E-9;LIMIT
16.0000E-9;:TRIGGER:A:LOGIC:PATTERN:DELTATIME 16.0000E-9
```

TRIGger:A:LOGIc:PATtern:DELTatime

Sets or returns the pattern trigger delta time value. The time value is used as part of the pattern trigger condition to determine if the duration of a logic pattern meets the specified time constraints.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:PATtern:DELTatime <NR3>
TRIGger:A:LOGIc:PATtern:DELTatime?
```

Arguments `<NR3>` is a floating point value with exponent that sets the pattern trigger time value. This argument has a range of 39.6E–9 (39.6 ns) to 10.0E0 (10 s), in increments of 13.2 ns. Values that are not an increment of 13.2 ns are rounded to the nearest correct value.

Examples `TRIGGER:A:LOGIC:PATTERN:DELTATIME 71.28E-8` sets the pattern trigger delta time value to 712.8 ns.

TRIGger:A:LOGIc:PATtern:INPut:D<x>

Sets or returns the A logic trigger input for the specified digital channel <x>, where x is the channel number. This command species the logic value used when the pattern trigger detects the threshold level.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:PATtern:INPut:D<x> {HIGH|LOW|X}
TRIGger:A:LOGIc:PATtern:INPut:D<x>?
```

Arguments

`HIGH` specifies a logic high.

`LOW` specifies a logic low.

`X` specifies a "do not care" state.

TRIGger:A:LOGIc:PATtern:WHEn

Sets or returns the pattern logic condition on which to trigger the oscilloscope.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:PATtern:WHEn
{TRUe|FALSe|LESSThan|MOREThan|EQUal|UNEQual}
TRIGger:A:LOGIc:PATtern:WHEn?
```

Arguments

`TRUE` triggers the oscilloscope when the pattern becomes true.

`FALSE` triggers the oscilloscope when the pattern becomes false.

`LESSTHAN` triggers the oscilloscope when the input pattern is true for a time period less than the time period specified in `TRIGGER:A:LOGIC:PATTERN:DELTATIME`.

`MORETHAN` triggers the oscilloscope when the input pattern is true for a time period more (greater) than the time period specified in `TRIGGER:A:LOGIC:PATTERN:DELTATIME`.

`EQUAL` triggers the oscilloscope when the input pattern is true for a time period equal to the time period specified in `TRIGGER:A:LOGIC:PATTERN:DELTATIME`, within a ±5% tolerance.

`UNEQUAL` triggers the oscilloscope when the input pattern is true for a time period greater than or less than (not equal to) the time period specified in `TRIGGER:A:LOGIC:PATTERN:DELTATIME`, within a ±5% tolerance.

Examples `TRIGGER:A:LOGIC:PATTERN:WHEN LESSTHAN` sets the oscilloscope to trigger when the pattern is true for a time period less than the pattern trigger delta time setting.

TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit

Sets or returns the maximum time that the selected pattern may be true and still generate an A logic pattern trigger.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit <NR3>
TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit?
```

Arguments `<NR3>` specifies the maximum amount of time to hold the pattern true.

Examples `TRIGGER:A:LOGIC:PATTERN:WHEN:LESSLIMIT 10.0E+00` sets the maximum time that the selected pattern may hold true (and generate an A logic pattern trigger) to 10 s.

`TRIGGER:A:LOGIC:PATTERN:WHEN:LESSLIMIT?` might return `TRIGGER:A:LOGIC:PATTERN:WHEN:LESSLIMIT 8.0000E-9` indicating that the selected pattern may hold true for up to 8 ns and still generate an A logic pattern trigger.

TRIGger:A:LOGIc:PATtern:WHEn:MORELimit

Sets or returns the minimum time that the selected pattern may be true and still generate an A logic pattern trigger.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:PATtern:WHEn:MORELimit <NR3>
TRIGger:A:LOGIc:PATtern:WHEn:MORELimit?
```

Arguments `<NR3>` specifies the minimum amount of time to hold the pattern true.

Examples `TRIGGER:A:LOGIC:PATTERN:WHEN:MORELIMIT 10.0E+00` sets the minimum time that the selected pattern may hold true (and generate an A logic pattern trigger) to 10 s.

`TRIGGER:A:LOGIC:PATTERN:WHEN:MORELIMIT?` might return `TRIGGER:A:LOGIC:PATTERN:WHEN:MORELIMIT 8.0000E-9` indicating that the selected pattern must hold true for at least 8 ns to generate an A logic pattern trigger.

TRIGger:A:LOGIc:THReshold:(CH<x>|D<x>)

This command sets or queries the logic trigger threshold voltage for both analog and digital channels (for MSO models), specified by x.

Group Trigger

Syntax

```
TRIGger:A:LOGIc:THReshold:(CH<x>|D<x>) {<NR3>|ECL|TTL}
TRIGger:A:LOGIc:THReshold:(CH<x>|D<x>)?
```

Arguments

`<NR3>` specifies the threshold voltage, in volts.

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

Examples

`TRIGGER:A:LOGIC:THRESHOLD:CH2 3.0E-3` sets the A logic trigger threshold voltage for Channel 2 to 3 mV.

`TRIGGER:A:LOGIC:THRESHOLD:CH3?` might return `:TRIGGER:A:LOGIC:THRESHOLD:CH3 1.2000E+00`, indicating that the A logic trigger threshold voltage for Channel 3 is 1.2 V.

TRIGger:A:LOWerthreshold:CH<x>

Sets or returns the lower threshold for the channel selected. Each channel can have an independent level. Used in Runt and Slew Rate triggers as the lower threshold. Used for all other trigger types as the single level/threshold.

Group Trigger

Syntax

```
TRIGger:A:LOWerthreshold:CH<x> {ECL|TTL|<NR3>}
TRIGger:A:LOWerthreshold:CH<x>?
```

Related Commands TRIGger:A:LEVel:CH<x>

Arguments

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

`<NR3>` is the clock level, in volts.

Examples

`TRIGGER:A:LOWERTHRESHOLD:CH2 50E-3` sets the lower limit threshold for CH2 of the pulse runt trigger to 50 mV.

`TRIGGER:A:LOWERTHRESHOLD:CH2?` might return `:TRIGGER:A:LOWERTHRESHOLD:CH2 1.2000E-01` indicating that the lower limit threshold for CH2 of the pulse runt trigger is set to 120 mV.

TRIGger:A:LOWerthreshold{:EXT|:AUX}

Sets or returns the lower threshold for the Auxiliary Input. It is similar to TRIGger:A:LEVel:AUXin.

Group

Trigger

Syntax

```
TRIGger:A:LOWerthreshold{:EXT|:AUX} {<NR3>|ECL|TTL}
TRIGger:A:LOWerthreshold{:EXT|:AUX}?
```

Arguments

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

`<NR3>` specifies the threshold level in volts.

TRIGger:A:MODe

Sets or returns the A trigger mode.

Group

Trigger

Syntax

```
TRIGger:A:MODe {AUTO|NORMal}
TRIGger:A:MODe?
```

Related Commands

TRIGger:A:LEVel

Arguments

`AUTO` generates a trigger if one is not detected within a specified time period.

`NORMal` waits for a valid trigger event.

Examples

`TRIGGER:A:MODE NORMAL` specifies that a valid trigger event must occur before a trigger is generated.

`TRIGGER:A:MODE ?` might return `:TRIGGER:A:MODE NORMAL` indicating that a valid trigger event must occur before a trigger is generated.

TRIGger:A:PULse? (Query Only)

Returns the A pulse trigger parameters.

Group Trigger

Syntax

```
TRIGger:A:PULse?
```

Related Commands TRIGger:A:EDGE?, TRIGger:A:LOGIc?

Examples `TRIGGER:A:PULSE?` might return `:TRIGGER:A:PULSE:CLASS TRAnsITION`

TRIGger:A:PULse:CLAss

Sets or returns the type of pulse on which to trigger.

Group Trigger

Syntax

```
TRIGger:A:PULse:CLAss {RUNt|WIDth|TRANsition}
TRIGger:A:PULse:CLAss?
```

Related Commands TRIGger:A:RUNT?, TRIGger:A:PULSEWIDth?, TRIGger:A{:TRANsition|:RISEFall}?, TRIGger:A:TYPe

Arguments `RUNt` triggers when a pulse crosses the first preset voltage threshold but does not cross the second preset threshold before recrossing the first.

`WIDth` triggers when a pulse is found that has the specified polarity and is either inside or outside the specified time limits.

`TRAnsition` triggers when a pulse crosses both thresholds in the same direction as the specified polarity and the transition time between the two threshold crossings is greater or less than the specified time delta.

Examples

`TRIGGER:A:PULSE:CLASS WIDTH` specifies a width pulse for the A trigger.

`TRIGGER:A:PULSE:CLASS?` might return `:TRIGGER:A:PULSE:CLASS WIDTH` indicating that a pulse was found that is of the specified polarity and width.

TRIGger:A:PULSEWIDth? (Query Only)

Returns the width parameters for the pulse width trigger.

Group

Trigger

Syntax

`TRIGger:A:PULSEWIDth?`

Examples

`TRIGGER:A:PULSEWIDTH?` might return
`:TRIGGER:A:PULSEWIDTH:POLARITY POSITIVE;WHEN LESSTHAN;WIDTH 8.0000E-9`

TRIGger:A:PULSEWidth:POLarity

Sets or returns the polarity for the width trigger.

Group

Trigger

Syntax

```
TRIGger:A:PULSEWidth:POLarity {NEGative|POSitive}
TRIGger:A:PULSEWidth:POLarity?
```

Arguments

`NEGative` specifies a negative pulse.

`POSitive` specifies a positive pulse.

Examples

`TRIGGER:A:PULSEWIDTH:POLARITY NEGATIVE` sets the pulse polarity to negative.

`TRIGGER:A:PULSEWIDTH:POLARITY?` might return `:TRIGGER:A:WIDTH:POLARITY POSITIVE` indicating a positive pulse.

TRIGger:A:PULSEWidth:SOUrce

Sets or returns the source for the pulse-width trigger.

Group Trigger

Syntax

```
TRIGger:A:PULSEWidth:SOUrce {{CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15|LINE|EXT}
TRIGger:A:PULSEWidth:SOUrce?
```

Arguments

`CH1–CH4` specifies an analog input channel as the A edge trigger source.

`D0–D15` specifies an digital input channel as the A edge trigger source.

`EXT` specifies an external trigger using the Aux In connector located on the front panel of the oscilloscope.

`LINE` specifies AC line voltage.

Examples

`TRIGGER:A:PULSEWIDTH:SOURCE CH1` sets channel 1 as the pulse width source.

`TRIGGER:A:PULSEWIDTH:SOURCE?` might return `:TRIGGER:A:PULSEWIDTH:SOURCE CH1` indicating that channel 1 is the pulse width source.

TRIGger:A:PULSEWidth:WHEn

Sets or returns whether to trigger on a pulse that meets, falls outside, or within the specified range of limits.

Group Trigger

Syntax

```
TRIGger:A:PULSEWidth:WHEn {LESSthan|MOREthan|EQual|UNEQual}
TRIGger:A:PULSEWidth:WHEn?
```

Related Commands TRIGger:A:PULSEWidth:WIDth

Arguments

`LESSthan` argument sets the oscilloscope to trigger if a pulse is detected with width less than the time set by the TRIGger:A:PULSEWidth:WIDth command.

`MOREthan` argument sets the oscilloscope to trigger if a pulse is detected with width more than the time set by the TRIGger:A:PULSEWidth:WIDth command.

`EQUal` argument sets the oscilloscope to trigger if a pulse is detected with width equal to the time period specified in TRIGger:A:PULSEWidth:WIDth within a ±5% tolerance.

`UNEQual` argument sets the oscilloscope to trigger if a pulse is detected with width greater than or less than (but not equal) the time period specified in TRIGger:A:PULSEWidth:WIDth within a ±5% tolerance.

Examples

`TRIGGER:A:PULSEWIDTH:WHEN LESSTHAN` specifies that the duration of the A pulse will fall within defined high and low limits.

`TRIGGER:A:PULSEWIDTH:WHEN?` might return `:TRIGGER:A:PULSEWIDTH:WHEN THAN` indicating that the conditions for generating a width trigger.

TRIGger:A:PULSEWidth:WIDth

Sets or returns the width setting for the pulse width trigger.

Group

Trigger

Syntax

```
TRIGger:A:PULSEWidth:WIDth <NR3>
TRIGger:A:PULSEWidth:WIDth?
```

Related Commands

TRIGger:A:PULSEWidth:WHEn

Arguments

`<NR3>` specifies the pulse width in seconds.

Examples

`TRIGGER:A:PULSEWIDTH:WIDTH 5.0E-6` sets the pulse width to 5 µs.

`TRIGGER:A:PULSEWIDTH:WIDTH?` might return `:TRIGGER:A:PULSEWIDTH:WIDTH 2.0000E-9` indicating that the pulse width is set to 2 ns.

TRIGger:A:RUNT? (Query Only)

Returns the current A runt trigger parameters.

Group

Trigger

Syntax

`TRIGger:A:RUNT?`

Examples

`TRIGGER:A:RUNT?` might return `:TRIGGER:A:RUNT:SOURCE CH1;POLARITY POSITIVE;WHEN OCCURS;WIDTH 4.0000E-9`.

TRIGger:A:RUNT:POLarity

Sets or returns the polarity for the runt trigger.

Group Trigger

Syntax

```
TRIGger:A:RUNT:POLarity {EITher|NEGative|POSitive}
TRIGger:A:RUNT:POLarity?
```

Arguments

`POSitive` indicates that the rising edge crosses the low threshold and the falling edge recrosses the low threshold without either edge ever crossing the high threshold.

`NEGative` indicates that the falling edge crosses the high threshold and the rising edge recrosses the high threshold without either edge ever crossing the low threshold.

`EITher` triggers on a runt of either polarity.

Examples

`TRIGGER:A:RUNT:POLARITY NEGATIVE` specifies that the polarity of the A pulse runt trigger is negative.

`TRIGGER:A:RUNT:POLARITY?` might return `:TRIGGER:A:RUNT:POLARITY POSITIVE` indicating that the polarity of the A pulse runt trigger is positive.

TRIGger:A:RUNT:SOUrce

Sets or returns the source for the A runt trigger.

Group Trigger

Syntax

```
TRIGger:A:RUNT:SOUrce {CH1|CH2|CH3|CH4}
TRIGger:A:RUNT:SOUrce?
```

Arguments

`CH1–CH4` specifies the input channel number, depending on the model of the oscilloscope.

Examples

`TRIGGER:A:RUNT:SOURCE CH4` sets channel 4 as the source for the A pulse trigger.

`TRIGGER:A:RUNT:SOURCE?` might return `:TRIGGER:A:RUNT:SOURCE CH2` indicating that channel 2 is the source for the A pulse trigger.

TRIGger:A:RUNT:WHEn

Sets or returns the type of pulse width the trigger checks for when it detects a runt.

Group Trigger

Syntax

```
TRIGger:A:RUNT:WHEn {LESSthan|MOREthan|EQual|UNEQual|OCCURS}
TRIGger:A:RUNT:WHEn?
```

Related Commands TRIGger:A:RUNT:WIDth

Arguments `OCCURS` argument specifies a trigger event if a runt of any detectable width occurs.

`LESSthan` argument sets the oscilloscope to trigger if the a runt pulse is detected with width less than the time set by the TRIGger:A:RUNT:WIDth command.

`MOREthan` argument sets the oscilloscope to trigger if the a runt pulse is detected with width more than the time set by the TRIGger:A:RUNT:WIDth command.

`EQUal` argument sets the oscilloscope to trigger if a runt pulse is detected with width equal to the time period specified in TRIGger:A:RUNT:WIDth within a ±5% tolerance.

`UNEQual` argument sets the oscilloscope to trigger if a runt pulse is detected with width greater than or less than (but not equal to) the time period specified in TRIGger:A:RUNT:WIDth within a ±5% tolerance.

Examples `TRIGGER:A:RUNT:WHEN THAN` sets the runt trigger to occur when the oscilloscope detects a runt in a pulse wider than the specified width.

`TRIGGER:A:RUNT:WHEN?` might return `:TRIGGER:A:PULSE:RUNT:WHEN OCCURS` indicating that a runt trigger will occur if the oscilloscope detects a runt of any detectable width.

TRIGger:A:RUNT:WIDth

Sets or returns the width for a runt trigger.

Group Trigger

Syntax

```
TRIGger:A:RUNT:WIDth <NR3>
TRIGger:A:RUNT:WIDth?
```

Related Commands TRIGger:A:RUNT:WHEn

Arguments `<NR3>` specifies the minimum width, in seconds.

Examples

`TRIGGER:A:RUNT:WIDTH 15E-6` sets the minimum width of the pulse runt trigger to 15 µs.

`TRIGGER:A:RUNT:WIDTH?` might return `:TRIGGER:A:PULSE:RUNT:WIDTH 2.0000E-09` indicating that the minimum width of a pulse runt trigger is 2 ns.

TRIGger:A:SETHold? (Query Only)

Returns the clock edge polarity, voltage threshold and source input; data voltage threshold and source; and both setup and hold times for setup and hold violation triggering.

Group

Trigger

Syntax

`TRIGger:A:SETHold?`

Examples

`TRIGGER:A:SETHOLD?` might return
`:TRIGGER:A:SETHOLD:CLOCK:SOURCE CH1;EDGE RISE;THRESHOLD 100.0000E-3;:TRIGGER:A:SETHOLD:DATA:SOURCE CH2;THRESHOLD 80.0000E-3;:TRIGGER:A:SETHOLD:HOLDTIME 20.0000E-9;SETTIME 8.0000E-9`

TRIGger:A:SETHold:CLOCk? (Query Only)

Returns the clock edge polarity, voltage threshold, and source input for setup and hold triggering.

Group

Trigger

Syntax

`TRIGger:A:SETHold:CLOCk?`

Examples

`TRIGGER:A:SETHOLD:CLOCK?` might return
`:TRIGGER:A:SETHOLD:CLOCK:SOURCE EXT;EDGE FALL;THRESHOLD 1.4000`

TRIGger:A:SETHold:CLOCk:EDGE

Sets or returns the clock edge polarity for setup and hold triggering.

Group Trigger

Syntax

```
TRIGger:A:SETHold:CLOCk:EDGE {FALL|RISe}
TRIGger:A:SETHold:CLOCk:EDGE?
```

Arguments

`FALL` specifies polarity as the clock falling edge.

`RISe` specifies polarity as the clock rising edge.

Examples

`TRIGGER:A:SETHOLD:CLOCK:EDGE RISE` specifies the polarity as the clock rising edge.

`TRIGGER:A:SETHOLD:CLOCK:EDGE?` might return `:TRIGGER:A:SETHOLD:CLOCK:EDGE RISE` indicating that polarity is specified as the clock rising edge.

TRIGger:A:SETHold:CLOCk:SOUrce

Sets or returns the clock source for the setup and hold triggering.

Group Trigger

Syntax

```
TRIGger:A:SETHold:CLOCk:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
TRIGger:A:SETHold:CLOCk:SOUrce?
```

Related Commands TRIGger:A:SETHold:DATa:SOUrce

Arguments

`CH1–CH4` specifies the input channel number.

`D0-D15` specifies the input digital channel number.

Examples

`TRIGGER:A:SETHOLD:CLOCK:SOURCE CH1` specifies channel 1 as the clock input for setup and hold input.

`TRIGGER:A:SETHOLD:CLOCK:SOURCE?` might return `:TRIGGER:A:SETHOLD:CLOCK:SOURCE CH4` indicating that channel 4 is the clock source for the setup and hold trigger input.

TRIGger:A:SETHold:CLOCk:THReshold

Sets or returns the clock voltage threshold for the setup and hold trigger.

Group Trigger

Syntax

```
TRIGger:A:SETHold:CLOCk:THReshold {<NR3>|TTL|ECL}
TRIGger:A:SETHold:CLOCk:THReshold?
```

Arguments

`TTL` specifies a preset `TTL` high level of 1.4 V.

`ECL` specifies a preset `ECL` high level of –1.3 V.

`<NR3>` is the clock level, in volts.

Examples

`TRIGGER:A:SETHOLD:CLOCK:THRESHOLD TTL` specifies the preset TTL value of 1.4 V as the clock threshold for the setup and hold trigger.

`TRIGGER:A:SETHOLD:CLOCK:THRESHOLD?` might return
`:TRIGGER:A:LOGIC:SETHOLD:CLOCK:THRESHOLD 1.2000E+00`
indicating that the clock threshold for the setup and hold trigger is 1.2 V.

TRIGger:A:SETHold:DATa? (Query Only)

Returns the voltage threshold and data source for the setup and hold trigger.

Group Trigger

Syntax

```
TRIGger:A:SETHold:DATa?
```

Related Commands TRIGger:A:SETHold:CLOCk?

Examples

`TRIGGER:A:SETHOLD:DATA?` might return
`:TRIGGER:A:SETHOLD:DATA:SOURCE CH2;THRESHOLD 80.0000E-3`

TRIGger:A:SETHold:DATa:SOUrce

Sets or returns the data source for the setup and hold trigger. You cannot specify the same source for both clock and data.

For DPO models, you can specify only a single data source. Data sources for DPO models may be one of CH1–CH4 or D0–D15.

Group Trigger

Syntax

DPO Models:

```
TRIGger:A:SETHold:DATa:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
TRIGger:A:SETHold:DATa:SOUrce?
```

Related Commands

TRIGger:A:SETHold:CLOCk:SOUrce

Arguments

DPO Models:

`<wfm>` specifies the source channel number and is one of CH1–CH4, and D0–D15. You can specify only one waveform on a DPO.

Examples

`TRIGGER:A:SETHOLD:DATA:SOURCE CH1` sets channel 1 as the clock source for the setup and hold trigger.

`TRIGGER:A:SETHOLD:DATA:SOURCE?` might return `:TRIGGER:A:LOGIC:SETHOLD:DATA:SOURCE CH2` indicating that channel 2 is the current clock source for the setup and hold trigger.

TRIGger:A:SETHold:DATa:THReshold

Sets or returns the data voltage threshold for setup and hold trigger. The digital channels do not have independent trigger levels. Channels D0–D7 share one common trigger level and D8–D15 share another common trigger level. For example changing the trigger level for D3 changes the levels for D0–D7.

Group

Trigger

Syntax

```
TRIGger:A:SETHold:DATa:THReshold {<NR3>|TTL|ECL}
TRIGger:A:SETHold:DATa:THReshold?
```

Arguments

`TTL` specifies the preset `TTL` high level of 1.4 V.

`ECL` specifies the preset `ECL` high level of –1.3 V.

`<NR3>` is the setup and hold data level, in V.

Examples

`TRIGGER:A:SETHOLD:DATA:THRESHOLD TTL` specifies the preset high level of 1.4 V as the current data voltage level for the setup and hold trigger.

`TRIGGER:A:SETHOLD:DATA:THRESHOLD?` might return `:TRIGGER:A:SETHOLD:DATA:THRESHOLD 1.2000E+00` indicating that 1.2 V is the current data voltage level for the setup and hold trigger.

TRIGger:A:SETHold:HOLDTime

Sets or returns the hold time for setup and hold violation triggering.

Group Trigger

Syntax

```
TRIGger:A:SETHold:HOLDTime <NR3>
TRIGger:A:SETHold:HOLDTime?
```

Arguments <NR3> specifies the hold time setting in seconds. Positive values for hold time occur after the clock edge. Negative values occur before the clock edge.

Examples `TRIGGER:A:SETHOLD:HOLDTIME 3.0E-3` sets the hold time for the setup and hold trigger to 3 ms.

`TRIGGER:A:SETHOLD:HOLDTIME?` might return `:TRIGGER:A:SETHOLD:HOLDTIME 2.0000E-09` indicating that the current hold time for the setup and hold trigger is 2 ns.

TRIGger:A:SETHold:SETTime

Sets or returns the setup time for setup and hold violation triggering.

Group Trigger

Syntax

```
TRIGger:A:SETHold:SETTime <NR3>
TRIGger:A:SETHold:SETTime?
```

Arguments <NR3> specifies the setup time for setup and hold violation triggering.

Examples `TRIGGER:A:SETHOLD:SETTIME 3.0E-6` specifies that the current setup time for setup and hold trigger is 3 µs.

`TRIGGER:A:SETHOLD:SETTIME?` might return `:TRIGGER:A:LOGIC:SETHOLD:SETTIME 2.0000E-09` indicating that the current setup time for setup and hold trigger is 2 ns.

TRIGger:A:SETHold:THReshold:CH<x>

Sets or queries the threshold for the channel specified by x. Affects all trigger types using the channel.

Group Trigger

Syntax

```
TRIGger:A:SETHold:THReshold:CH<x> {<NR3>|ECL|TTL}
TRIGger:A:SETHold:THReshold:CH<x>?
```

Arguments

`<NR3>` specifies the threshold voltage, in volts.

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

Examples

`TRIGGER:A:SETHOLD:THRESHOLD:CH1 1.5` sets the channel 1 threshold to 1.5 volts.

`TRIGGER:A:SETHOLD:THRESHOLD:CH1?` might return `TRIGGER:A:SETHOLD:THRESHOLD:CH1 0.0E+0` indicating that the channel 1 threshold is set to 0.0 volts.

TRIGger:A:SETHold:THReshold:D<x>

Sets the Threshold for the Selected Digital Channel. Affects all trigger types using the Digital Channel.

Group MSO

Syntax

```
TRIGger:A:SETHold:THReshold:D<x> {<NR3>|ECL|TTL}
TRIGger:A:SETHold:THReshold:D<x>?
```

Arguments

`<NR3>` specifies the threshold voltage, in volts.

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

Examples

`TRIGGER:A:SETHOLD:THRESHOLD:D1 ECL` sets the threshold to ECL levels.

`TRIGGER:A:SETHOLD:THRESHOLD:D1?` might return `TRIGGER:A:SETHOLD:THRESHOLD:D1 -1.3000` indicating that the threshold is set to –1.3 volts.

TRIGger:A{:TRANsition|:RISEFall}? (Query Only)

Returns transition time trigger parameters.

Group Trigger

Syntax
```
TRIGger:A{:TRANsition|:RISEFall}?
```

Related Commands TRIGger:A:UPPerthreshold:CH<x> , TRIGger:A:LOWerthreshold:CH<x>

Examples
```
TRIGGER:A::TRAnsITION? might return
:TRIGGER:A:TRANSITION:POLARITY POSITIVE;WHEN
SLOWER;DELTATIME 8.0000E-9
```

TRIGger:A{:TRANsition|:RISEFall}:DELTatime

Sets or returns the delta time used in calculating the transition value for the transition trigger.

Group Trigger

Syntax
```
TRIGger:A{:TRANsition|:RISEFall}:DELTatime <NR3>
TRIGger:A{:TRANsition|:RISEFall}:DELTatime?
```

Arguments `<NR3>` specifies the delta time, in seconds.

Examples `TRIGGER:A:TRAnsITION:DELTATIME 15E-6` sets the delta time of the transition trigger to 15 µs.

`TRIGGER:A:TRAnsITION:DELTATIME?` might return `:TRIGGER:A :TRANSITION:DELTATIME 2.0000E-09` indicating that the delta time of the transition trigger is set to 2 ns.

TRIGger:A{:TRANsition|:RISEFall}:POLarity

Sets or returns the polarity for the transition trigger.

Group Trigger

Syntax
```
TRIGger:A{:TRANsition|:RISEFall}:POLarity
{EITher|NEGative|POSitive}
TRIGger:A{:TRANsition|:RISEFall}:POLarity?
```

Arguments

`POSitive` indicates that a pulse edge must traverse from the lower (most negative) to higher (most positive) level for transition triggering to occur.

`NEGative` indicates that a pulse edge must traverse from the upper (most positive) to lower (most negative) level for transition triggering to occur.

`EITher` indicates either positive or negative polarity.

Examples

`TRIGGER:A:TRAnsITION:POLARITY NEGATIVE` sets the transition polarity to negative.

`TRIGGER:A:TRAnsITION:POLARITY?` might return `:TRIGGER:A:TRANSITION:POLARITY EITHER` indicating that the polarity can be either positive or negative.

TRIGger:A{:TRANsition|:RISEFall}:SOUrce

Sets or returns the source for transition trigger.

Group

Trigger

Syntax

```
TRIGger:A{:TRANsition|:RISEFall}:SOUrce {CH1|CH2|CH3|CH4|
D0|D1|D2|D3|D4|D5|D6|D7|D8|D9|D10|D11|D12|D13|D14|D15}
TRIGger:A{:TRANsition|:RISEFall}:SOUrce?
```

Arguments

`CH1–CH4` specifies one of the input channels.

`D0–D15` specifies one of the input digital channels.

Examples

`TRIGGER:A:TRAnsITION:SOURCE CH4` sets channel 4 as the source for the transition trigger.

`TRIGGER:A:TRAnsITION:SOURCE?` might return `:TRIGGER:A:TRANSITION:SOURCE CH2` indicating that channel 2 is the source for the A transition trigger.

TRIGger:A{:TRANsition|:RISEFall}:WHEn

Sets or returns whether to check for a transitioning signal that is faster or slower than the specified delta time.

Group

Trigger

Syntax

```
TRIGger:A{:TRANsition|:RISEFall}:WHEn
{SLOWer|FASTer|EQual|UNEQual}
TRIGger:A{:TRANsition|:RISEFall}:WHEn?
```

Arguments

`FASTer` sets the trigger to occur when the signal transition time is faster than the time set by `TRIGger:A{TRAnsition|:RISEFall}:DELTatime`.

`SLOWer` sets the trigger to occur when the signal transition time is slower than the time set by.`TRIGger:A{TRAnsition|:RISEFall}:DELTatime`.

`EQual` sets the trigger to occur when the signal transition time is equal to the time set by `TRIGger:A{TRAnsition|:RISEFall}:DELTatime`.

`UNEQual` sets the trigger to occur when the signal transition time is not equal to the time set by `TRIGger:A{TRAnsition|:RISEFall}:DELTatime`.

Examples

`TRIGGER:A:TRAnsITION:WHEN SLOWER` sets the trigger to occur when the signal transition time is slower than the time set by.`TRIGger:A{TRAnsition|:RISEFall}:DELTatime`.

`TRIGGER:A:TRAnsITION:WHEN?` might return `:TRIGGER:A:TRANSITION:WHEN FASTER`

TRIGger:A:TYPe

Sets or returns the type of A trigger. The five types of triggers are of Edge, Logic, Pulse, Serial, and Video. Logic and Pulse triggers contain classes. Logic triggers consist of State, Pattern, and SetHold classes; Pulse triggers consist of Runt, Width, and Transition logic classes. Once you have set the trigger type, you may also need to identify the associated trigger class. For details on selecting Logic and Pulse trigger classes, see TRIGger:A:LOGIc:CLAss and TRIGger:A:PULse:CLAss respectively.

Group

Trigger

Syntax

```
TRIGger:A:TYPe {EDGe|LOGic|PULSe|BUS|VIDeo}
TRIGger:A:TYPe?
```

Related Commands

TRIGger:A:EDGE?, TRIGger:A:LOGIc:CLAss, TRIGger:A:PULse:CLAss

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:A:EDGE? commands.

`LOGic` specifies that a trigger occurs when specified conditions are met and is controlled by the TRIGger:A:LOGIc? commands.

`PULSe` specifies that a trigger occurs when a specified pulse is found and is controlled by the TRIGger:A:PULse? commands.

`BUS` specifies that a trigger occurs when a communications signal is found. Supports CAN, I²C, SPI, and RS232 communications signals.

`VIDeo` specifies that the trigger occurs when a video signal is found.

Examples

`TRIGGER:A:TYPE EDGE` sets the A trigger type to EDGE.

`TRIGGER:A:TYPE?` might return `:TRIGGER:A:TYPE PULSE` indicating that the A trigger type is a pulse trigger.

TRIGger:A:UPPerthreshold:CH<x>

Sets the upper threshold for channel <x>, where x is the channel number. Each channel can have an independent level. Used only for runt and slew rate trigger types.

Group

Trigger

Syntax

```
TRIGger:A:UPPerthreshold:CH<x> {<NR3>|ECL|TTL}
TRIGger:A:UPPerthreshold:CH<x>?
```

Arguments

`<NR3>` is the threshold level in volts.

`ECL` specifies a preset `ECL` high level of –1.3 V.

`TTL` specifies a preset `TTL` high level of 1.4 V.

Examples

`TRIGGER:A:UPPERTHRESHOLD:CH2 50E-3` sets the upper limit of the pulse runt trigger to 50 mV for channel 2.

`TRIGGER:A:UPPERTHRESHOLD:CH2?` might return `:TRIGGER:A:UPPERTHRESHOLD:CH2 1.2000E-01` indicating that the upper limit of the pulse runt trigger is set to 120 mV.

TRIGger:A:VIDeo? (Query Only)

Returns the A trigger video parameters.

Group Trigger

Syntax

```
TRIGger:A:VIDeo?
```

Examples

`TRIGGER:A:VIDEO?` might return `:TRIGGER:A:VIDEO:POLARITY POSITIVE;SOURCE CH1;STANDARD NTSC;SYNC ALLLINES;HOLDOFF:FIELD 0.0000;:TRIGGER:A:VIDEO:CUSTOM:FORMAT PROGRESSIVE;SCANRATE15K;:TRIGGER: A:VIDEO:LINE 1;HDTV:FORMAT HD1080I60`

TRIGger:A:VIDeo:HOLDoff:FIELD

Sets or returns the video trigger holdoff in terms of video fields.

Group Trigger

Syntax

```
TRIGger:A:VIDeo:HOLDoff:FIELD <NR3>
TRIGger:A:VIDeo:HOLDoff:FIELD?
```

Arguments

`<NR3>` argument is a real number from 0.0 to 8.5 in increments of 0.5. The argument sets the number of fields that the oscilloscope waits before rearming the video trigger.

Examples

`TRIGGER:A:VIDEO:HOLDOFF:FIELD?` might return `:TRIGGER:A:VIDEO:HOLDOFF:FIELD 5` indicating that the oscilloscope is set to wait 5 video fields before rearming the trigger.

`TRIGGER:A:VIDEO:HOLDOFF:FIELD 4.5` sets the oscilloscope to wait 4.5 video fields before rearming the trigger.

TRIGger:A:VIDeo:LINE

Sets or returns the video line number on which the oscilloscope triggers. Use the TRIGger:A:VIDeo{:SYNC|:FIELD} command to actually trigger the oscilloscope on the line that you specify with this command.

Group Trigger

Syntax

```
TRIGger:A:VIDeo:LINE <NR1>
TRIGger:A:VIDeo:LINE?
```

Related Commands TRIGger:A:VIDeo{:SYNC|:FIELD}

Arguments `<NR1>` argument is an integer that sets the video line number on which the oscilloscope triggers. The following table lists the valid choices, depending on the active video standard.

Table 2-41: Video Line Numbering Ranges

Video Standard	Line Number Range
525/NTSC	1–525
625/PAL, SECAM	1–625
SECAM	1–625

Examples `TRIGGER:A:VIDEO:LINE 23` sets the oscilloscope to trigger on the line 23.

`TRIGGER:A:VIDEO:LINE ?` might return `:TRIGGER:A:VIDEO:LINE 10` indicating that the oscilloscope is set to trigger on line 10.

TRIGger:A:VIDeo:POLarity

Sets or returns the polarity of the A video trigger.

Group Trigger

Syntax

```
TRIGger:A:VIDeo:POLarity {NEGative|POSitive|NORMal|INVERTed}
TRIGger:A:VIDeo:POLarity?
```

Arguments `POSitive` argument sets the oscilloscope to trigger on a positive video sync pulse.

`NEGative` argument sets the oscilloscope to trigger on a negative video sync pulse.

Examples `TRIGGER:A:VIDEO:POLARITY NEGATIVE` sets the oscilloscope to trigger on a negative video pulse.

`TRIGGER:A:VIDEO:POLARITY?` might return `:TRIGGER:A:VIDEO:POLARITY POSITIVE` indicating that the oscilloscope is set to trigger on a positive video sync pulse.

TRIGger:A:VIDeo:SOUrce

Sets or returns the source for the A video trigger.

Group Trigger

Syntax

```
TRIGger:A:VIDeo:SOUrce {CH1|CH2|CH3|CH4}
TRIGger:A:VIDeo:SOUrce?
```

Arguments `CH1–CH4` specifies the input channel to use as the A video trigger.

`D0–D15` specifies the input channel to use as the A video trigger.

Examples `TRIGGER:A:VIDEO:SOURCE CH1` sets the source for A video trigger to Channel 1.

`TRIGGER:A:VIDEO:SOURCE?` might return `:TRIGGER:A:VIDEO:SOURCE CH2` indicating that the source for the A video trigger is set to Channel 2.

TRIGger:A:VIDeo:STANdard

Sets or returns the standard for the video trigger.

Group Trigger

Syntax

```
TRIGger:A:VIDeo:STANdard {NTSc|PAL|SECAM}
TRIGger:A:VIDeo:STANdard?
```

Arguments `NTSc` sets the oscilloscope to trigger on video signals that meet the NTSC 525/60/2:1 standard (a line rate of 525 lines per frame and a field rate of 60 Hz).

`PAL` sets the oscilloscope to trigger on video signals that meet the NTSC 625/50/2:1 standard (a line rate of 625 lines per frame and a field rate of 50 Hz).

`SECAM` sets the oscilloscope to trigger on video signals that meet the SECAM standard.

Examples `TRIGGER:A:VIDEO:STANDARD NTSC` sets the oscilloscope to trigger on NTSC-standard video signals.

`TRIGGER:A:VIDEO:STANDARD?` might return `:TRIGGER:A:VIDEO:STANDARD HDTV` indicating that the oscilloscope is set to trigger on an HDTV format.

TRIGger:A:VIDeo{:SYNC|:FIELD}

Sets or returns the video field or line that the trigger detects.

Group Trigger

Syntax

```
TRIGger:A:VIDeo{:SYNC|:FIELD}
{ODD|EVEN|ALLFields|ALLLines|NUMERic}
TRIGger:A:VIDeo{:SYNC|:FIELD}?
```

Arguments `ODD` argument sets the oscilloscope to trigger on interlaced video odd fields.

`EVEN` argument sets the oscilloscope to trigger on interlaced video even fields.

`ALLFields` argument sets the oscilloscope to trigger on all fields.

`ALLLines` argument sets the oscilloscope to trigger on all video lines.

`NUMERic` argument sets the oscilloscope to trigger on the video signal line specified by the TRIGger:A:VIDeo:LINE command.

Examples `TRIGGER:A:VIDEO:FIELD EVEN` sets the A video trigger so that it will trigger on even fields.

`TRIGGER:A:VIDEO:FIELD?` might return `:TRIGGER:A:VIDEO:FIELD ALLFIELDS` indicating that the A video will trigger on all video fields.

TRIGger:EXTernal? (Query Only)

Returns all external trigger parameters.

Group Trigger

Syntax

```
TRIGger:EXTernal?
```

TRIGger:EXTernal:PRObe

Sets or returns the attenuation factor value of the external probe connector.

Group Trigger

Syntax

```
TRIGger:EXTernal:PRObe <NR3>
TRIGger:EXTernal:PRObe?
```

Arguments `<NR3>` is the attenuation factor of the probe.

Examples `TRIGGER:EXTERNAL:PROBE?` might return `:TRIGGER:EXTERNAL:PROBE 1.0E1` for a 10X probe.

TRIGger:EXTernal:YUNIts? (Query Only)

Returns the external trigger vertical (Y) units value.

Group Trigger

Syntax `TRIGger:EXTernal:YUNIts?`

Examples `TRIGGER:EXTERNAL:YUNITS?` might return `TRIGGER:EXTERNAL:YUNITS "V"` if the vertical unit is volts.

TRIGger:FREQuency? (Query Only)

Returns the available trigger frequency in hertz. If the trigger frequency is not currently available, the IEEE Not A Number (NaN = 9.9100E+37) value is returned. The maximum precision of the returned frequency is 6 digits.

Group Trigger

Syntax `TRIGger:FREQuency?`

TRIGger:STATE? (Query Only)

Returns the current state of the triggering system: `ARMED, AUTO, READY, SAVE` or `TRIGGER`.

Group Trigger

Syntax `TRIGger:STATE?`

Related Commands TRIGger:A:MODe

Returns `ARMED` indicates that the oscilloscope is acquiring pretrigger information.

AUTO indicates that the oscilloscope is in the automatic mode and acquires data even in the absence of a trigger.

READY indicates that all pretrigger information has been acquired and that the oscilloscope is ready to accept a trigger.

SAVE indicates that the oscilloscope is in save mode and is not acquiring data.

TRIGGER indicates that the oscilloscope triggered and is acquiring the post trigger information.

Examples

TRIGGER:STATE? might return :TRIGGER:STATE ARMED indicating that the pretrigger data is being acquired.

*TST? (Query Only)

Tests (self-test) the interface and returns a 0.

Group

Miscellaneous

Syntax

*TST?

Examples

*TST? always returns 0.

UNLock (No Query Form)

Unlocks the front panel. The command is equivalent to LOCk NONe.

Group

Miscellaneous

Syntax

UNLock ALL

Related Commands

LOCk

Arguments

ALL specifies that all front-panel buttons and knobs are unlocked.

Examples

UNLOCK ALL unlocks all front-panel buttons and knobs.

USBDevice:CONFigure

Enables or disables the rear USB port for use with Pictbridge printers.

Group PictBridge

Syntax

```
USBDevice:CONFigure {DISabled|IMAge|USBTmc}
USBDevice:CONFigure?
```

Arguments

`DISabled` disables the rear USB port.

`IMAge` enables the rear USB port as an SIC device.

`USBTmc` enables the rear USB port as a USBTMC device.

Examples

`USBDEVICE:CONFIGURE?` might return `USBDEVICE:CONFIGURE USBTMC` indicating a USBTMC device.

`USBDEVICE:CONFIGURE IMAGE` enables the rear USB port as an SIC device

USBTMC

Returns the USBTMC information used by the USB hosts to determine the instrument interfaces.

Group Misc

Syntax

```
USBTMC?
```

USBTMC:PRODUCTID:DECimal? (Query Only)

Returns the product ID of the USBTMC device in decimal format.

Group Misc

Syntax

```
USBTMC:PRODUCTID:DECimal?
```

Examples

`USBTMC:PRODUCTID:DECIMAL?` might return `USBTMC:PRODUCTID:DECIMAL 882` indicating that the product ID is 882 in decimal format.

USBTMC:PRODUCTID:HEXadecimal? (Query Only)

Returns the product ID of the USBTMC device in hexadecimal format.

Group Misc

Syntax `USBTMC:PRODUCTID:HEXadecimal?`

Examples `USBTMC:PRODUCTID:HEXADECIMAL?` might return `USBTMC:PRODUCTID:HEXADECIMAL 0x0372` indicating that the product ID in hexadecimal format is 0x0372.

USBTMC:SERIALnumber? (Query Only)

Returns the serial number of the USBTMC device.

Group Misc

Syntax `USBTMC:SERIALnumber?`

Examples `USBTMC:SERIALNUMBER?` might return `USBTMC:SERAILNUMBER NO_SERIAL` indicating that there is no serial number for the instrument.

USBTMC:VENDORID:DECimal? (Query Only)

Returns the vendor ID of the USBTMC device in decimal format.

Group Misc

Syntax `USBTMC:VENDORID:DECimal?`

Examples `USBTMC:VENDORID:DECIMAL?` might return `USBTMC:VENDORID:DECIMAL 1689` indicating that 1689 is the Vendor ID in decimal format.

USBTMC:VENDORID:HEXadecimal? (Query Only)

Returns the vendor ID of the USBTMC device in hexadecimal format.

Group Misc

Syntax `USBTMC:VENDORID:HEXadecimal?`

Examples `USBTMC:VENDORID:HEXADECIMAL?` might return `USBTMC:VENDORID:HEXADECIMAL 0x0699` indicating that 0x0689 is the Vendor ID in hexadecimal format.

VERBose

Sets or returns the Verbose state that controls the length of keywords on query responses. Keywords can be both headers and arguments.

NOTE. *This command does not affect IEEE Std 488.2-1987 Common Commands (those starting with an asterisk). This command does affects the verbose state of both the USBTMC and VXI-11 interfaces. Refer to the Introduction for information.*

Group Miscellaneous

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

Related Commands HEADer, *LRN?, SET?

Arguments `OFF` sets the Verbose state to false, which returns minimum-length keywords for applicable setting queries.

`ON` sets the Verbose state to true, which returns full-length keywords for applicable setting queries.

`<NR1>` a 0 returns minimum-length keywords for applicable setting queries; any other value returns full-length keywords.

Examples `VERBOSE ON` enables the Verbose state.

`VERBOSE ?` might return `:VERB 0` indicating that the Verbose state is disabled.

*WAI (No Query Form)

Prevents the oscilloscope from executing further commands or queries until all pending commands that generate an OPC message are complete. This command allows you to synchronize the operation of the oscilloscope with your application program.

Group Status and Error

Syntax `*WAI`

Related Commands BUSY?, *OPC

Examples `*WAI` prevents the oscilloscope from executing any further commands or queries until all pending commands that generate an OPC message are complete.

WAVFrm? (Query Only)

Returns WFMOutpre? and CURVe? data for the waveform as specified by the DATA:SOUrce command. This command is equivalent to sending both WFMOutpre? and CURVe?, with the additional provision that the response to WAVFrm? is guaranteed to provide a synchronized preamble and curve. The source waveform, as specified by :DATA:SOURCE, must be active or the query will not return any data and will generate an error indicator.

NOTE. *For MSO models, reduced resolution waveform data is not available for the digital channels.*

Group Waveform Transfer

Syntax `WAVFrm?`

Related Commands CURVe, DATa:SOUrce, WFMOutpre?

Examples `WAVFRM?` might return the waveform data as: `:WFMOUTPRE:BYT_NR 1;BIT_NR 8;ENCDG ASCII;BN_FMT RI;BYT_OR MSB;WFID “Ch1, DC coupling, 100.0mV/div, 4.000us/div, 10000 points, Sample mode”;NR_PT 20;PT_FMT Y;XUNIT “s”;XINCR 4.0000E-9;XZERO -20.0000E-6;PT_OFF 0;YUNIT`

```
"V";YMULT 4.0000E-3;YOFF 0.0000;YZERO 0.0000;:CURVE
2,1,4,2,4,3,0,3,3,3,3,3,3,4,3,5,6,6,7,3
```

WFMInpre? (Query Only)

Returns the waveform formatting and scaling specifications to be applied to the next incoming CURVe command data.

Group Waveform Transfer

Syntax

```
WFMInpre?
```

Related Commands WFMOutpre?

Examples `WFMINPRE?` might return the waveform formatting as `:WFMINPRE:BIT_NR 8;BN_FMT RI;BYT_NR 1; BYT_OR MSB;ENCDG BIN;NR_PT 500;PT_FMT Y; PT_OFF 0;XINCR 2.0000E-6;XZERO 1.7536E-6; XUNIT "s";YMULT 1.0000E-3;YOFF 0.0000; YZERO 0.0000;YUNIT "V"`

WFMInpre:BIT_Nr

Sets or returns the number of bits per binary waveform point for the incoming waveform. Changing the value of WFMInpre:BIT_Nr also changes the value of WFMInpre:BYT_Nr.

Group Waveform Transfer

Syntax

```
WFMInpre:BIT_Nr <NR1>
WFMInpre:BIT_Nr?
```

Related Commands WFMInpre:BYT_Nr

Arguments `<NR1>` number of bits per data point can be 8 or 16.

Examples `WFMINPRE:BIT_NR 16` sets the number of bits per waveform point to 16, for incoming data.

`WFMINPRE:BIT_NR?` might return `:WFMINPRE:BIT_NR 8` indicating that incoming waveform data uses 8 bits per waveform point.

WFMInpre:BN_Fmt

Sets or returns the format of binary data for incoming waveforms.

Group Waveform Transfer

Syntax

```
WFMInpre:BN_Fmt {RI|RP}
WFMInpre:BN_Fmt?
```

Related Commands WFMOutpre:BN_Fmt

Arguments `RI` specifies signed integer data point representation.

`RP` specifies positive integer data point representation.

Examples `WFMINPRE:BN_FMT RP` specifies positive integer data point representation.

`WFMINPRE:BN_FMT?` might return `:WFMINPRE:BN_FMT RI` indicating that the incoming data is currently interpreted as signed integers.

WFMInpre:BYT_Nr

Sets or returns the data width for the incoming waveform. Changing the value of WFMInpre:BYT_Nr also changes the value of WFMInpre:BIT_Nr.

Group Waveform Transfer

Syntax

```
WFMInpre:BYT_Nr <NR1>
WFMInpre:BYT_Nr?
```

Related Commands WFMInpre:BIT_Nr

Arguments `<NR1>` is the number of bytes per data point and can be 1 or 2.

Examples `WFMINPRE:BYT_NR 1` sets the number of bytes per incoming waveform data point to 1, which is the default setting.

`WFMINPRE:BYT_NR?` might return `:WFMINPRE:BYT_NR 2` indicating that there are 2 bytes per incoming waveform data point.

WFMInpre:BYT_Or

Sets or returns which byte of binary waveform data is expected first for incoming waveform data when data points require than one byte. This specification only has meaning when WFMInpre:ENCdg is set to BIN and WFMInpre:BYT_Nr is 2.

Group Waveform Transfer

Syntax

```
WFMInpre:BYT_Or {LSB|MSB}
WFMInpre:BYT_Or?
```

Related Commands WFMInpre:ENCdg, WFMInpre:BYT_Nr, WFMInpre:FILTERFreq

Arguments

`LSB` specifies that the least significant byte will be expected first.

`MSB` specifies that the most significant byte will be expected first.

Examples

`WFMINPRE:BYT_OR MSB` sets the most significant incoming byte of incoming waveform data to be expected first.

`WFMINPRE:BYT_OR?` might return `:WFMINPRE:BYT_OR LSB` indicating that the least significant incoming CURVe data byte will be expected first.

WFMInpre:COMPosition

Sets and queries the type of waveform data to be transferred to the instrument with the CURVe command.

Group Waveform Transfer

Syntax

```
WFMInpre:COMPosition
{COMPOSITE_YT|COMPOSITE_ENV|SINGULAR_YT}
WFMInpre:COMPosition?
```

Related Commands DATa:COMPosition

Arguments

`COMPOSITE_YT` uses the sample part of the composite waveform.

`COMPOSITE_ENV` uses the peak-detect part of the composite waveform.

`SINGULAR_YT` uses the sample part of the singular waveform.

Examples

`WFMINPRE:COMPOSITION?` might return `:WFMINPRE:COMPOSITION SINGULAR_YT` indicating that the sample part of singular waveform is transferred to the instrument.

`WFMINPRE:COMPOSITIONCOMPOSITE_YT` sets the sample part of composite waveform to be transferred to the instrument.

WFMInpre:ENCdg

Sets or returns the type of encoding for incoming waveform data.

Group

Waveform Transfer

Syntax

```
WFMInpre:ENCdg {ASCii|BINary}
WFMInpre:ENCdg?
```

Related Commands

WFMOutpre:ENCdg

Arguments

`ASCii` specifies that the incoming data is in ASCII format.

`BINary` specifies that the incoming data is in a binary format whose further interpretation requires knowledge of BYT_NR, BIT_NR, BN_FMT, and BYT_OR.

Examples

`WFMINPRE:ENCDG ASC` sets the format of incoming waveform data to ASCII format.

`WFMINPRE:ENCDG ?` might return `:WFMINPRE:ENCDG BIN` indicating that the incoming waveform data is in binary format.

WFMInpre:FILTERFreq

Specifies or returns the FilterVu low pass filter frequency, which was applied to the waveform being sent to the instrument.

Group

Waveform Transfer

Syntax

```
WFMInpre:FILTERFreq {NR1}
WFMInpre:FILTERFreq?
```

Arguments

`<NR1>` is an signed integer.

Examples

`WFMINPRE:FILTERFREQ?` might return `:WFMINPRE:FILTERFREQ 200000000` indicating that the FilterVu frequency for the incoming waveform is 200 MHz.

`WFMINPRE:FILTERFREQ 3000000000` sets the FilterVu frequency for the incoming waveform to 300 MHz.

WFMInpre:NR_Pt

Sets or returns the number of data points that are in the incoming waveform record.

Group

Waveform Transfer

Syntax

```
WFMInpre:NR_Pt <NR1>
WFMInpre:NR_Pt?
```

Related Commands

CURVe, DATa, DATa:STARt, DATa:STOP, SAVe:WAVEform, SAVe:WAVEform:FILEFormat, WFMOutpre:NR_Pt?

Arguments

`<NR1>` is the number of data points if WFMInpre:PT_Fmt is set to Y. It is the number of min-max pairs if WFMInpre:PT_Fmt is set to ENV.

Examples

`WFMINPRE:NR_PT 10000` specifies that 10000 data points will be expected.

`WFMINPRE:NR_PT ?` might return `:WFMINPRE:NR_PT 10000` indicating that there are 10000 data points in the expected incoming waveform record.

WFMInpre:PT_Fmt

Sets or returns the point format of the incoming waveform data. Regardless of the argument used, the scale, offset, and so on are interpreted similarly. When ENV is used, waveform data is interpreted over the min-max pair; when Y is used, it is interpreted over a single point.

Group

Waveform Transfer

Syntax

```
WFMInpre:PT_Fmt {ENV|Y}
WFMInpre:PT_Fmt?
```

Related Commands

WFMOutpre:PT_Fmt?

Arguments

`ENV` specifies that the waveform is transmitted in envelope mode as maximum and minimum point pairs. Only `Y` values are explicitly transmitted. Absolute coordinates are given by:

Xn = XZEro + XINcr (n – PT_Off)

Ynmax = YZEro + YMUlt (ynmax – YOFf)

Ynmin = YZEro + YMUlt (ynmin – 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:

Xn = XZEro + XINcr (n – PT_Off)

Yn = YZEro + YMUlt (yn – YOFf)

Examples

`WFMINPRE:PT_FMT ENV` sets the incoming waveform data point format to enveloped.

`WFMINPRE:PT_FMT?` might return `:WFMINPRE:PT_FMT ENV` indicating that the waveform is transmitted as maximum and minimum point pairs.

WFMInpre:PT_Off

The set form of this command is ignored. The query form always returns a 0. This command is listed for compatibility with other Tektronix oscilloscopes.

Group

Waveform Transfer

Syntax

```
WFMInpre:PT_Off <NR1>
WFMInpre:PT_Off?
```

Arguments

Arguments are ignored.

WFMInpre:XINcr

Sets or returns the horizontal interval between incoming waveform points in units specified by WFMInpre:XUNit.

Group

Waveform Transfer

Syntax

```
WFMInpre:XINcr <NR3>
WFMInpre:XINcr?
```

Related Commands

WFMInpre:XUNit, WFMOutpre:XINcr?

Arguments

`<NR3>` is the horizontal interval representation.

Examples

`WFMINPRE:XINCR 3E-3` sets the interval between incoming waveform points to 3 ms.

`WFMINPRE:XINCR ?` might return `:WFMINPRE:XINCR 1.0000E-3` indicating that if WFMInpre:XUNit is set to "s", there is a 1 ms interval between incoming waveform points.

WFMInpre:XUNit

Sets or returns the horizontal units of the incoming waveform.

Supported units are:
%, /Hz, A, A/A, A/V, A/W, A/dB, A/s, AA, AW, AdB, As, B, Hz, IRE, S/s, V, V/A, V/V, V/W, V/dB, V/s, VV, VW, VdB, Volts, Vs, W, W/A, W/V, W/W, W/dB, W/s, WA, WV, WW, WdB, Ws, dB, dB/A, dB/V, dB/W, dB/dB, dBA, dBV, dBW, dBdB, day, degrees, div, hr, min, ohms, percent, s

Group

Waveform Transfer

Syntax

```
WFMInpre:XUNit <QString>
WFMInpre:XUNit?
```

Related Commands

WFMOutpre:XUNit?

Arguments

`<QString>` contains a maximum of three alpha characters that represent the horizontal unit of measure for the incoming waveform.

Examples

`WFMINPRE:XUNIT "HZ"` specifies that the horizontal units for the incoming waveform are hertz.

`WFMINPRE:XUNIT?` might return `:WFMINPRE:XUNIT "s"` indicating that the horizontal units for the incoming waveform are seconds.

WFMInpre:XZEro

Sets or returns the position value, in XUNits, of the first sample of the incoming waveform.

Group Waveform Transfer

Syntax

```
WFMInpre:XZEro <NR3>
WFMInpre:XZEro?
```

Related Commands WFMInpre:XINcr, WFMInpre:XUNit, WFMOutpre:XZEro?

Arguments `<NR3>` argument is the floating point value of the position, in XUNits, of the first sample in the incoming waveform. If XUNits is "s", `<NR3>` is the time of the first sample in the incoming waveform.

Examples `WFMINPRE:XZERO 5.7E-6`, which indicates the trigger occurred 5.7 µs before the first sample in the waveform.

`WFMINPRE:XZERO?` might return `:WFMINPRE:XZEro -7.5000E-6` indicating that the trigger occurs 7.5 µs after the first sample in the waveform.

WFMInpre:YMUlt

Sets or returns the vertical scale factor of the incoming waveform, expressed in YUNits per waveform data point level. For one byte waveform data, there are 256 data point levels. For two byte waveform data there are 65,536 data point levels.

YMUlt, YOFf, and YZEro are used to convert waveform record values to YUNit values using the following formula (where dl is the data level; curve_in_dl is a data point in CURVe):

value_in_units = ((curve_in_dl – YOFf_in_dl) * YMUlt) + YZEro_in_units

NOTE. *For a given waveform record, YMUlt, YOFf, and YZEro have to be a consistent set, otherwise vertical cursor readouts and vertical measurements may give incorrect results.*

Group Waveform Transfer

Syntax

```
WFMInpre:YMUlt <NR3>
WFMInpre:YMUlt?
```

Related Commands DATa:DESTination, WFMInpre:BYT_Nr, WFMInpre:YUNit

Arguments `<NR3>` is the vertical scale factor per digitizing level of the incoming waveform points.

Examples `WFMINPRE:YMULT?` might return `:WFMINPRE:YMULT 40.0000E-3`, which (if YUNit is "V") indicates that the vertical scale is 40 mV/digitizing level (1V/div for 8-bit data).

`WFMINPRE:YMULT 20E-3` specifies that (if WFMInpre:YUNit is "V" and WFMInpre:BYT_Nr is 1), the vertical scale is 20 mV/digitizing level (500 mV/div).

WFMInpre:YOFf

Sets or returns the vertical position of the incoming waveform in digitizing levels. Variations in this number are analogous to changing the vertical position of the waveform.

YMUlt, YOFf, and YZEro are used to convert waveform record values to YUNit values using the following formula (where dl is the data level; curve_in_dl is a data point in CURVe):

value_in_units = ((curve_in_dl - YOFf_in_dl) * YMUlt) + YZEro_in_units

NOTE. *For a given waveform record, YMUlt, YOFf, and YZEro have to be a consistent set, otherwise vertical cursor readouts and vertical measurements may give incorrect results.*

Group Waveform Transfer

Syntax

```
WFMInpre:YOFf <NR3>
WFMInpre:YOFf?
```

Related Commands WFMInpre:BYT_Nr, WFMInpre:YMUlt, WFMOutpre:YOFf?

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

Examples `WFMINPRE:YOFF 50` specifies that the zero reference point for the incoming waveform is 50 digitizing levels (2 divisions, for 8-bit data) above the center of the data range.

`WFMINPRE:YOFF?` might return `:WFMINPRE:YOFF 25` indicating that the vertical position of the incoming waveform in digitizing levels.

WFMInpre:YUNit

Sets or returns the vertical units of the incoming waveform.

Supported units are: %, /Hz, A, A/A, A/V, A/W, A/dB, A/s, AA, AW, AdB, As, B, Hz, IRE, S/s, V, V/A, V/V, V/W, V/dB, V/s, VV, VW, VdB, Volts, Vs, W, W/A, W/V, W/W, W/dB, W/s, WA, WV, WW, WdB, Ws, dB, dB/A, dB/V, dB/W, dB/dB, dBA, dBV, dBW, dBdB, day, degrees, div, hr, min, ohms, percent, s

Group Waveform Transfer

Syntax

```
WFMInpre:YUNit <QString>
WFMInpre:YUNit?
```

Related Commands WFMOutpre:YUNit?

Arguments `<QString>` contains a maximum of three alpha characters that represent the vertical unit of measure for the incoming waveform.

Examples `WFMINPRE:YUNIT?` might return `:WFMINPRE:YUNIT "V"` indicating that the vertical units for the incoming waveform are volts.

`WFMINPRE:YUNIT "A"` specifies that the vertical units for the incoming waveform are Amperes.

WFMInpre:YZEro

Sets or returns the vertical offset of the incoming waveform in units specified by WFMInpre:YUNit. Variations in this number are analogous to changing the vertical offset of the waveform.

YMUlt, YOFf, and YZEro are used to convert waveform record values to YUNit values using the following formula (where dl is the data level; curve_in_dl is a data point in CURVe):

value_in_units = ((curve_in_dl - YOFf_in_dl) * YMUlt) + YZEro_in_units

NOTE. *For a given waveform record, YMUlt, YOFf, and YZEro have to be a consistent set, otherwise vertical cursor readouts and vertical measurements may give incorrect results.*

Group Waveform Transfer

Syntax

```
WFMInpre:YZEro <NR3>
WFMInpre:YZEro?
```

Related Commands WFMInpre:YUNit, WFMOutpre:YZEro?

Arguments <NR3> is the offset in YUNits.

Examples WFMINPRE:YZERO 1.5E+0 specifies that the zero reference point for the incoming waveform is 1.5 V below the center of the data range (given that WFMInpre:YUNit is set to V).

WFMINPRE:YZERO? might return :WFMINPRE:YZEro 7.5000E-6 indicating that the zero reference for the incoming waveform is 7.5 µV below the center of the data range (given that WFMInpre:YUNit is set to V).

WFMOutpre? (Query Only)

Returns waveform transmission and formatting parameters for the waveform specified by DATa:SOUrce command. If the waveform specified by the DATa:SOUrce command is not displayed, the oscilloscope returns only the waveform transmission parameters (BYT_Nr, BIT_Nr, ENCdg, BN_Fmt, BYT_Or).

For WFMOutpre? queries, first set DATa:RESOlution and DATa:COMPosition appropriately. For MSO/DPO2000/B math FFT queries, set them to: DATA:RESOLUTION FULL and DATA:COMPOSITION SINGULAR_YT.

For more information, see: Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments.

(See page B-1.)

Group Waveform Transfer

Syntax

```
WFMOutpre?
```

Examples WFMOUTPRE? ? might return the waveform formatting data as:

```
:WFMOUTPRE:BYT_NR 2;BIT_NR 16;ENCDG ASCII;BN_FMT RI;BYT_OR
MSB;WFID "Ch1, DC coupling, 100.0mV/div, 4.000us/div, 10000
points, Sample mode";NR_PT 10000;PT_FMT Y;XUNIT "s";XINCR
```

```
4.0000E-9;XZERO - 20.0000E-6;PT_OFF 0;YUNIT “V”;YMULT
15.6250E-6;YOFF :”6.4000E+3;YZERO 0.0000
```

WFMOutpre:BIT_Nr

Sets and returns the number of bits per waveform point that outgoing waveforms contain, as specified by the DATa:SOUrce command. Changing the value of WFMOutpre:BIT_Nr also changes the values of WFMInpre:FILTERFreq and DATa:WIDth.

Group Waveform Transfer

Syntax

```
WFMOutpre:BIT_Nr <NR1>
WFMOutpre:BIT_Nr?
```

Related Commands DATa:SOUrce, DATa:WIDth, WFMOutpre:BN_Fmt

Arguments `<NR1>` is the number of bits per data point and can be 8 or 16.

Examples `WFMOUTPRE:BIT_NR 16` sets the number of bits per waveform point to 16 for outgoing waveforms.

`WFMOUTPRE:BIT_NR?` might return `:WFMOUTPRE:BIT_NR 8` indicating that outgoing waveforms use 8 bits per waveform point.

WFMOutpre:BN_Fmt

Sets or returns the format of binary data for outgoing waveforms specified by the DATa:SOUrce command. Changing the value of WFMOutpre:BN_Fmt also changes the value of DATa:ENCdg.

Group Waveform Transfer

Syntax

```
WFMOutpre:BN_Fmt {RI|RP}
WFMOutpre:BN_Fmt?
```

Related Commands DATa:ENCdg, DATa:SOUrce

Arguments

`RI` specifies signed integer data point representation.

`RP` specifies positive integer data point representation.

Examples

`WFMOUTPRE:BN_FMT RP` specifies that outgoing waveform data will be in positive integer format.

`WFMOUTPRE:BN_FMT?` might return `:WFMOUTPRE:BN_FMT RI` indicating that the outgoing waveform data is currently in signed integer format.

WFMOutpre:BYT_Nr

Sets or returns the data width for the outgoing waveform specified by the DATa:SOUrce command. Changing WFMOutpre:BYT_Nr also changes WFMOutpre:BIT_Nr and DATa:WIDth.

Group

Waveform Transfer

Syntax

```
WFMOutpre:BYT_Nr <NR1>
WFMOutpre:BYT_Nr?
```

Related Commands

DATa:SOUrce, DATa:WIDth, WFMOutpre:BIT_Nr

Arguments

`<NR1>` is the number of bytes per data point and can be 1 or 2.

Examples

`WFMOUTPRE:BYT_NR 1` sets the number of bytes per outgoing waveform data point to 1, which is the default setting.

`WFMOUTPRE:BYT_NR?` might return `:WFMOUTPRE:BYT_NR 2` indicating that there are 2 bytes per outgoing waveform data point.

WFMOutpre:BYT_Or

Sets or returns which byte of binary waveform data is transmitted first, during a waveform data transfer, when data points require than one byte. This specification only has meaning when WFMOutpre:ENCdg is set to BIN and WFMOutpre:BYT_Nr is 2. Changing WFMInpre:FILTERFreq also changes DATa:ENCdg (if DATa:ENCdg is not ASCIi).

Group

Waveform Transfer

Syntax

```
WFMOutpre:BYT_Or {LSB|MSB}
WFMOutpre:BYT_Or?
```

Related Commands

WFMOutpre:ENCdg, WFMOutpre:BYT_Nr

Arguments

`LSB` specifies that the least significant byte will be transmitted first.

`MSB` specifies that the most significant byte will be transmitted first.

Examples

`WFMOUTPRE:BYT_OR MSB` sets the most significant outgoing byte of waveform data to be transmitted first.

`WFMOUTPRE:BYT_OR?` might return `:WFMOUTPRE:BYT_OR LSB` indicating that the least significant data byte will be transmitted first.

WFMOutpre:COMPosition

Controls or indicates the type of data used by the CURVe? query.

Group

Waveform Transfer

Syntax

```
WFMOutpre:COMPosition
{COMPOSITE_YT|COMPOSITE_ENV|SINGULAR_YT}
WFMOutpre:COMPosition?
```

Arguments

`COMPOSITE_YT` uses the sample part of the composite waveform.

`COMPOSITE_ENV` uses the peak-detect part of the composite waveform.

`SINGULAR_YT` uses the sample part of the singular waveform.

Examples

`WFMOUTPRE:COMPOSITION?` might return `:WFMOUTPRE:COMPOSITION SINGULAR_YT` indicating that the singular data type is used by the CURVe? query.

`WFMOUTPRE:COMPOSITIONCOMPOSITE_YT` sets the data type used by the CURVe? query.

WFMOutpre:ENCdg

Sets and queries the type of encoding for outgoing waveforms.

Group

Waveform Transfer

Syntax

```
WFMOutpre:ENCdg {ASCii|BINary}
WFMOutpre:ENCdg?
```

Related Commands

DATa:ENCdg, WFMOutpre:BYT_Nr, WFMInpre:FILTERFreq, WFMOutpre:BIT_Nr, WFMOutpre:BN_Fmt

Arguments

`ASCii` specifies that the outgoing data is to be in ASCII format. Waveforms will be sent as `<NR1>` numbers.

`BINary` specifies that outgoing data is to be in a binary format whose further specification is determined by WFMOutpre:BYT_Nr, WFMOutpre:BIT_Nr, WFMOutpre:BN_Fmt and WFMInpre:FILTERFreq.

Examples

`WFMOUTPRE:ENCDG?` might return `:WFMOUTPRE:ENCDG BIN` indicating that outgoing waveform data will be sent in binary format.

`WFMOUTPRE:ENCDG ASC` specifies that the outgoing waveform data will be sent in ASCII format.

WFMOutpre:FILTERFreq? (Query Only)

Returns the FilterVu low pass filter frequency applied to the LRL waveform record of the source waveform specified by DATa:SOUrce. If the DATa:RESOlution is set to FULL, then this is the filter frequency applied to the full resolution (LRL) waveform. If the DATa:RESOlution is set to REDUced, then this is the filter frequency applied to the reduced resolution (thumbnail) waveform.

Group

Waveform Transfer

Syntax

```
WFMOutpre:FILTERFreq?
```

Examples

`WFMOUTPRE:FILTERFREQ?` might return `:WFMOUTPRE:FILTERFREQ 200000000` indicating that the FilterVu frequency for the outgoing waveform is 200 MHz.

WFMOutpre:NR_Pt? (Query Only)

Returns the number of points for the DATa:SOUrce waveform that will be transmitted in response to a CURVe? query. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

Group Waveform Transfer

Syntax `WFMOutpre:NR_Pt?`

Related Commands CURVe, DATa, DATa:STARt, DATa:STOP, SAVe:WAVEform, SAVe:WAVEform:FILEFormat, WFMInpre:NR_Pt

Examples `WFMOUTPRE:NR_PT?` might return `:WFMOUTPRE:NR_PT 10000` indicating that there are 10000 data points to be sent.

WFMOutpre:PT_Fmt? (Query Only)

Returns the point format for the outgoing waveform specified by the DATa:SOUrce command. Returned values are either ENV, which indicates envelope mode format in which the data is returned as a series of min/max pairs, or Y, which indicates normal waveform points. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

Group Waveform Transfer

Syntax `WFMOutpre:PT_Fmt?`

Related Commands CURVe, DATa:SOUrce

Examples `WFMOUTPRE:PT_FMT?` might return `:WFMOutpre:PT_Fmt ENV` indicating that the waveform data is a series of min-max pairs.

WFMOutpre:PT_Off? (Query Only)

The set form of this command is ignored. The query form always returns a 0, if the waveform specified by DATA:SOUrce is on or displayed. If the waveform is not displayed, the query form generates an error and returns event code 2244. This command is for compatibility with other Tektronix oscilloscopes.

Group Waveform Transfer

Syntax `WFMOutpre:PT_Off?`

Related Commands DATa:SOUrce

Arguments Arguments are ignored.

Examples `WFMOUTPRE:PT_OFF?` might return `WFMOUTPRE:PT_OFF 0` indicating that the waveform specified by `DATA:SOURCE` is on or displayed.

WFMOutpre:PT_ORder? (Query Only)

This query is for compatibility with other Tektronix oscilloscopes and always returns LINEAR.

Group Waveform Transfer

Syntax `WFMOutpre:PT_ORder?`

Related Commands DATa:SOUrce

Examples `WFMOUTPRE:PT_ORDER?` returns `:WFMOUTPRE:PT_ORDER LINEAR`.

WFMOutpre:RECOrdlength? (Query Only)

Returns the record length for the source waveform as specified by the DATa:SOUrce command.

Group Waveform Transfer

Syntax `WFMOutpre:RECOrdlength?`

Examples `WFMOUTPRE:RECORDLENGTH?` might return `WFMOUTPRE:RECORDLENGTH 6250` indicating that 6250 is the source waveform record length.

WFMOutpre:WFId? (Query Only)

Returns a string describing several aspects of the acquisition parameters for the waveform specified by the DATa:SOUrce command. The query command will

time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

Group Waveform Transfer

Syntax `WFMOutpre:WFId?`

Related Commands DATa:SOUrce

Returns `<QString>` comprises the following comma-separated fields documented in the tables below:

Table 2-42: Waveform Suffixes

Field	Description	Examples
Source	The source identification string as it appears in the front-panel scale factor readouts.	“CH1–4” “Math1 “Ref1–2"
Coupling	A string describing the vertical coupling of the waveform (the Source1 waveform in the case of Dual Waveform Math).	“AC coupling” “DC couplng” “GND coupling”
Vert Scale	A string containing the vertical scale factor of the unzoomed waveform. The numeric portion will always be four digits. The examples cover all known internal units.	“100.0 mV/div” “20.00 dB/div” “45.00 deg/div” “785.4 mrad/div” “500.0 µVs/div” “10.00 kV/s/div” “200.0 mV/div” “50.00 unk/div”
Horiz Scale	A string containing the horizontal scale factor of the unzoomed waveform. The numeric portion will always be four digits. The examples cover all known internal units.	“100 ms/div” “10.00 kHz/div”
Record Length	A string containing the number of waveform points available in the entire record. The numeric portion is given as an integer.	“1000 points” “1000000 points”
Acquisition Mode	A string describing the mode used to acquire the waveform.	“Sample mode” “Pk Detect mode” “Envelope mode” “Average mode”

Examples `WFMOUTPRE:WFID?` might return `:WFMOUTPRE:WFID "Ch1, DC coupling,100.0mVolts/div,500.0µs/div, 1000 points, Sample mode"`

WFMOutpre:XINcr? (Query Only)

Returns the horizontal point spacing in units of WFMOutpre:XUNit for the waveform specified by the DATa:SOUrce command. This value corresponds to the sampling interval. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

Group Waveform Transfer

Syntax `WFMOutpre:XINcr?`

Related Commands DATa:SOUrce, WFMOutpre:XUNit?

Examples `WFMOUTPRE:XINCR?` might return `:WFMOUTPRE:XINCR 10.0000E-6` indicating that the horizontal sampling interval is 10 µs/point.

WFMOutpre:XUNit? (Query Only)

Returns the horizontal units for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

Group Waveform Transfer

Syntax `WFMOutpre:XUNit?`

Related Commands DATa:SOUrce

Examples `WFMOUTPRE:XUNIT?` might return `:WFMOUTPRE:XUNIT "HZ"` indicating that the horizontal units for the waveform are in Hertz.

WFMOutpre:XZEro? (Query Only)

Returns the time coordinate of the first point in the outgoing waveform.

This value is in units of WFMOutpre:XUNit?. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

Group Waveform Transfer

Syntax `WFMOutpre:XZEro?`

Related Commands DATa:SOUrce, WFMOutpre:XUNit?

Examples `WFMOUTPRE:XZERO?` might return `:WFMOUTPRE:XZERO 5.6300E-9` indicating that the trigger occurred 5.63 ns before the first sample in the waveform record.

WFMOutpre:YMUlt? (Query Only)

Returns the vertical scale factor per digitizing level in units specified by WFMOutpre:YUNit for the waveform specified by the DATa:SOUrce command. The query command will time out and an error is generated if the waveform specified by DATa:SOUrce is not turned on.

See the description of WFMInpre:YMUlt to see how this scale factor is used to convert waveform sample values to volts.

Group Waveform Transfer

Syntax `WFMOutpre:YMUlt?`

Related Commands DATa:SOUrce, WFMInpre:YMUlt

Examples `WFMOUTPRE:YMULT?` might return `:WFMOUTPRE:YMULT 4.0000E-3` indicating that the vertical scale for the corresponding waveform is 100 mV/div (for 8-bit waveform data).

WFMOutpre:YOFf? (Query Only)

Returns the vertical position in digitizing levels for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

See the description of WFMInpre:YOFf to see how this position is used to convert waveform sample values to volts.

Group Waveform Transfer

Syntax `WFMOutpre:YOFf?`

Related Commands DATa:SOUrce, WFMOutpre:BYT_Nr

Examples `WFMOUTPRE:YOFF?` might return `:WFMOUTPRE:YOFF -50.0000E+0` indicating that the position indicator for the waveform was 50 digitizing levels (2 divisions) below center screen (for 8-bit waveform data).

WFMOutpre:YUNit? (Query Only)

Returns the vertical units for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

Group Waveform Transfer

Syntax `WFMOutpre:YUNit?`

Related Commands DATa:SOUrce

Examples `WFMOUTPRE:YUNIT?` might return `:WFMOUTPRE:YUNIT "dB"` indicating that the vertical units for the waveform are measured in decibels.

WFMOutpre:YZEro? (Query Only)

Returns the vertical offset in units specified by WFMOutpre:YUNit? for the waveform specified by the DATa:SOUrce command. The query command will time out and an error will be generated if the waveform specified by DATa:SOUrce is not turned on.

See the description of WFMInpre:YZEro to see how this offset is used to convert waveform sample values to volts.

Group Waveform Transfer

Syntax `WFMOutpre:YZEro?`

Related Commands DATa:SOUrce, WFMOutpre:YUNit?

Examples `WFMOUTPRE:YZERO?` might return `:WFMOUTPRE:YZERO -100.0000E-3` indicating that vertical offset is set to –100 mV.

ZOOm? (Query Only)

Returns the current vertical and horizontal positioning and scaling of the display.

Group Zoom

Syntax `ZOOm?`

Examples `ZOOM?` might return `:ZOOM:MODE 1;GRATICULE:SIZE 80;SPLIT EIGHTYTWENTY;:ZOOM:ZOOM1:STATE 1;SCALE 400.0000E-12;POSITION 46.8986;FACTOR 50.0000E+3;HORIZONTAL:POSITION 46.8986;SCALE 400.0000E-12`

ZOOm{:MODe|:STATE}

Turns Zoom mode on or off. The Zoom mode query returns the current state of Zoom mode. This command is equivalent to pressing the zoom button located on the front panel.

Group Zoom

Syntax

```
ZOOm{:MODe|:STATE} {ON|OFF|<NR1>}
ZOOm{:MODe|:STATE}
```

Arguments `ON` turns on Zoom mode.

`OFF` turns off Zoom mode.

`<NR1>` = 0 turns off Zoom mode; any other value turns on Zoom mode.

Examples `ZOOM:MODE OFF` turns off Zoom mode.

`ZOOM:MODE?` might return `:ZOOM:MODE 1` indicating that Zoom mode is currently turned on.

ZOOm:ZOOM1? (Query Only)

Returns the current horizontal positioning and scaling of the display.

Group Zoom

Syntax `ZOOm:ZOOM1?`

Examples `ZOOM:ZOOM1?` might return `:ZOOM:ZOOM1:STATE 1;SCALE 400.0000E-12;POSITION 46.8986;FACTOR 50.0000E+3;HORIZONTAL:POSITION 46.8986;SCALE 400.0000E-12`.

ZOOm:ZOOM1:FACtor? (Query Only)

Returns the zoom factor of a particular zoom box.

Group Zoom

Syntax `ZOOm:ZOOM1:FACtor?`

Returns `<NR1>` is the zoom factor of a zoom box.

ZOOm:ZOOM1:HORizontal:POSition

Sets or returns the horizontal position for the specified zoom.

Group Zoom

Syntax
```
ZOOm:ZOOM1:HORizontal:POSition <NR3>
ZOOm:ZOOM1:HORizontal:POSition?
```

Arguments `<NR3>` is a value from 0 to 100.00 and is the percent of the upper window that is to the left of screen center, when the zoom factor is 1× or greater.

Examples `ZOOM:ZOOM1:HORIZONTAL:POSITION 50` sets the Zoom1 reference pointer at 50% of acquired waveform.

`ZOOM:ZOOM1:HORIZONTAL:POSITION?` might return `:ZOOM1:HORIZONTAL:POSITION 50.0000` indicating that the Zoom1 reference pointer is currently set at 50% of acquired waveform.

ZOOm:ZOOM1:HORizontal:SCAle

Sets or returns the zoom horizontal scale for the specified zoom.

Group Zoom

Syntax

```
ZOOm:ZOOM1:HORizontal:SCAle <NR3>
ZOOm:ZOOM1:HORizontal:SCAle?
```

Arguments `<NR3>` is the amount of expansion in the horizontal direction and ranges from 1.0E-3 to 5.0.

Examples `ZOOM:ZOOM1:HORIZONTAL:SCALE 5` sets the horizontal scale to 5 seconds.

`ZOOM:ZOOM2:HORIZONTAL:SCALE?` might return `:ZOOM2:HORIZONTAL:SCALE 1`, indicating that the horizontal scale is 1 second.

ZOOm:ZOOM1:POSition

Sets or returns the horizontal position for the specified zoom.

Group Zoom

Syntax

```
ZOOm:ZOOM1:POSition <NR3>
ZOOm:ZOOM1:POSition?
```

Arguments `<NR3>` is a value from 0 to 100.00 and is the percent of the upper window that is to the left of screen center, when the zoom factor is 1× or greater.

ZOOm:ZOOM1:SCAle

Sets or returns the zoom horizontal scale for the specified zoom.

Group Zoom

Syntax

```
ZOOm:ZOOM1:SCAle <NR3>
ZOOm:ZOOM1:SCAle?
```

Arguments

<NR3> is the amount of expansion in the horizontal direction and ranges from 1.0E-3 to 5.0.

ZOOm:ZOOM1:STATE

Sets or returns the specified zoom on or off, where x is the integer 1 representing the single zoom window.

Group

Zoom

Syntax

```
ZOOm:ZOOM1:STATE {ON|OFF|<NR1>}
ZOOm:ZOOM1:STATE?
```

Arguments

ON turns Zoom 1 on.

OFF turns Zoom 1 off.

<NR1> = 0 disables the specified zoom; any other value enables the specified zoom.

Examples

ZOOM:ZOOM1:STATE ON turns Zoom1 on.

ZOOM:ZOOM1:STATE? might return :ZOOM:ZOOM1:STATE 1 indicating that Zoom1 is on.

Status and Events

The oscilloscope provides a status and event reporting system for the Ethernet, GPIB (with the TEK-USB-488 Adapter), and USB interfaces. This system informs you of certain significant events that occur within the oscilloscope.

The oscilloscope status handling system consists of five 8-bit registers and two queues for each interface. The remaining Status subtopics describe these registers and components. They also explain how the event handling system operates.

Registers

Overview

The registers in the event handling system fall into two functional groups:

- Status Registers contain information about the status of the oscilloscope. They include the Standard Event Status Register (SESR).
- 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 oscilloscope is in use. IEEE Std 488.2-1987 defines these registers.

Each bit in a Status Register records a particular type of event, such as an execution error or message available. When an event of a given type occurs, the oscilloscope sets the bit that represents that type of event to a value of one. (You can disable bits so that they ignore events and remain at zero. See Enable Registers). Reading the status registers tells you what types of events have occurred.

The Standard Event Status Register (SESR). The SESR records eight types of events that can occur within the oscilloscope. Use the *ESR? query to read the SESR register. Reading the register clears the bits of the register so that the register can accumulate information about new events.

NOTE. *TekVISA applications use SESR bit 6 to respond to any of several events, including some front panel actions.*

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 oscilloscope was powered on. On completion, the diagnostic self tests also set this bit.
6	URQ	`User Request.` Indicates that an application event has occurred. *See note.
5	CME	`Command Error.` Shows that an error occurred while the oscilloscope was parsing a command or query.
4	EXE	`Execution Error.` Shows that an error executing a command or query.
3	DDE	`Device Error.` Shows that a device error occurred.
2	QYE	`Query Error.` 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.` This is not used.
0 (LSB)	OPC	`Operation Complete.` Shows that the operation is complete. This bit is set when all pending operations complete following an *OPC command.

The Status Byte Register (SBR). Records whether output is available in the Output Queue, whether the oscilloscope requests service, and whether the SESR has recorded any events.

Use a Serial Poll or the *STB? query to read the contents of the SBR. The bits in the SBR are set and cleared depending on the contents of the SESR, the Event Status Enable Register (ESER), and the Output Queue. When you use a Serial Poll to obtain the SBR, bit 6 is the RQS bit. When you use the *STB? query to obtain the SBR, bit 6 is the MSS bit. Reading the SBR does not clear the bits.

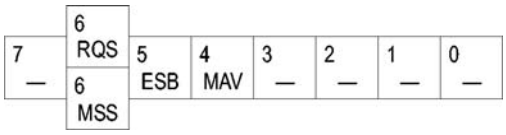


Figure 3-2: The Status Byte Register (SBR)

Table 3-2: SBR Bit Functions

Bit	Function	
7 (MSB)	———	Not used.
6	RQS	`Request Service.` Obtained from a serial poll. Shows that the oscilloscope requests service from the GPIB controller.
6	MSS	`Master Status Summary.` Obtained from *STB? query. Summarizes the ESB and MAV bits in the SBR.
5	ESB	`Event Status Bit.` Shows that status is enabled and present in the SESR.

Table 3-2: SBR Bit Functions (cont.)

Bit	Function	
4	MAV	`Message Available.` Shows that output is available in the Output Queue.
3	———	Not used.
2	———	Not used.
1–0	———	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 a 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). 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.

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). This register 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). This register 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 the register. 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)

*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

Output Queue

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

CAUTION. *When a controller sends a query, an <EOM>, and a second query, the oscilloscope normally clears the first response and outputs the second while reporting a Query Error (QYE bit in the ESER) to indicate the lost response. A fast controller, however, may receive a part or all 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.*

Event Queue

The Event Queue stores detailed information on up to 128 events. If more than 128 events stack up in the Event Queue, the 128th event is replaced by event code 350, "Queue Overflow."

Read the Event Queue with the EVENT? query (which returns only the event number), with the EVMSG? query (which returns the event number and a text description of the event), or with the ALLEV? query (which returns all the event numbers along with a description of the event). Reading an event removes it from the queue.

Before reading an event from the Event Queue, you must use the *ESR? query to read the summary of the event from the SESR. This makes the events summarized by the *ESR? read available to the EVENT? and EVMSG? queries, and empties the SESR.

Reading the SESR erases any events that were summarized by previous *ESR? reads but not read from the Event Queue. Events that follow an *ESR? read are put in the Event Queue but are not available until *ESR? is used again.

Event Handling Sequence

The figure below shows how to use the status and event handling system. In the explanation that follows, numbers in parentheses refer to numbers in the figure.

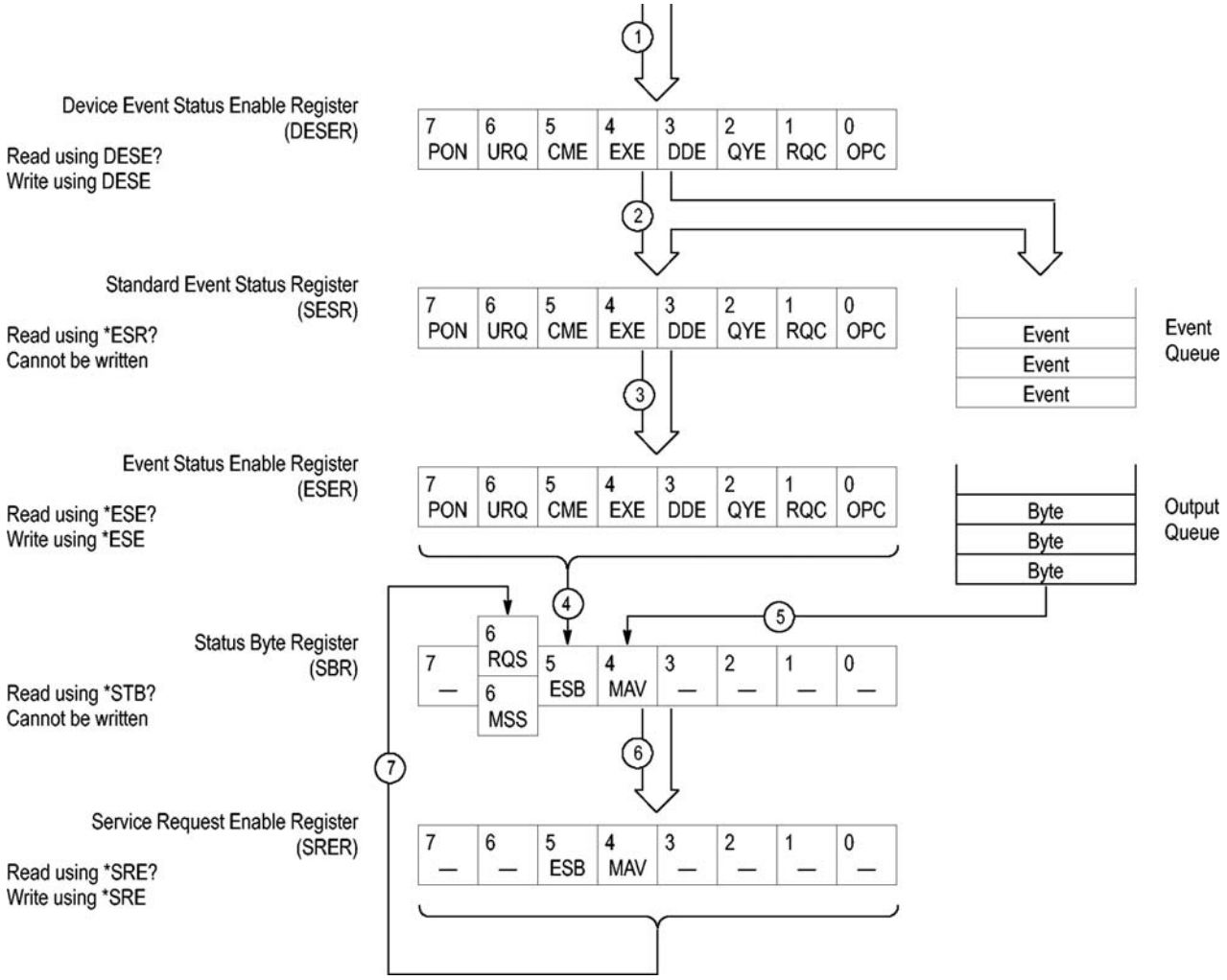


Figure 3-6: Status and Event Handling Process

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

Overview

Although most commands are completed almost immediately after being received by the oscilloscope, some commands start a process that requires time. For example, once a single sequence acquisition command is executed, depending upon the applied signals and trigger settings, it may take an indeterminate amount of time before the acquisition is complete. Rather than remain idle while the operation is in process, the oscilloscope will continue processing other commands. This means that some operations will not be completed in the order that they were sent.

Sometimes the result of an operation depends on the result of an earlier operation. A first operation must complete before the next one is processed. The oscilloscope status and event reporting system is designed to accommodate this process.

The Operation Complete (OPC) bit of the Standard Event Status Register (SESR) can be programmed to indicate when certain oscilloscope operations have completed and, by setting the Event Status Enable Register (ESER) to report OPC in the Event Status Bit (ESB) of the Status Byte Register (SBR) and setting the Service Request Enable Register (SRER) to generate service request upon a positive transition of the ESB, a service request (SRQ) interrupt can be generated when certain operations complete as described in this section.

The following oscilloscope operations can generate an OPC:

- :ACQuire:STATE <non-zero nr1> | ON | RUN only when in single sequence acquisition mode
- :CALibrate:FACtory STARt

  :CALibrate:FACtory CONTinue

  :CALibrate:FACtory PREVious
- :HARDCopy

  :HARDCopy START
- :DIAg:STATE EXECute
- :SAVe:IMAGe <file as quoted string>
- :SAVe:SETUp <file as quoted string>
- :RECAll:SETUp <file as quoted string>
- :SAVe:WAVEform <source wwfm>,(<REF1-REF4> | <file as quoted string>
- :RECAll:WAVEform <ISF file as quoted string>,<REF1-REF4>
- :CH<1-4>:PRObe:DEGAUss EXECute
- :CH<1-4>:PRObe:AUTOZero EXECute
- TEKSecure

For example, a typical application might involve acquiring a single-sequence waveform and then taking a measurement on the acquired waveform. You could use the following command sequence to do this:

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE
```

The acquisition of the waveform requires extended processing time. It may not finish before the oscilloscope takes an amplitude measurement (see the following figure). This can result in an incorrect amplitude value.

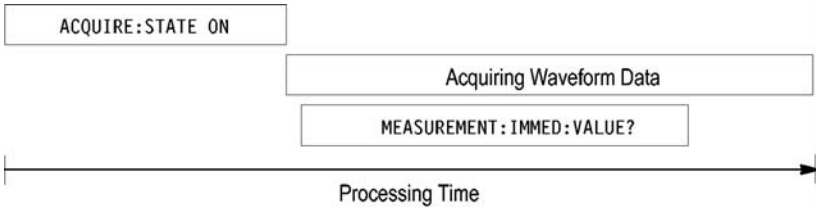


Figure 3-7: Command Processing Without Using Synchronization

To ensure the oscilloscope completes waveform acquisition before taking the measurement on the acquired data, you can synchronize the program.

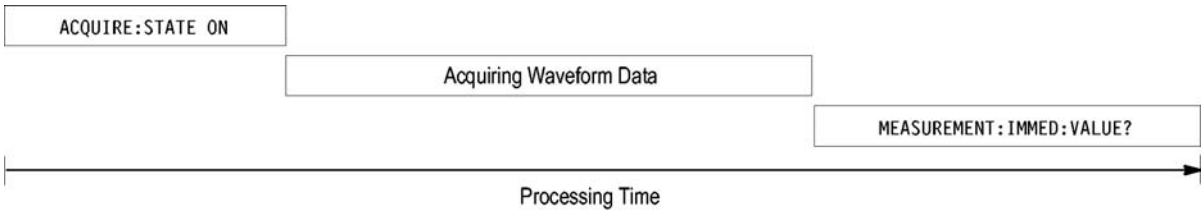


Figure 3-8: Processing Sequence With Synchronization

You can use four different methods to synchronize the operation of the oscilloscope with your application program: *WAI, BUSY, *OPC, and *OPC?. The *OPC? query method is the most simple approach.

Using the *WAI Command

The *WAI command forces completion of previous commands that generate an OPC message. No commands after the *WAI are processed before the OPC message(s) are generated.

The same command sequence using the *WAI command for synchronization looks like this:

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking
the measurement */
*WAI
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE?
```

The controller can continue to write commands to the input buffer of the oscilloscope, but the commands will not be processed by the oscilloscope until all in-process OPC operations are complete. If the input buffer becomes full, the controller will be unable to write commands to the buffer. This can cause a time-out.

Using the BUSY Query

The BUSY? query allows you to find out whether the oscilloscope is busy processing a command that generates OPC.

The same command sequence, using the BUSY? query for synchronization, looks like this:

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking
the measurement */
While BUSY keep looping
/* Take amplitude measurement */
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 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 conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Enable the status registers */
DESE 1
*ESE 1
*SRE 0
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking the
measurement.*/
*OPC
While serial poll = 0, keep looping
/* Take amplitude measurement */
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, the oscilloscope will generate a Service Request.

The same command sequence using the *OPC command for synchronization looks like this

```
/* Set up conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Enable the status registers */
DESE 1
*ESE 1
*SRE 32
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
/* Wait until the acquisition is complete before taking the
measurement*/
*OPC
```

The program can now do different tasks such as talk to other devices. The SRQ, when it comes, interrupts those tasks and returns control to this task.

```
/* Take amplitude measurement */
MEASUREMENT:IMMED:VALUE?
```

This technique is efficient but requires sophisticated programming.

Using the *OPC? Query

The *OPC? query is the most simple synchronization method. It does not require you to set registers or poll, and it does not change the operation of subsequent commands or queries.

The *OPC? query places a 1 in the Output Queue once an operation that generates an OPC message 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 conditional acquisition */
ACQUIRE:STATE OFF
SELECT:CH1 ON
ACQUIRE:MODE SAMPLE
ACQUIRE:STOPAFTER SEQUENCE
/* Acquire waveform data */
ACQUIRE:STATE ON
/* Set up the measurement parameters */
MEASUREMENT:IMMED:TYPE AMPLITUDE
MEASUREMENT:IMMED:SOURCE CH1
```

```
/* Wait until the acquisition is complete */
*OPC?
/* Take amplitude measurement */
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

Messages

The information contained in the tables below covers all the programming interface messages the oscilloscope generates in response to commands and queries.

For most messages, a secondary message from the oscilloscope gives 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, the associated SESR bit is specified in the table title, with exceptions noted with the error message text.

No Event

The following table 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?

Command Error

The following table shows the command error messages generated by improper syntax. Check that the command is properly formed and that it follows the rules in the section on command Syntax.

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

Table 3-4: Command Error Messages (CME Bit 5) (cont.)

Code	Message
108	Parameter not allowed
109	Missing parameter
110	Command header error
112	Program mnemonic too long
113	Undefined header
120	Numeric data error
121	Invalid character in numeric
123	Exponent too large
124	Too many digits
130	Suffix error
131	Invalid suffix
134	Suffix too long
140	Character data error
141	Invalid character data
144	Character data too long
150	String data error
151	Invalid string data
152	String data too long
160	Block data error
161	Invalid block data
170	Command expression error
171	Invalid expression

Execution Error

The following table lists the execution errors that are detected during execution of a command.

Table 3-5: Execution Error Messages (EXE Bit 4)

Code	Message
200	Execution error
221	Settings conflict
222	Data out of range
224	Illegal parameter value
241	Hardware missing
250	Mass storage error
251	Missing mass storage
252	Missing media

Table 3-5: Execution Error Messages (EXE Bit 4) (cont.)

Code	Message
253	Corrupt media
254	Media full
255	Directory full
256	File name not found
257	File name error
258	Media protected
259	File name too long
270	Hardcopy error
271	Hardcopy device not responding
272	Hardcopy is busy
273	Hardcopy aborted
274	Hardcopy configuration error
280	Program error
282	Insufficient network printer information
283	Network printer not responding
284	Network printer server not responding
286	Program runtime error
287	Print server not found
2200	Measurement error, Measurement system error
2201	Measurement error, Zero period
2202	Measurement error, No period, second waveform
2203	Measurement error, No period, second waveform
2204	Measurement error, Low amplitude, second waveform
2205	Measurement error, Low amplitude, second waveform
2206	Measurement error, Invalid gate
2207	Measurement error, Measurement overflow
2208	Measurement error, No backwards Mid Ref crossing
2209	Measurement error, No second Mid Ref crossing
2210	Measurement error, No Mid Ref crossing, second waveform
2211	Measurement error, No backwards Mid Ref crossing
2212	Measurement error, No negative crossing
2213	Measurement error, No positive crossing
2214	Measurement error, No crossing, target waveform
2215	Measurement error, No crossing, second waveform
2216	Measurement error, No crossing, target waveform
2217	Measurement error, Constant waveform
2219	Measurement error, No valid edge - No arm sample

Table 3-5: Execution Error Messages (EXE Bit 4) (cont.)

Code	Message
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	Measurement error, Null Waveform
2227	Measurement error, Positive and Negative Clipping
2228	Measurement error, Positive Clipping
2229	Measurement error, Negative Clipping
2230	Measurement error, High Ref < Low Ref
2231	Measurement error, No statistics available
2233	Requested waveform is temporarily unavailable
2235	Math error, invalid math description
2240	Invalid password
2241	Waveform requested is invalid
2244	Source waveform is not active
2245	Saveref error, selected channel is turned off
2250	Reference error, the reference waveform file is invalid
2253	Reference error, too many points received
2254	Reference error, too few points received
2259	File too big
2270	Alias error
2271	Alias syntax error
2273	Illegal alias label
2276	Alias expansion error
2277	Alias redefinition not allowed
2278	Alias header not found
2285	TekSecure(R) Pass
2286	TekSecure(R) Fail
2500	Setup error, file does not look like a setup file
2501	Setup warning, could not recall all values from external setup
2620	Mask error, too few points received
2760	Mark limit reached
2761	No mark present
2762	Search copy failed

Device Error

The following table lists the device errors that can occur during oscilloscope operation. These errors may indicate that the oscilloscope needs repair.

Table 3-6: Device Error Messages (DDE Bit 3)

Code	Message
310	System error
311	Memory error
312	PUD memory lost
314	Save/recall memory lost

System Event

The following table 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)
468	Knob/Keypad value changed
472	Application variable changed

Execution Warning

The following table lists warning messages that do not interrupt the flow of command execution. They also notify you of a possible unexpected results.

Table 3-8: Execution Warning Messages (EXE Bit 4)

Code	Message
528	Parameter out of range
532	Curve data too long, Curve truncated
533	Curve error, Preamble values are inconsistent
540	Measurement warning, Uncertain edge
541	Measurement warning, Low signal amplitude
542	Measurement warning, Unstable histogram

Table 3-8: Execution Warning Messages (EXE Bit 4) (cont.)

Code	Message
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: Execution Warning Messages (EXE Bit 4)

Code	Message
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 min max
546	Measurement warning, Need 3 edges
547	Measurement warning, Clipping positive/negative
548	Measurement warning, Clipping positive
549	Measurement warning, Clipping negative

Internal Warning

The following table shows internal errors that indicate an internal fault in the oscilloscope.

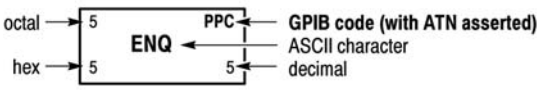
Table 3-10: Internal Warning Messages

Code	Message
630	Internal warning, 50Ω overload

Appendix A: Character Set

B7 B6 B5 BITS / B4 B3 B2 B1	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
	CONTROL		NUMBERS SYMBOLS		UPPER CASE		LOWER CASE	
0 0 0 0	0 NUL 0	20 DLE 16	40 LA0 SP 32	60 LA16 0 48	100 TA0 @ 64	120 TA16 P 80	140 SA0 ` 96	160 SA16 p 112
0 0 0 1	1 GTL SOH 1	21 LL0 DC1 17	41 LA1 ! 33	61 LA17 1 49	101 TA1 A 65	121 TA17 Q 81	141 SA1 a 97	161 SA17 q 113
0 0 1 0	2 STX 2	22 DC2 18	42 LA2 " 34	62 LA18 2 50	102 TA2 B 66	122 TA18 R 82	142 SA2 b 98	162 SA18 r 114
0 0 1 1	3 ETX 3	23 DC3 19	43 LA3 # 35	63 LA19 3 51	103 TA3 C 67	123 TA19 S 83	143 SA3 c 99	163 SA19 s 115
0 1 0 0	4 SDC EOT 4	24 DCL DC4 20	44 LA4 $ 36	64 LA20 4 52	104 TA4 D 68	124 TA20 T 84	144 SA4 d 100	164 SA20 t 116
0 1 0 1	5 PPC ENQ 5	25 PPU NAK 21	45 LA5 % 37	65 LA21 5 53	105 TA5 E 69	125 TA21 U 85	145 SA5 e 101	165 SA21 u 117
0 1 1 0	6 ACK 6	26 SYN 22	46 LA6 & 38	66 LA22 6 54	106 TA6 F 70	126 TA22 V 86	146 SA6 f 102	166 SA22 v 118
0 1 1 1	7 BEL 7	27 ETB 23	47 LA7 ' 39	67 LA23 7 55	107 TA7 G 71	127 TA23 W 87	147 SA7 g 103	167 SA23 w 119
1 0 0 0	10 GET BS 8	30 SPE CAN 24	50 LA8 (40	70 LA24 8 56	110 TA8 H 72	130 TA24 X 88	150 SA8 h 104	170 SA24 x 120
1 0 0 1	11 TCT HT 9	31 SPD EM 25	51 LA9) 41	71 LA25 9 57	111 TA9 I 73	131 TA25 Y 89	151 SA9 i 105	171 SA25 y 121
1 0 1 0	12 LF 10	32 SUB 26	52 LA10 * 42	72 LA26 : 58	112 TA10 J 74	132 TA26 Z 90	152 SA10 j 106	172 SA26 z 122
1 0 1 1	13 VT 11	33 ESC 27	53 LA11 + 43	73 LA27 ; 59	113 TA11 K 75	133 TA27 [91	153 SA11 k 107	173 SA27 { 123
1 1 0 0	14 FF 12	34 FS 28	54 LA12 , 44	74 LA28 < 60	114 TA12 L 76	134 TA28 \ 92	154 SA12 l 108	174 SA28 ¦ 124
1 1 0 1	15 CR 13	35 GS 29	55 LA13 – 45	75 LA29 = 61	115 TA13 M 77	135 TA29] 93	155 SA13 m 109	175 SA29 } 125
1 1 1 0	16 SO 14	36 RS 30	56 LA14 . 46	76 LA30 > 62	116 TA14 N 78	136 TA30 ^ 94	156 SA14 n 110	176 SA30 ~ 126
1 1 1 1	17 SI 15	37 US 31	57 LA15 / 47	77 UNL ? 63	117 TA15 O 79	137 UNT – 95	157 SA15 o 111	177 RUBOUT (DEL) 127
	ADDRESSED COMMANDS	UNIVERSAL COMMANDS	LISTEN ADDRESSES		TALK ADDRESSES		SECONDARY ADDRESSES OR COMMANDS	

KEY



Tektronix
REF: ANSI STD X3.4-1977
IEEE STD 488.1-1987
ISO STD 646-2973

Appendix B: Waveform Data in MSO/DPO2000B and MSO/DPO2000 Series Instruments

An analog channel of an MSO/DPO2000B and MSO/DPO2000 series instrument has a sample waveform and a peak detect waveform being acquired simultaneously. This combination waveform is referred to as a "composite" waveform. In some cases, for example when averaging is turned on, or for math waveforms, a composite waveform is not available. In these cases, the waveform is termed "singular", and is available only in sample mode.

Full Versus Reduced Resolution Waveforms

The MSO/DPO2000/B series introduces the concept of reduced resolution waveforms available for curve queries. The MSO/DPO2000/B oscilloscopes acquire channel data into "full" resolution records that are nominally 1 million (1M) or 100,000 (100K) points in size (the acquisition record length); these are termed "full resolution", or LRL (long record length) waveforms. These LRL records are "shrunk" (decimated) down to "thumbnail", "short", or "reduced resolution" waveforms; these are what appear on the oscilloscope's display screen. During decimation, a low-pass filter operation may be applied to the LRL sample data that results in a filtered thumbnail waveform. In general, both full and reduced resolution waveforms can be queried from the oscilloscope. The size of the reduced resolution waveform varies from 800 to 6250 points, depending on instrument settings.

Math FFT waveform data is available only in FULL resolution.

For applications which need to query waveform data rapidly for the purpose of displaying it, reduced resolution waveform data will provide a faster display update than a full resolution waveform data.

Nominal Versus Actual Record Length

The MSO/DPO2000/B series also introduces the concept of nominal versus actual acquisition record length. Like other oscilloscopes, the acquisition record length can be set using the HORizontal:RECOrdlength command. However, the actual number of points in the waveform record may differ from the nominal record length established by the HORizontal:RECOrdlength setting, depending on the HORizontal:SCAle in effect at the time of the acquisition. The valid values for setting acquisition record length are 1,000,000 (1M) and 100,000 (100K) points. Depending on the horizontal scale in effect at the time of the acquisition, a nominal 1M point record may be comprised of 1.0M or 1.25M (1,250,000) points and a nominal 100K point record may be comprised of 100K or 125K (125,000) points. The actual number of points in the waveform record for the source waveform (specified by the DATa:SOUrce command) may be obtained using the WFMOutpre:RECOrdlength? query.

Composition of Waveform Data

The MSO/DPO2000/B series introduces the concept of composite waveform data for analog channels. Sample data is always available for acquired analog channel waveforms. Composite waveforms are available in SAMPLE acquisition mode but are not available in AVERAGE acquisition mode. Peak data may be available depending on the instrument bandwidth, acquisition mode, nominal record length, and the horizontal scale settings.

- Sample waveform data which may be a component of a composite waveform is referred to as COMPOSITE_YT (vertical units versus time) data.
- Peak detect waveform data which may be a component of a composite waveform is referred to as COMPOSITE_ENV (envelope or peak detect min/max pairs versus time) data.
- Analog channel waveforms acquired in AVERAGE acquisition mode and MATH waveforms are not composite waveforms and are termed SINGULAR_YT which indicates that the waveform is a single waveform composed of data points representing vertical units versus time. The composition of digital waveform data (:DATa:SOUrce D0 through D15, DIGital) is SINGULAR_YT only.

The following table lists the various waveform data compositions:

Composition	Description
COMPOSITE_YT (analog channels)	This is sample waveform data in vertical units versus time. Its native width is 8-bits and is written as 8-bit data in saved ISF waveform files and reference waveforms. The data can be queried as 1- or 2-byte data in CURVe? queries by setting the data width to 1 or 2.
COMPOSITE_ENV (analog channels)	This is min/max pair waveform data with the minimum value preceding the maximum value. Its native width is 8-bits and is written as 8-bit data in saved ISF waveform files and reference waveforms. The data can be queried as 1- or 2-byte data in CURVe? queries by setting the data width to 1 or 2.
SINGULAR_YT (analog and math channels)	This is 16-bit waveform data which originates as averaged acquired waveform data or math data. It is written to saved ISF waveform files and reference waveforms as 16- bit data. The data can be queried as 1-or 2- byte data in CURVe? queries by setting the data width to 1 or 2.
SINGULAR_YT (digital channels)	This is sample waveform data in binary vertical units (0 or 1) versus time. Its native width is 8-bits for individual digital channels (D0 through D15) and 32 bits for the digital collection (D0 through D15, digital representation of CH1 through CH4). The only resolution available for digital waveform data is FULL.

The available waveform compositions, which can be obtained using the CURVe? or WAVFrm? queries, can be determined using the DATa:COMPosition:AVAILable? query. This query returns the available waveform compositions for the last acquired waveform for the source specified by the DATa:SOUrce command. The data returned by the DATa:COMPosition:AVAILable? query may be one of the following:

- COMPOSITE_YT
- COMPOSITE_YT, COMPOSITE_ENV
- SINGULAR_YT

The composition of the waveform data to be queried using the CURVe? or WAVFrm? queries is set using the DATa:COMPosition or WFMOutpre:COMPosition command. The compositions which can be set are:

- COMPOSITE_YT
- COMPOSITE_ENV
- SINGULAR_YT

If the composition set is not available with the current acquisition and horizontal settings, no response is returned for the CURVe? or WAVFrm? query and an event is set.

For example, if COMPOSITE_ENV data is not available and a CURVe? query is sent with the composition set to COMPOSITE_ENV, no data will be returned and an error event is set:

```
> :DATA:COMPOSITION:AVAILABLE?
COMPOSITE_YT
> DATA:COMPOSITION?
COMPOSITE_YT
> DATA:COMPOSITION
COMPOSITE_ENV
> *ESR? 0 > CURVE? {no data returned}
> *ESR?
20
> EVMSG?
221,"Settings conflict; Envelope waveform not available;
CURVE?" >
```

The composition of the waveform data to be sent to the instrument for transfer to the reference waveform memory using the CURVe command is set using the WFMInpre:COMPosition command. The compositions which can be set are:

- COMPOSITE_YT
- COMPOSITE_ENV
- SINGULAR_YT

Filtering of Waveform Data with FilterVu

The FilterVu feature uses digital signal processing (DSP) techniques to apply low pass filtering to the acquired waveform data for the analog channels. The FilterVu feature provides filtering of the displayed waveform so that you can see your signal without the effects of noise. This is the reduced resolution waveform. A side effect of filtering the reduced resolution waveform is that the full resolution waveform may be subjected to filtering to achieve the desired filtering of the

reduced resolution waveform. The filtering frequency of the full resolution waveform is not under your direct control. It is a function of the horizontal and acquisition settings and the selected FilterVu frequency. The 3 dB cutoff frequency for this filtering may be different for the LRL record and the thumbnail record. The actual filter frequency applied to the acquired (LRL) and displayed (thumbnail) waveforms are available in the waveform preamble:

- WFMOutpre:FILTERFreq?

  This query returns the 3 dB low pass filter cutoff frequency for:

  - The acquired (LRL) waveform when the DATa:RESOlution is set to `FULL`.
  - The displayed (thumbnail) waveform when the DATa:RESOlution is set to `REDUced`.

When waveforms are sent to the instrument using the CURVe command or recalled from ISF files, the filtering frequency is obtained from the waveform preamble:

- WFMInpre:FILTERFreq

  This command sets the 3 dB low pass filter cutoff frequency for the waveform being transferred to the instrument's reference waveform memory.

Only `FULL` resolution waveforms may be sent to the instrument for storage and display in reference waveform memory.

The filtering of reduced resolution waveforms is controlled using the FILTERVu:FREQuency command. The available low pass frequencies depends on the instrument bandwidth and the horizontal settings (scale and record length) and may be queried for the last acquired waveform, (as specified by the DATa:SOUrce command) using the FILTERVu:FREQuency:AVAILable? query. This query returns a comma separated list of available FilterVu frequencies in decreasing order for the last acquired source waveform . For example, on a 100 MHz bandwidth instrument at a full resolution record length of 1 M points and a horizontal scale of 20 microseconds per division, the query returns these values which apply to the reduced resolution (displayed) waveform: 100000000, 75000000, 42000000, 21000000, 11000000, 5500000, and 2200000.

The low pass filter frequency in effect can be queried using the FILTERVu:FREQuency? query.

FilterVu Controls for the Displayed Waveform

The FilterVu feature can be turned on or off using the DISplay:GLITch command and the display intensity of the glitch capture waveform can be controlled using the DISplay:INTENSITy:GLITch command.

Querying Acquisition Peak Detect Data from the Instrument

1. Use the DATa:COMPosition:AVAILable? query to determine if peak detect data (`COMPOSITE_ENV`) is available. This query returns a list of waveform data types that are available from the instrument under the current instrument settings:
   - `COMPOSITE_YT` for sample data
   - `COMPOSITE_ENV` for envelope or peak detect data
   - `SINGULAR_YT` for sample data
2. If `COMPOSITE_ENV` is available, set the data composition to `COMPOSITE_ENV` (:DATa:COMPosition COMPOSITE_ENV). Otherwise, the peak detect data is not available and cannot be queried from the instrument.
3. CURVe? will now return peak detect data.

Querying Acquisition YT Sample Data from the Instrument

The sequence used to query sample data is nearly identical to the one used for peak detect data. Set the data composition to `COMPOSITE_YT` or `SINGULAR_YT` (based on results of the DATa:COMPosition:AVAILable? query); CURVe? will now return sample data.

Selecting Acquisition Record Length

Like other oscilloscopes, the acquisition record length can be set using the HORizontal:RECOrdlength command. However, the actual number of points in the acquired waveform record may differ from the nominal record length established by the HORizontal:RECOrdlength setting, depending on the HORizontal:SCAle in effect at the time of the acquisition. The allowed values for setting the acquisition record length are 1,000,000 (1 million) and 100,000 (100 thousand) points. Depending on the horizontal scale in effect at the time of the acquisition, a nominal 1 million point record may be composed of 1.0 million or 1.25 million points and a nominal 100 thousand point record may be composed of 100 thousand or 125 thousand points. The actual number of points acquired for the source waveform (specified by the DATa:SOUrce command) may be obtained using the WFMOutpre:RECOrdlength? query.

For reduced resolution waveforms, the actual record length depends on the FilterVu frequency in effect in addition to the horizontal settings. The reduced resolution record lengths vary with the FilterVu frequency.

High Resolution and Envelope acquisition modes are not supported in MSO/DPO2000/B series instruments.

For the digital waveforms (D0 through D15), the actual record length may differ from the nominal record length established by the HORizontal:RECOrdlength setting, depending on the HORizontal:SCAle in effect at the time of the acquisition and whether digital waveforms are selected from one or both of the digital pods (whether digital channels are turned on for D0 through D7, and for D8 through D15). For faster horizontal scales, the actual record length may be half of the nominal record length when digital waveforms from both digital pods are

turned on. Use the WFMOutpre:RECOrdlength? query to determine the actual record length for digital waveforms.

Extensions to the DATA Commands

Like the MSO/DPO4000/B series, the MSO/DPO2000/B series provides the DATa:WIDth command that allows you to specify the data word width (in bytes) used when querying data from the instrument or when sending data to the instrument with the CURVe command. The following are the extensions to data commands:

- DATa:RESOlution
- DATa:COMPosition:AVAILable?
- DATa:COMPosition

WFMOutpre Parameters

The WFMOutpre? parameters correspond to the source waveform specified by the DATa:SOUrce command. The following are the extensions to Waveform Outpre commands:

- WFMOutpre:COMPosition
- WFMOutpre:FILTERFreq?
- WFMOutpre:RECOrdlength?

WFMInpre Parameters

The WFMInpre? parameters correspond to the destination reference waveform specified by the DATa:DESTination command. The following are the extensions to Waveform Inpre commands:

- WFMInpre:FILTERFreq
- WFMInpre:COMPosition

FilterVu Frequency Control Commands

The following are the FilterVu commands:

- FILTERVu:FREQuency
- FILTERVu:FREQuency:AVAILable?
- DISplay:GLITch
- DISplay:INTENSITy:GLITch

Appendix C: Reserved Words

This is a list of reserved words for your instrument. Capital letters identify the required minimum spelling.

```
*CAL
*CLS
*DDT
*ESE
*ESR
*IDN
*LRN
*OPC
*PSC
*PUD
*RCL
*RST
*SAV
*SRE
*STB
*TRG
*TST
*WAI
1
1CH
7
8
9
A
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
ABORt
ABOrt
ABSolute
AC
ACKMISS
ACQ
ACQLENGTH
ACQuire
ACTIVeprinter
ADD
ADDR10
ADDR7
ADDRANDDATA
ADDRess
ADDress
ALIas
ALIas[
ALL
ALLEv
ALLFields
ALLLines
ALTERNATE
ALWAYS
AMPlitude
AND
ANY
APPKey
AREa
ASCII
ASCIi
ASCii
ASSIgn
ATRIGger
AUTO
AUTOSet
AUTOZero
AUX
AUXOut
AUXin
AVAILable
AVErage
Auto
B
B0
B1
B2
B3
B4
B5
B6
B7
B8
B9
BACKLight
BACKWards
BANdwidth
BASe
BDIFFBP
BINary
BITOrder
BITRate
BIT_Nr
BLAckmanharris
BM
BMP
BN_Fmt
BOTh
BOX
BTRIGger
BURst
BUS
BUSY
BY
BYPass
BYT_Nr
BYT_Or
CALibrate
CAN
CANH
CANL
CARD
CAREa
CATalog
CH
CH1
CH2
CH3
CH4
CHannel
CHecksum
CLAss
CLEAR
CLEARMenu
CLEARSNapshot
CLEar
CLOCK
CLOCk
CM10BY15
CM13BY18
CM15BY21
CM18BY24
CM6BY8
CM7BY10
CM9BY13
CMEan
COLUMN
COMMAND
COMPOSITE_ENV
COMPOSITE_YT
COMPosition
CONDition
CONFigure
CONTROl
CONTinue
COPy
COUNt
COUPling
CPU
CR
CRC
CRCHeader
CRCTrailer
CREATE
CRMs
CROSSHair
CURSor
CURSors
CURVe
CURrent
CUSTom
CWD
CYCLEcount
D
D0
D1
D10
D11
D12
D13
D14
D15
D2
D3
D4
D5
D6
D7
D8
D9
DATA
```

DATABits
DATE
DATEPrint
DATa
DAll
DB
DC
DECimal
DEFLT
DEFault
DEFine
DEGAUss
DEGrees
DELEte
DELIMiter
DELTa
DELTatime
DELay
DELayed
DELete
DELta
DESE
DESKew
DESTination
DHCPbootp
DIAg
DIFFerential
DIGItal
DIGital
DIGtal
DIR
DIRection
DISabled
DISplay
DISplaymode
DNS
DOMAINname
DPOModels
DRAFT
DUAL
DUmp
DYNAMic
E
ECL
EDGE
EDGe
EEPROM
EITher
ENAble
ENCdg
END
ENET
ENGLish
ENV
ENVelope
EOF
EOFTYPE
EQUal
EQual
ERROR
ERRTYPE
ERRlog
ERRor
ETHERnet
EVEN
EVENT
EVENTS
EVENTtable
EVMsg
EVQty
EXECute
EXT
EXTended
EXTernal
FACtor
FACtory
FAIL
FALL
FALSe
FALling
FASTPHOTO
FASTer
FAStest
FFT
FIELD
FIFty
FILEFormat
FILESystem
FILTERFreq
FILTERVu
FINE
FIRst
FIVEdivs
FLAg
FOCUS
FOLder
FORCEDRange
FORCe
FORMAt
FORMat
FORWards
FPAnel
FRACTional
FRAMEID
FRAMEType
FRAMETypeid
FRAMEtype
FRAMING
FRAme
FREE
FREESpace
FRENch
FREQuency
FULL
FULl
FUNCtion
GAIN
GATEWay
GATIng
GATing
GENeralcall
GERMan
GLITch
GND
GPIBUsb
GRAticule
GRId
HAGAKIPC
HAGAKIPCARD
HALFgrat
HAMming
HANning
HARDCopy
HBArs
HD1080I50
HD1080I60
HD1080P24
HD1080P25
HD1080PSF24
HD480P60
HD720P60
HDtv
HEADER
HEADer
HEIght
HERtz
HEXadecimal
HFRej
HIGH
HIGHLimit
HIRes
HIStogram
HIVALue
HOLDTime
HOLDoff
HORZ
HORizontal
HPOS
HSmode
HTTPPort
I2C
ID
IDANDDATA
IDFORmat
IDLETime
IDLEtime
IDPrint
IDentifier
IMAGESize
IMAGe
IMAge
IMMed
IMPedance
IN
IN11BY17
IN2P5BY3P25
IN4BY6
IN8BY10
INDBits
INDICators
INDIvidual
INDependent
INFInite
INIT
INKSaver
INPut
INTENSITy
INTERLAced
INTERNal
INVERTed
INVert
INVerted
INrange
IO
IPADDress
ISCLOCKed
ITALian
JAPAnese
KOREan
L
L2
L4
LABEL
LABel
LANGuage
LANdscape
LARge
LAYout
LESSEQual
LESSLimit
LESSThan
LESSthan

LETTER
LEVel
LF
LFRej
LIN
LINE
LINEAr
LINEPeriod
LIST
LOCk
LOG
LOGIC
LOGIc
LOGic
LOOP
LOW
LOWLimit
LOWerthreshold
LSB
MAG
MAIN
MAIn
MAIn]
MARK
MARKSINCOLumn
MATH
MATH1
MATHVAR
MATH[1]
MAXSamplerate
MAXimum
MEAN
MEAS
MEASUrement
MEDium
MEG
MESSage
METHod
MID
MID2
MINImum
MINMax
MISO
MISOMOSI
MIXed
MKDir
MM100BY150
MM54BY86
MODE
MODe
MOREEQual
MORELimit
MORETHan
MOREthan
MOSI
MSB
NAME
NAMe]
NANd
NEGative
NEWpass
NEXT
NEXt
NO
NOCARE
NOISErej
NONE
NONe
NOPARity
NORMal
NOTCOMPuted
NR1
NRMAL
NR_Pt
NTIMES
NTIMes
NTSc
NULL
NULLFRDynamic
NULLFRStatic
NULl
NUMACq
NUMAVg
NUMENTries
NUMERic
NUMEnv
NUMHORZ
NUMVERT
OCCURS
ODD
OFF
OFFSet
ON
ONCE
ONFAIL
OPTion
OUT
OUTrange
OVERLoad
OWNer
PACKET
PAL
PAPERSize
PAPERType
PARallel
PARity
PASS
PASSWord
PATtern
PAYLength
PAYLoad
PERCent
PERIod
PERSistence
PERcent
PHAse
PHOTO
PictBridge
PING
PK2Pk
PLAIN
PNG
POLARity
POLar
POLarity
PORTRait
PORTUguese
POSITION
POSition
POSitive
POVershoot
POWerupstatus
PPULSECount
PRESS
PREVIEW
PREViewstate
PREVious
PRINTQual
PRINTer
PRODDELta
PRODUCT
PRODUCTID
PROGressive
PRObe
PT_Fmt
PT_ORder
PT_Off
PULSEWIDth
PULSEWidth
PULSe
PULse
PWIdth
QUALifier
RADIUS
RATDELta
RATE15K
RATE1M
RATE20K
RATE25K
RATE35K
RATE50K
RATE800K
RATIO
RDELta
READ
READFile
RECAll
RECOrdlength
RECTX
RECTY
RECTangular
REDUced
REF
REF1
REF2
REFLevel
REM
REMote
REName
REPEATstart
RESET
RESOlution
RESUlt
RESistance
RESults
RI
RIBinary
RISEFall
RISe
RISing
RMDir
RMS
ROLL100MM
ROLL127MM
ROLL210MM
ROLL89MM
ROM
RP
RPBinary
RS232
RS232C
RUN
RUNSTop
RUNT
RUNt
RUSSian
RWINClude
RX
RXDATA
RXENDPacket
RXPARity
RXSTArt

SAMPLERate
SAMPLEpoint
SAMple
SAVe
SCAN
SCAle
SCLK
SCREEN
SCREen
SDATA
SEARCH
SEARCHtotrigger
SECAM
SEConds
SELect
SELected
SEQuence
SERIALnumber
SERnumber
SET
SETHold
SETLevel
SETTime
SETUP
SETUp
SEVENTYFive
SHOW
SIGnal
SIMPlifiedchinese
SINGULAR_YT
SIZe
SLEEP
SLEWRate
SLOWer
SLOpe
SMAll
SNAPShot
SNAp
SOF
SOURCE
SOURCE2
SOURCE[1]
SOUrce
SOUrce2
SPANish
SPC
SPECTral
SPI
SPREADSheet
SPace
SRIbinary
SRPbinary
SS
STANDard
STANdard
START
STARTofframe
STARTup
STARTupnosync
STARt
STARtbyte
STATE
STATE]
STATIstics
STATUS
STATe
STATic
STATus
STDdev
STOP
STOPAfter
STRing
STYle
STandard
SUBNETMask
SYNC
SYNCField
SYNCFrame
SYNCInterval
TEKSecure
TEMPerature
TERmination
TESTnumber
THDELta
THETA
THREshold
THReshold
TIFf
TIME
TIMe
TOTAL
TOTal
TOTaluptime
TRACk
TRADitionalchinese
TRANsition
TRansition
TRIGger
TRIGgertosearch
TRUe
TTL
TURN
TWEnty
TWOfifty
TX
TXDATA
TXENDPacket
TXPARity
TXRX
TXSTArt
TYPE
TYPe
Than
UNDo
UNEQual
UNIts
UNLock
UPPerthreshold
USBDevice
USBTMC
USBTmc
USE
USER
V1X
V2X
VALue
VAR
VBArs
VDELTa
VENDORID
VERBose
VERT
VERTical
VIDeo
VOLts
WAKEup
WAVEform
WAVFrm
WEIghting
WFId
WFMInpre
WFMOutpre
WHEn
WIDth
WINdow
WRITE
WRITEFile
X
XDELta
XFF
XINcr
XUNit
XY
XZEro
Y
YDELta
YES
YMUlt
YOFf
YT
YUNIts
YUNit
YUNits
YZEro
ZOOM
ZOOm

Appendix D: Application Module-enabled Commands

This table lists the commands that become available with installation of an application module.

Application Module	Commands
DPO2AUTO	Automotive Serial Triggering and Analysis Module. Enables triggering on packet-level information on CAN and LIN buses as well as analytical tools such as digital views of the signal, bus views, packet decoding, search tools, and packet decode tables with time-stamp information.
	BUS:B<x>:CAN:BITRate
	BUS:B<x>:CAN:PRObe
	BUS:B<x>:CAN:SAMPLEpoint
	BUS:B<x>:CAN:SOUrce
	BUS:B<x>:LIN:BITRate
	BUS:B<x>:LIN:IDFORmat
	BUS:B<x>:LIN:POLARity
	BUS:B<x>:LIN:SAMPLEpoint
	BUS:B<x>:LIN:SOUrce
	BUS:B<x>:LIN:STANDard
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:CONDition
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:DIRection
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:SIZe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:FRAMEtype
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:CONDition
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:SIZe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:ERRTYPE
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue
	TRIGger:A:BUS:B<x>:CAN:CONDition
	TRIGger:A:BUS:B<x>:CAN:DATa:DIRection
	TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier
	TRIGger:A:BUS:B<x>:CAN:DATa:SIZe
	TRIGger:A:BUS:B<x>:CAN:DATa:VALue
	TRIGger:A:BUS:B<x>:CAN:FRAMEtype
	TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe
	TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue
	TRIGger:A:BUS:B<x>:LIN:CONDition
	TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue
	TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier
	TRIGger:A:BUS:B<x>:LIN:DATa:SIZe
	TRIGger:A:BUS:B<x>:LIN:DATa:VALue
	TRIGger:A:BUS:B<x>:LIN:ERRTYPE
	TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue

Application Module	Commands
DPO2COMP	Computer Serial Triggering and Analysis Module. Enables triggering on packet-level information on RS-232/422/485/UART buses as well as analytical tools such as digital views of the signal, bus views, packet decoding, search tools, and packet decode tables with time-stamp information.
	BUS:B<x>:RS232C:BITRate
	BUS:B<x>:RS232C:DATABits
	BUS:B<x>:RS232C:DELIMiter
	BUS:B<x>:RS232C:DISplaymode
	BUS:B<x>:RS232C:PARity
	BUS:B<x>:RS232C:POLarity
	BUS:B<x>:RS232C:RX:SOUrce
	BUS:B<x>:RS232C:TX:SOUrce
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:CONDition
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue
	TRIGger:A:BUS:B<x>:RS232C:CONDition
	TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe
	TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue
	TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe
	TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue

Application Module	Commands
DPO2EMBD	Embedded Serial Triggering and Analysis Module. Enables triggering on packet-level information on I2C and SPI buses as well as analytical tools such as digital views of the signal, bus views, packet decoding, search tools, and packet decode tables with time-stamp information.
	BUS:B<x>:I2C:ADDRess:RWINClude
	BUS:B<x>:I2C{:CLOCK|:SCLK}:SOUrce
	BUS:B<x>:I2C{:DATA|:SDATA}:SOUrce
	BUS:B<x>:SPI:BITOrder
	BUS:B<x>:SPI{:CLOCK|:SCLK}:POLARity
	BUS:B<x>:SPI{:CLOCK|:SCLK}:SOUrce
	BUS:B<x>:SPI:DATA{:IN|:MISO}:POLARity
	BUS:B<x>:SPI:DATA{:IN|:MISO}:SOUrce
	BUS:B<x>:SPI:DATA{:OUT|:MOSI}:POLARity
	BUS:B<x>:SPI:DATA{:OUT|:MOSI}:SOUrce
	BUS:B<x>:SPI:DATA:SIZe
	BUS:B<x>:SPI:FRAMING
	BUS:B<x>:SPI:IDLETime
	BUS:B<x>:SPI{:SELect|:SS}:POLARity
	BUS:B<x>:SPI{:SELect|:SS}:SOUrce
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:CONDition
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:DIRection
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:SIZe
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:CONDition
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MISO|:IN}:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MOSI|:OUT}:VALue
	SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa:SIZe
	TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe
	TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe
	TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue
	TRIGger:A:BUS:B<x>:I2C:CONDition
	TRIGger:A:BUS:B<x>:I2C:DATa:DIRection
	TRIGger:A:BUS:B<x>:I2C:DATa:SIZe
	TRIGger:A:BUS:B<x>:I2C:DATa:VALue
	TRIGger:A:BUS:B<x>:SPI:CONDition
	TRIGger:A:BUS:B<x>:SPI:DATa{:IN|:MISO}:VALue
	TRIGger:A:BUS:B<x>:SPI:DATa{:OUT|:MOSI}:VALue
	TRIGger:A:BUS:B<x>:SPI:DATa:SIZe

Appendix E: Search and Trigger Command Sequence Examples

The following are some example command sequences that show a variety of different searches and triggers. The commands in these sequences are not order-dependent.

To use these examples, connect channel 1, channel 2, channel 3 and channel 4 (if available) to the probe compensation signal located on the right hand side of the front panel.

The search and trigger command group sections contain more information on general search and trigger concepts.

Example 1: Single Threshold Edge Search

Goal: Search the channel 2 waveform and place a mark at each instance where it crosses below a threshold of 1.4 volts.

Command	Comment
*RST	Resets the oscilloscope. Wait for the reset to complete (approximately 3 seconds).
:SELect:CH2 1	Turns the CH2 waveform on.
:AUTOSet EXECute	Autosets the displayed waveform CH2. Wait for the autoset to complete (approximately 3 seconds).
:SEARCH:SEARCH1:TRIGger:A:TYPe EDGE	Specifies that this will be an edge search (a mark will be placed when the source waveform passes through a specified threshold level in the specified direction).
:SEARCH:SEARCH1:TRIGger:A:EDGE:SOUrce CH2	Specifies the CH2 waveform as the source waveform.
:SEARCH:SEARCH1:TRIGger:A:LOWerthreshold:CH2 TTL	Specifies 1.4 volts as the threshold level.
:SEARCH:SEARCH1:TRIGger:A:EDGE:SLOpe FALL	Specifies the falling edge as the direction.
:SEARCH:SEARCH1:STATE 1	Turns the search on.
:SEARCH:SEARCH1:TOTal?	Returns 4, indicating that CH2 fell below the 1.4 volt threshold 4 times.

Example 2: Single Threshold Edge Trigger

Goal: Trigger on the channel 2 waveform when the waveform crosses below a threshold of 1.4 volts.

Command	Comment
*RST	Resets the oscilloscope. Wait for the reset to complete (approximately 3 seconds).
:SELect:CH2 1	Turns the CH2 waveform on.

Command	Comment
:TRIGger:A:TYPe EDGE	Specifies that this will be an edge trigger (trigger will occur when the source waveform passes through a specified threshold level in the specified direction).
:TRIGger:A:EDGE:SOUrce CH2	Specifies the CH2 waveform as the source waveform.
:TRIGger:A:LOWerthreshold:CH2 1.4	Specifies 1.4 volts as the threshold level.
:TRIGger:A:EDGE:SLOpe FALL	Specifies as falling edge as the direction.
:TRIGger:STATE?	Should return TRIGGER, not AUTO.

Example 3: Dual Threshold Runt Search

Goal: Search the channel 3 waveform for negative runt pulses and place a mark at each instance when the waveform drops below an upper threshold of 1.4 volts, but does not cross a lower threshold of -2 volts before re-crossing the upper threshold. The pulse width must be less than 600 microseconds.

Command	Comment
*RST	Resets the oscilloscope. Wait for the reset to complete (approximately 3 seconds).
:SELect:CH3 1	Turns the CH3 waveform on.
:AUTOSet EXECute	Autosets the displayed waveform CH3. Wait for the autoset to complete (approximately 3 seconds).
:SEARCH:SEARCH1:TRIGger:A:TYPe RUNT	Specifies that this will be a runt search (a mark will be placed on a pulse amplitude that crosses one threshold but fails to cross a second threshold before re-crossing the first).
:SEARCH:SEARCH1:TRIGger:A:EDGE:SOUrce CH3	Specifies to use channel 3 as the source waveform.
:SEARCH:SEARCH1:TRIGger:A:LOWerthreshold:CH3 -2	Specifies to use -2 volts as the lower threshold.
:SEARCH:SEARCH1:TRIGger:A:UPPerthreshold: CH3 1.4	Specifies to use 1.4 volts as the upper threshold.
:SEARCH:SEARCH1:TRIGger:A:RUNT:POLarity NEGative	Specifies to search for when the runt polarity is negative.
:SEARCH:SEARCH1:TRIGger:A:RUNT:WIDth 600E-6	Specifies a pulse width of 600E-6 seconds.
:SEARCH:SEARCH1:TRIGger:A:RUNT:WHEn LESSthan	Specifies to search for when the pulse width is less than 600E-6 seconds.
:SEARCH:SEARCH1:STATE 1	Turns the search on.
:SEARCH:SEARCH1:TOTal?	Returns 3, indicating that 3 total negative runt pulses were less than 600E-6 seconds wide

Example 4: Single Threshold Logic Search on Three Waveforms

Goal: Search the channel 1, 2 and 3 waveforms and place a mark at each instance when either channel 1 is above 1.4 volts, channel 2 is above 1.5 volts, or channel 3 is above 1.3 volts.

Command	Comment
*RST	Resets the oscilloscope. Wait for the reset to complete (approximately 3 seconds).
:SELect:CH1 1	Turns the CH1 waveform on.
:SELect:CH2 1	Turns the CH2 waveform on.
:SELect:CH3 1	Turns the CH3 waveform on.
:AUTOSet EXECute	Autosets the displayed waveforms CH1, CH2 and CH3. Wait for the autoset to complete (approximately 3 seconds).
:SEARCH:SEARCH1:TRIGger:A:TYPe LOGIc	Specifies that this will be a logic search (a mark will be placed when all channels transition to the specified state).
:SEARCH:SEARCH1:TRIGger:A:LOGIc:INPut:CH1 HIGH	Specifies the Boolean logic criteria for channel 1; in this case, high.
:SEARCH:SEARCH1:TRIGger:A:LOGIc:INPut:CH2 HIGH	Specifies the Boolean logic criteria for channel 2; in this case, high.
:SEARCH:SEARCH1:TRIGger:A:LOGIc:INPut:CH3 HIGH	Specifies the Boolean logic criteria for channel 3; in this case, high.
:SEARCH:SEARCH1:TRIGger:A:LOGIc:THReshold:CH1 1.4	Specifies to use 1.4 volts as the threshold for CH1.
:SEARCH:SEARCH1:TRIGger:A:LOGIc:THReshold:CH2 1.5	Specifies to use 1.5 volts as the threshold for CH2.
:SEARCH:SEARCH1:TRIGger:A:LOGIc:THReshold:CH3 1.3	Specifies to use 1.3 volts as the threshold for CH3.
:SEARCH:SEARCH1:TRIGger:A:LOGIc:PATtern:WHEn TRUE	Specifies the condition for generating a logic pattern search; in this case, true.
:SEARCH:SEARCH1:TRIGger:A:LOGIc:FUNCtion OR	Specifies the logic operator for the logic search; in this case, OR.
:SEARCH:SEARCH1:STATE 1	Turns the search on.
:SEARCH:SEARCH1:TOTal?	Returns 3 or 4, indicating 3 or 4 times when one of channels 1, 2 or 3 became high. Note: Depending upon the number of transitions displayed, you may get 3 or 4 search marks for this example. If you get 3 search marks, try adjusting the horizontal position until you see 4 search marks.

Index

A

ACQuire:MAXSamplerate?, 2-59
ACQuire:MODe, 2-59
ACQuire:NUMACq?, 2-60
ACQuire:NUMAVg, 2-61
ACQuire:STATE, 2-61
ACQuire:STOPAfter, 2-63
ACQuire?, 2-59
Acquisition Command Group, 2-11
Alias Command Group, 2-12
ALIas, 2-63
ALIas:CATalog?, 2-64
ALIas:DEFine, 2-64
ALIas:DELEte, 2-65
ALIas:DELEte:ALL, 2-65
ALIas:DELEte[:NAMe], 2-66
ALIas[:STATE], 2-66
ALLEv?, 2-67
AUTOSet, 2-67
AUTOSet:ENAble, 2-68
AUXin:PRObe, 2-68
AUXin:PRObe:AUTOZero, 2-69
AUXin:PRObe:COMMAND, 2-69
AUXin:PRObe:DEGAUss, 2-70
AUXin:PRObe:DEGAUss:STATE?, 2-70
AUXin:PRObe:FORCEDRange, 2-70
AUXin:PRObe:GAIN, 2-71
AUXin:PRObe:ID:SERnumber?, 2-71
AUXin:PRObe:ID:TYPE?, 2-71
AUXin:PRObe:RESistance?, 2-71
AUXin:PRObe:SIGnal, 2-72
AUXin:PRObe:UNIts?, 2-72
AUXin?, 2-68

B

BUS, 2-73
BUS:B<x>:CAN:BITRate, 2-73
BUS:B<x>:CAN:PRObe, 2-73
BUS:B<x>:CAN:SAMPLEpoint, 2-74
BUS:B<x>:CAN:SOUrce, 2-74
BUS:B<x>:DISplay:FORMAt, 2-75
BUS:B<x>:I2C:ADDRess:RWINClude, 2-75
BUS:B<x>:I2C{:CLOCK|:SCLK}:SOUrce, 2-76
BUS:B<x>:I2C{:DATA|:SDATA}:SOUrce, 2-76
BUS:B<x>:LABel, 2-77
BUS:B<x>:LIN:BITRate, 2-77
BUS:B<x>:LIN:IDFORmat, 2-77
BUS:B<x>:LIN:POLARity, 2-78
BUS:B<x>:LIN:SAMPLEpoint, 2-78
BUS:B<x>:LIN:SOUrce, 2-79
BUS:B<x>:LIN:STANDard, 2-79
BUS:B<x>:PARallel:BIT<x>:SOUrce, 2-80
BUS:B<x>:PARallel:CLOCK:EDGE, 2-80
BUS:B<x>:PARallel:CLOCK:ISCLOCKed, 2-81
BUS:B<x>:PARallel:CLOCK:SOUrce, 2-81
BUS:B<x>:PARallel:WIDth, 2-81
BUS:B<x>:POSition, 2-82
BUS:B<x>:RS232C:BITRate, 2-82
BUS:B<x>:RS232C:DATABits, 2-83
BUS:B<x>:RS232C:DELIMiter, 2-83
BUS:B<x>:RS232C:DISplaymode, 2-83
BUS:B<x>:RS232C:PARity, 2-84
BUS:B<x>:RS232C:POLarity, 2-84
BUS:B<x>:RS232C:RX:SOUrce, 2-85
BUS:B<x>:RS232C:TX:SOUrce, 2-85
BUS:B<x>:SPI:BITOrder, 2-86
BUS:B<x>:SPI:DATA:SIZe, 2-88
BUS:B<x>:SPI:DATA{:IN|:MISO}:POLARity, 2-87
BUS:B<x>:SPI:DATA{:IN|:MISO}:SOUrce, 2-87
BUS:B<x>:SPI:DATA{:OUT|:MOSI}:
POLARity, 2-88
BUS:B<x>:SPI:DATA{:OUT|:MOSI}:SOUrce, 2-88
BUS:B<x>:SPI:FRAMING, 2-89
BUS:B<x>:SPI:IDLETime, 2-89
BUS:B<x>:SPI{:CLOCK|:SCLK}:POLARity, 2-86
BUS:B<x>:SPI{:CLOCK|:SCLK}:SOUrce, 2-86
BUS:B<x>:SPI{:SELect|:SS}:POLARity, 2-90
BUS:B<x>:SPI{:SELect|:SS}:SOUrce, 2-90
BUS:B<x>:STATE, 2-91
BUS:B<x>:TYPE, 2-91
BUS:LOWerthreshold:CH<x>, 2-92
BUS:THReshold:CH<x>, 2-92
BUS:THReshold:D<x>, 2-93
BUS:UPPerthreshold:CH<x>, 2-93
BUSY?, 2-94

C

*CAL?, 2-95
CALibrate:FACtory, 2-95
CALibrate:FACtory:STATus?, 2-96
CALibrate:INTERNal, 2-96
CALibrate:INTERNal:STARt, 2-96
CALibrate:INTERNal:STATus?, 2-97
CALibrate:RESults:FACtory?, 2-98
CALibrate:RESults:SPC?, 2-98
CALibrate:RESults?, 2-97
CALibrate:TEMPerature?, 2-98
Calibration and Diagnostic Command Group, 2-16
CH<x>:BANdwidth, 2-99
CH<x>:COUPling, 2-99
CH<x>:DESKew, 2-100
CH<x>:IMPedance, 2-100
CH<x>:INVert, 2-101
CH<x>:LABel, 2-101
CH<x>:OFFSet, 2-102
CH<x>:POSition, 2-102
CH<x>:PRObe:AUTOZero, 2-103
CH<x>:PRObe:COMMAND, 2-104
CH<x>:PRObe:DEGAUss, 2-104
CH<x>:PRObe:DEGAUss:STATE?, 2-104
CH<x>:PRObe:FORCEDRange, 2-105
CH<x>:PRObe:GAIN, 2-105
CH<x>:PRObe:ID:SERnumber?, 2-106
CH<x>:PRObe:ID:TYPE?, 2-107
CH<x>:PRObe:ID?, 2-106
CH<x>:PRObe:RESistance?, 2-107
CH<x>:PRObe:SIGnal, 2-107
CH<x>:PRObe:UNIts?, 2-108
CH<x>:PRObe?, 2-103
CH<x>:SCAle, 2-108
CH<x>:TERmination, 2-109
CH<x>:VOLts, 2-109
CH<x>:YUNits, 2-110
CH<x>?, 2-99
CLEARMenu, 2-110
*CLS, 2-110
Command Groups, 2-11
Cursor Command Group, 2-17
CURSor:FUNCtion, 2-111
CURSor:HBArs:DELTa?, 2-112
CURSor:HBArs:POSITION<x>, 2-113
CURSor:HBArs:UNIts, 2-113
CURSor:HBArs:USE, 2-114
CURSor:HBArs?, 2-112
CURSor:MODe, 2-114
CURSor:VBArs:ALTERNATE<x>?, 2-115
CURSor:VBArs:DELTa?, 2-116
CURSor:VBArs:HPOS<x>?, 2-116
CURSor:VBArs:POSITION<x>, 2-117
CURSor:VBArs:UNIts, 2-117
CURSor:VBArs:USE, 2-118
CURSor:VBArs:VDELTa?, 2-118
CURSor:VBArs?, 2-115
CURSor:XY:POLar:RADIUS:DELta?, 2-119
CURSor:XY:POLar:RADIUS:POSITION<x>?, 2-119
CURSor:XY:POLar:RADIUS:UNIts?, 2-119
CURSor:XY:POLar:THETA:DELta?, 2-120
CURSor:XY:POLar:THETA:POSITION<x>?, 2-120
CURSor:XY:POLar:THETA:UNIts?, 2-120
CURSor:XY:PRODUCT:DELta?, 2-120
CURSor:XY:PRODUCT:POSITION<x>?, 2-121
CURSor:XY:PRODUCT:UNIts?, 2-121
CURSor:XY:RATIO:DELta?, 2-121
CURSor:XY:RATIO:POSITION<x>?, 2-121
CURSor:XY:RATIO:UNIts?, 2-122
CURSor:XY:RECTangular:X:DELta?, 2-122
CURSor:XY:RECTangular:X:POSITION<x>, 2-122
CURSor:XY:RECTangular:X:UNIts?, 2-122
CURSor:XY:RECTangular:Y:DELta?, 2-123
CURSor:XY:RECTangular:Y:POSITION<x>, 2-123
CURSor:XY:RECTangular:Y:UNIts?, 2-123
CURSor?, 2-111
CURVe, 2-123

D

D<x>, 2-145
D<x>:LABel, 2-145
D<x>:POSition, 2-145
D<x>:THREshold, 2-146
DATa, 2-125
DATa:COMPosition, 2-126
DATa:COMPosition:AVAILable?, 2-126
DATa:DESTination, 2-127
DATa:ENCdg, 2-127
DATa:RESOlution, 2-129
DATa:SOUrce, 2-129
DATa:STARt, 2-131
DATa:STOP, 2-132
DATa:WIDth, 2-133
DATE, 2-133

*DDT, 2-134
DESE, 2-135
DIAg:LOOP:OPTion, 2-135
DIAg:LOOP:OPTion:NTIMes, 2-136
DIAg:LOOP:STOP, 2-136
DIAg:RESUlt:FLAg?, 2-137
DIAg:RESUlt:LOG?, 2-137
DIAg:SELect, 2-138
DIAg:SELect:<function>, 2-138
DIAg:STATE, 2-139
Display Command Group, 2-18
DISplay:CLOCk, 2-139
DISplay:DIGital:HEIght, 2-140
DISplay:FORMat, 2-140
DISplay:GLITch, 2-141
DISplay:GRAticule, 2-141
DISplay:INTENSITy:BACKLight, 2-142
DISplay:INTENSITy:GLITch, 2-143
DISplay:INTENSITy:GRAticule, 2-143
DISplay:INTENSITy:WAVEform, 2-143
DISplay:INTENSITy?, 2-142
DISplay:PERSistence, 2-144
DISplay?, 2-139

E

*ESE, 2-146
*ESR?, 2-147
Ethernet Command Group, 2-19
ETHERnet:DHCPbootp, 2-147
ETHERnet:DNS:IPADDress, 2-148
ETHERnet:DOMAINname, 2-148
ETHERnet:ENET:ADDress?, 2-149
ETHERnet:GATEWay:IPADDress, 2-149
ETHERnet:HTTPPort, 2-149
ETHERnet:IPADDress, 2-150
ETHERnet:NAME, 2-150
ETHERnet:PASSWord, 2-151
ETHERnet:PING, 2-151
ETHERnet:PING:STATUS?, 2-151
ETHERnet:SUBNETMask, 2-152
EVENT?, 2-152
EVMsg?, 2-153
EVQty?, 2-153

F

FACtory, 2-154
File System Command Group, 2-20
FILESystem, 2-155
FILESystem:CWD, 2-155
FILESystem:DELEte, 2-156
FILESystem:DIR?, 2-156
FILESystem:FORMat, 2-157
FILESystem:FREESpace?, 2-157
FILESystem:MKDir, 2-158
FILESystem:READFile, 2-158
FILESystem:REName, 2-159
FILESystem:RMDir, 2-159
FILESystem:WRITEFile, 2-160
FILTERVu:FREQuency, 2-160
FILTERVu:FREQuency:AVAILable?, 2-161
FPAnel:PRESS, 2-161
FPAnel:TURN, 2-163

G

GPIBUsb:ID?, 2-164

H

Hard copy Command Group, 2-22
HARDCopy, 2-164
HARDCopy:INKSaver, 2-165
HARDCopy:PREVIEW, 2-165
HEADer, 2-166
Horizontal Command Group, 2-23
HORizontal:ACQLENGTH?, 2-167
HORizontal:DELay:MODe, 2-167
HORizontal:DELay:TIMe, 2-168
HORizontal:DIGital:RECOrdlength:MAIN?, 2-168
HORizontal:DIGital:SAMPLERate:MAIN?, 2-168
HORizontal:POSition, 2-169
HORizontal:PREViewstate?, 2-169
HORizontal:RECOrdlength, 2-169
HORizontal:RESOlution, 2-170
HORizontal:SAMPLERate, 2-170
HORizontal:SCAle, 2-170
HORizontal:TRIGger:POSition, 2-171
HORizontal?, 2-166

I

ID?, 2-171
*IDN?, 2-172

L

LANGuage, 2-172
LOCk, 2-172
*LRN?, 2-173

M

Mark Command Group, 2-23
MARK, 2-174
MARK:CREATE, 2-174
MARK:DELEte, 2-175
MARK:FREE?, 2-175
MARK:SELected:END?, 2-176
MARK:SELected:FOCUS?, 2-176
MARK:SELected:MARKSINCOLumn?, 2-176
MARK:SELected:OWNer?, 2-176
MARK:SELected:SOURCE?, 2-177
MARK:SELected:STARt?, 2-177
MARK:SELected:STATe?, 2-177
MARK:SELected:ZOOm:POSition?, 2-177
MARK:TOTal?, 2-178
Math Command Group, 2-25
MATH[1]:DEFine, 2-179
MATH[1]:HORizontal:POSition, 2-180
MATH[1]:HORizontal:SCAle, 2-180
MATH[1]:HORizontal:UNIts, 2-181
MATH[1]:LABel, 2-178
MATH[1]:SPECTral:GATing:INDICators, 2-181
MATH[1]:SPECTral:GATing:INDICators: END?, 2-181
MATH[1]:SPECTral:GATing:INDICators: STARt?, 2-182
MATH[1]:SPECTral:MAG, 2-182
MATH[1]:SPECTral:NYQUISTFreq?, 2-182
MATH[1]:SPECTral:WINdow, 2-183
MATH[1]:TYPe, 2-183
MATH[1]:VERTical:POSition, 2-184
MATH[1]:VERTical:SCAle, 2-184
MATH[1]:VERTical:UNIts?, 2-185
MATH[1]?, 2-178
Measurement Command Group, 2-26
MEASUrement:CLEARSNapshot, 2-186
MEASUrement:GATing, 2-186
MEASUrement:IMMed:DELay:DIRection, 2-188
MEASUrement:IMMed:DELay:EDGE<x>, 2-188
MEASUrement:IMMed:DELay?, 2-187
MEASUrement:IMMed:SOUrce<x>, 2-190
MEASUrement:IMMed:SOUrce1, 2-189
MEASUrement:IMMed:SOUrce2, 2-190
MEASUrement:IMMed:TYPe, 2-191
MEASUrement:IMMed:UNIts?, 2-193
MEASUrement:IMMed:VALue?, 2-194
MEASUrement:IMMed?, 2-187
MEASUrement:INDICators:HORZ<x>?, 2-195
MEASUrement:INDICators:NUMHORZ?, 2-195
MEASUrement:INDICators:NUMVERT?, 2-196
MEASUrement:INDICators:STATE, 2-196
MEASUrement:INDICators:VERT<x>?, 2-197
MEASUrement:INDICators?, 2-195
MEASUrement:MEAS<x>:COUNt?, 2-197
MEASUrement:MEAS<x>:DELay:DIRection, 2-198
MEASUrement:MEAS<x>:DELay:EDGE<x>, 2-199
MEASUrement:MEAS<x>:DELay?, 2-198
MEASUrement:MEAS<x>:MAXimum?, 2-199
MEASUrement:MEAS<x>:MEAN?, 2-200
MEASUrement:MEAS<x>:MINImum?, 2-200
MEASUrement:MEAS<x>:SOURCE[1], 2-200
MEASUrement:MEAS<x>:SOUrce<x>, 2-201
MEASUrement:MEAS<x>:SOURCE2, 2-201
MEASUrement:MEAS<x>:STATE, 2-202
MEASUrement:MEAS<x>:STDdev?, 2-203
MEASUrement:MEAS<x>:TYPe, 2-203
MEASUrement:MEAS<x>:UNIts?, 2-206
MEASUrement:MEAS<x>:VALue?, 2-206
MEASUrement:MEAS<x>?, 2-197
MEASUrement:METHod, 2-207
MEASUrement:REFLevel:ABSolute:HIGH, 2-208
MEASUrement:REFLevel:ABSolute:LOW, 2-208
MEASUrement:REFLevel:ABSolute:MID, 2-209
MEASUrement:REFLevel:ABSolute:MID<x>, 2-210
MEASUrement:REFLevel:ABSolute:MID2, 2-210
MEASUrement:REFLevel:METHod, 2-211
MEASUrement:REFLevel:PERCent:HIGH, 2-211
MEASUrement:REFLevel:PERCent:LOW, 2-212
MEASUrement:REFLevel:PERCent:MID, 2-213
MEASUrement:REFLevel:PERCent:MID<x>, 2-214
MEASUrement:REFLevel:PERCent:MID2, 2-213
MEASUrement:REFLevel?, 2-207
MEASUrement:SNAPShot, 2-214
MEASUrement?, 2-185
MESSage, 2-215
MESSage:BOX, 2-215
MESSage:CLEAR, 2-216
MESSage:SHOW, 2-216
MESSage:STATE, 2-219

Miscellaneous Command Group, 2-29

N

NEWpass, 2-219

O

*OPC, 2-220

P

PASSWord, 2-221
PICTBridge:DATEPrint, 2-221
PICTBridge:DEFault, 2-222
PICTBridge:IDPrint, 2-222
PICTBridge:IMAGESize, 2-223
PICTBridge:PAPERSize, 2-224
PICTBridge:PAPERType, 2-225
PICTBridge:PRINTQual, 2-226
*PSC, 2-226
*PUD, 2-227

R

*RCL, 2-227
RECAll:SETUp, 2-228
RECAll:WAVEform, 2-228
REF<x>:DATE?, 2-229
REF<x>:HORizontal:DELay:TIMe, 2-230
REF<x>:HORizontal:SCAle, 2-230
REF<x>:LABel, 2-230
REF<x>:POSition, 2-231
REF<x>:SCAle, 2-231
REF<x>:TIMe?, 2-231
REF<x>:VERTical:POSition, 2-232
REF<x>:VERTical:SCAle, 2-232
REF<x>?, 2-229
REM, 2-233
*RST, 2-234

S

*SAV, 2-234
Save and Recall Command Group, 2-31
SAVe:ASSIgn:TYPe, 2-235
SAVe:EVENTtable:BUS<x>, 2-235
SAVe:IMAGe, 2-236
SAVe:IMAGe:FILEFormat, 2-236
SAVe:IMAGe:LAYout, 2-237
SAVe:SETUp, 2-237
SAVe:WAVEform, 2-238
SAVe:WAVEform:FILEFormat, 2-239
SAVe:WAVEform:GATIng, 2-240
SAVe:WAVEform:SPREADSheet:RESOlution, 2-240
Search Commands Group, 2-33
SEARCH:SEARCH<x>:COPy, 2-244
SEARCH:SEARCH<x>:STATE, 2-244
SEARCH:SEARCH<x>:TOTAL?, 2-244
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:CONDition, 2-247
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:DIRection, 2-247
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier, 2-248
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:SIZe, 2-249
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:DATa:VALue, 2-249
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN:FRAMEtype, 2-250
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:MODe, 2-250
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:VALue, 2-251
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe, 2-251
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe, 2-251
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue, 2-252
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:CONDition, 2-252
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:DIRection, 2-253
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:SIZe, 2-253
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:I2C:DATa:VALue, 2-254
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:CONDition, 2-254
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue, 2-255
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier, 2-255
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:SIZe, 2-256

SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:DATa:VALue, 2-257
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:ERRTYPE, 2-257
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue, 2-258
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:PARallel:VALue, 2-258
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:CONDition, 2-259
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:SIZe, 2-259
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:RX:DATa:VALue, 2-260
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe, 2-260
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue, 2-260
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:CONDition, 2-261
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa:SIZe, 2-262
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MISO|:IN}:VALue, 2-261
SEARCH:SEARCH<x>:TRIGger:A:BUS:B<x>:SPI:DATa{:MOSI|:OUT}:VALue, 2-262
SEARCH:SEARCH<x>:TRIGger:A:BUS:SOUrce, 2-262
SEARCH:SEARCH<x>:TRIGger:A:BUS?, 2-245
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SLOpe, 2-263
SEARCH:SEARCH<x>:TRIGger:A:EDGE:SOUrce, 2-263
SEARCH:SEARCH<x>:TRIGger:A:LEVel, 2-264
SEARCH:SEARCH<x>:TRIGger:A:LEVel:CH<x>, 2-264
SEARCH:SEARCH<x>:TRIGger:A:LEVel:MATH, 2-264
SEARCH:SEARCH<x>:TRIGger:A:LEVel:REF<x>, 2-265
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:FUNCtion, 2-265
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CH<x>, 2-265
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:EDGE, 2-266
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:CLOCk:SOUrce, 2-266
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:D<x>, 2-267
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:MATH, 2-267
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:INPut:REF<x>, 2-268
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:CH<x>, 2-268
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:D<x>, 2-268
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:MATH, 2-269
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:INPut:REF<x>, 2-269
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn, 2-270
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit, 2-270
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:PATtern:WHEn:MORELimit, 2-271
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:CH<x>, 2-271
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:MATH, 2-271
SEARCH:SEARCH<x>:TRIGger:A:LOGIc:THReshold:REF<x>, 2-272
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:CH<x>, 2-272
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:MATH, 2-273
SEARCH:SEARCH<x>:TRIGger:A:LOWerthreshold:REF<x>, 2-273
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:POLarity, 2-273
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:SOUrce, 2-274
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WHEn, 2-274
SEARCH:SEARCH<x>:TRIGger:A:PULSEWidth:WIDth, 2-275
SEARCH:SEARCH<x>:TRIGger:A:RUNT:POLarity, 2-275
SEARCH:SEARCH<x>:TRIGger:A:RUNT:SOUrce, 2-276

SEARCH:SEARCH<x>:TRIGger:A:RUNT:
WHEn, 2-276
SEARCH:SEARCH<x>:TRIGger:A:RUNT:
WIDth, 2-277
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
CLOCk:EDGE, 2-277
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
CLOCk:SOUrce, 2-277
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
CLOCk:THReshold, 2-278
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:
SOUrce, 2-278
SEARCH:SEARCH<x>:TRIGger:A:SETHold:DATa:
THReshold, 2-279
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
HOLDTime, 2-279
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
SETTime, 2-279
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
THReshold:CH<x>, 2-280
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
THReshold:REF<x>, 2-281
SEARCH:SEARCH<x>:TRIGger:A:SETHold:
THReshold{:MATH|:MATH1}, 2-280
SEARCH:SEARCH<x>:TRIGger:A:TYPe, 2-283
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:
CH<x>, 2-283
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:
MATH, 2-284
SEARCH:SEARCH<x>:TRIGger:A:UPPerthreshold:
REF<x>, 2-284
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:
RISEFall}:DELTatime, 2-281
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:
RISEFall}:POLarity, 2-281
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:
RISEFall}:SOUrce, 2-282
SEARCH:SEARCH<x>:TRIGger:A{:TRANsition|:
RISEFall}:WHEn, 2-282
SEARCH?, 2-241
SELect, 2-284
SELect:BUS<x>, 2-285
SELect:CH<x>, 2-285
SELect:CONTROl, 2-286
SELect:D<x>, 2-286
SELect:MATH[1], 2-287
SELect:REF<x>, 2-288
SET?, 2-288
SETUP<x>:DATE?, 2-289
SETUP<x>:LABEL, 2-290
SETUP<x>:TIME?, 2-290
*SRE, 2-290
Status and Error Command Group, 2-37
*STB?, 2-291

T

TEKSecure, 2-291
TIME, 2-292
TOTaluptime?, 2-293
*TRG, 2-293
Trigger Command Group, 2-38
TRIGger, 2-294
TRIGger:A, 2-294
TRIGger:A:BUS, 2-295
TRIGger:A:BUS:B<x>:CAN:CONDition, 2-296
TRIGger:A:BUS:B<x>:CAN:DATa:DIRection, 2-297
TRIGger:A:BUS:B<x>:CAN:DATa:QUALifier, 2-297
TRIGger:A:BUS:B<x>:CAN:DATa:SIZe, 2-298
TRIGger:A:BUS:B<x>:CAN:DATa:VALue, 2-298
TRIGger:A:BUS:B<x>:CAN:FRAMEtype, 2-299
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:
MODe, 2-299
TRIGger:A:BUS:B<x>:CAN{:IDentifier|:ADDRess}:
VALue, 2-300
TRIGger:A:BUS:B<x>:I2C:ADDRess:MODe, 2-300
TRIGger:A:BUS:B<x>:I2C:ADDRess:TYPe, 2-301
TRIGger:A:BUS:B<x>:I2C:ADDRess:VALue, 2-301
TRIGger:A:BUS:B<x>:I2C:CONDition, 2-302
TRIGger:A:BUS:B<x>:I2C:DATa:DIRection, 2-303
TRIGger:A:BUS:B<x>:I2C:DATa:SIZe, 2-303
TRIGger:A:BUS:B<x>:I2C:DATa:VALue, 2-304
TRIGger:A:BUS:B<x>:LIN:CONDition, 2-304
TRIGger:A:BUS:B<x>:LIN:DATa:HIVALue, 2-305
TRIGger:A:BUS:B<x>:LIN:DATa:QUALifier, 2-305
TRIGger:A:BUS:B<x>:LIN:DATa:SIZe, 2-306
TRIGger:A:BUS:B<x>:LIN:DATa:VALue, 2-306
TRIGger:A:BUS:B<x>:LIN:ERRTYPE, 2-307
TRIGger:A:BUS:B<x>:LIN:IDentifier:VALue, 2-308
TRIGger:A:BUS:B<x>:PARallel:VALue, 2-308
TRIGger:A:BUS:B<x>:RS232C:CONDition, 2-308
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:
SIZe, 2-309
TRIGger:A:BUS:B<x>:RS232C:RX:DATa:
VALue, 2-309

TRIGger:A:BUS:B<x>:RS232C:TX:DATa:SIZe, 2-310
TRIGger:A:BUS:B<x>:RS232C:TX:DATa:VALue, 2-310
TRIGger:A:BUS:B<x>:SPI:CONDition, 2-311
TRIGger:A:BUS:B<x>:SPI:DATa:SIZe, 2-312
TRIGger:A:BUS:B<x>:SPI:DATa{:IN|:MISO}:VALue, 2-311
TRIGger:A:BUS:B<x>:SPI:DATa{:OUT|:MOSI}:VALue, 2-312
TRIGger:A:BUS:SOUrce, 2-312
TRIGger:A:EDGE:COUPling, 2-313
TRIGger:A:EDGE:SLOpe, 2-314
TRIGger:A:EDGE:SOUrce, 2-314
TRIGger:A:EDGE?, 2-313
TRIGger:A:HOLDoff:TIMe, 2-315
TRIGger:A:HOLDoff?, 2-315
TRIGger:A:LEVel, 2-316
TRIGger:A:LEVel:AUXin, 2-316
TRIGger:A:LEVel:CH<x>, 2-317
TRIGger:A:LEVel:D<x>, 2-317
TRIGger:A:LOGIc:CLAss, 2-318
TRIGger:A:LOGIc:FUNCtion, 2-319
TRIGger:A:LOGIc:INPut:CH<x>, 2-320
TRIGger:A:LOGIc:INPut:CLOCk:EDGE, 2-320
TRIGger:A:LOGIc:INPut:CLOCk:SOUrce, 2-321
TRIGger:A:LOGIc:INPut:D<x>, 2-321
TRIGger:A:LOGIc:INPut?, 2-320
TRIGger:A:LOGIc:PATtern:DELTatime, 2-322
TRIGger:A:LOGIc:PATtern:INPut:D<x>, 2-322
TRIGger:A:LOGIc:PATtern:WHEn, 2-323
TRIGger:A:LOGIc:PATtern:WHEn:LESSLimit, 2-324
TRIGger:A:LOGIc:PATtern:WHEn:MORELimit, 2-324
TRIGger:A:LOGIc:PATtern?, 2-322
TRIGger:A:LOGIc:THReshold:(CH<x>|D<x>), 2-325
TRIGger:A:LOGIc?, 2-318
TRIGger:A:LOWerthreshold:CH<x>, 2-325
TRIGger:A:LOWerthreshold{:EXT|:AUX}, 2-326
TRIGger:A:MODe, 2-326
TRIGger:A:PULse:CLAss, 2-327
TRIGger:A:PULse?, 2-327
TRIGger:A:PULSEWidth:POLarity, 2-328
TRIGger:A:PULSEWidth:SOUrce, 2-328
TRIGger:A:PULSEWidth:WHEn, 2-329
TRIGger:A:PULSEWidth:WIDth, 2-330
TRIGger:A:PULSEWIDth?, 2-328
TRIGger:A:RUNT:POLarity, 2-330
TRIGger:A:RUNT:SOUrce, 2-331
TRIGger:A:RUNT:WHEn, 2-331
TRIGger:A:RUNT:WIDth, 2-332
TRIGger:A:RUNT?, 2-330
TRIGger:A:SETHold:CLOCk:EDGE, 2-333
TRIGger:A:SETHold:CLOCk:SOUrce, 2-334
TRIGger:A:SETHold:CLOCk:THReshold, 2-334
TRIGger:A:SETHold:CLOCk?, 2-333
TRIGger:A:SETHold:DATa:SOUrce, 2-335
TRIGger:A:SETHold:DATa:THReshold, 2-336
TRIGger:A:SETHold:DATa?, 2-335
TRIGger:A:SETHold:HOLDTime, 2-337
TRIGger:A:SETHold:SETTime, 2-337
TRIGger:A:SETHold:THReshold:CH<x>, 2-337
TRIGger:A:SETHold:THReshold:D<x>, 2-338
TRIGger:A:SETHold?, 2-333
TRIGger:A:TYPe, 2-341
TRIGger:A:UPPerthreshold:CH<x>, 2-342
TRIGger:A:VIDeo:HOLDoff:FIELD, 2-343
TRIGger:A:VIDeo:LINE, 2-343
TRIGger:A:VIDeo:POLarity, 2-344
TRIGger:A:VIDeo:SOUrce, 2-344
TRIGger:A:VIDeo:STANdard, 2-345
TRIGger:A:VIDeo?, 2-342
TRIGger:A:VIDeo{:SYNC|:FIELD}, 2-345
TRIGger:A{:TRANsition|:RISEFall}:DELTatime, 2-339
TRIGger:A{:TRANsition|:RISEFall}:POLarity, 2-339
TRIGger:A{:TRANsition|:RISEFall}:SOUrce, 2-340
TRIGger:A{:TRANsition|:RISEFall}:WHEn, 2-340
TRIGger:A{:TRANsition|:RISEFall}?, 2-338
TRIGger:EXTernal:PRObe, 2-346
TRIGger:EXTernal:YUNIts?, 2-347
TRIGger:EXTernal?, 2-346
TRIGger:FREQuency?, 2-347
TRIGger:STATE?, 2-347
*TST?, 2-348

U

UNLock, 2-348
USBDevice:CONFigure, 2-349
USBTMC, 2-349
USBTMC:PRODUCTID:DECimal?, 2-349
USBTMC:PRODUCTID:HEXadecimal?, 2-350

USBTMC:SERIALnumber?, 2-350
USBTMC:VENDORID:DECimal?, 2-350
USBTMC:VENDORID:HEXadecimal?, 2-350

V

VERBose, 2-351
Vertical Command Group, 2-47

W

*WAI, 2-352
Waveform Transfer Command Group, 2-50
WAVFrm?, 2-352
WFMInpre:BIT_Nr, 2-353
WFMInpre:BN_Fmt, 2-354
WFMInpre:BYT_Nr, 2-354
WFMInpre:BYT_Or, 2-355
WFMInpre:COMPosition, 2-355
WFMInpre:ENCdg, 2-356
WFMInpre:FILTERFreq, 2-356
WFMInpre:NR_Pt, 2-357
WFMInpre:PT_Fmt, 2-357
WFMInpre:PT_Off, 2-358
WFMInpre:XINcr, 2-358
WFMInpre:XUNit, 2-359
WFMInpre:XZEro, 2-360
WFMInpre:YMUlt, 2-360
WFMInpre:YOFf, 2-361
WFMInpre:YUNit, 2-362
WFMInpre:YZEro, 2-362
WFMInpre?, 2-353
WFMOutpre:BIT_Nr, 2-364
WFMOutpre:BN_Fmt, 2-364
WFMOutpre:BYT_Nr, 2-365
WFMOutpre:BYT_Or, 2-365
WFMOutpre:COMPosition, 2-366
WFMOutpre:ENCdg, 2-366
WFMOutpre:FILTERFreq?, 2-367
WFMOutpre:NR_Pt?, 2-367
WFMOutpre:PT_Fmt?, 2-368
WFMOutpre:PT_Off?, 2-368
WFMOutpre:PT_ORder?, 2-369
WFMOutpre:RECOrdlength?, 2-369
WFMOutpre:WFId?, 2-369
WFMOutpre:XINcr?, 2-371
WFMOutpre:XUNit?, 2-371
WFMOutpre:XZEro?, 2-371
WFMOutpre:YMUlt?, 2-372
WFMOutpre:YOFf?, 2-372
WFMOutpre:YUNit?, 2-373
WFMOutpre:YZEro?, 2-373
WFMOutpre?, 2-363

Z

Zoom Command Group, 2-57
ZOOm:ZOOM1:FACtor?, 2-375
ZOOm:ZOOM1:HORizontal:POSition, 2-375
ZOOm:ZOOM1:HORizontal:SCAle, 2-376
ZOOm:ZOOM1:POSition, 2-376
ZOOm:ZOOM1:SCAle, 2-376
ZOOm:ZOOM1:STATE, 2-377
ZOOm:ZOOM1?, 2-375
ZOOm?, 2-374
ZOOm{:MODe|:STATE}, 2-374